GT CSE6250 Big Data Healthcare

Instructor: Jimeng Sun

Description

In healthcare, large amounts of heterogeneous medical data have become available in various healthcare organizations (payers, providers, pharmaceuticals). This data could be an enabling resource for deriving insights for improving care delivery and reducing waste. The enormity and complexity of these data-sets present great challenges in analyses and subsequent applications to a practical clinical environment. In this course, we introduce the characteristics of medical data and associated data mining challenges on dealing with such data. We cover various algorithms and systems for big data analytics. We focus on studying those big data techniques in the context of concrete healthcare analytic applications such as predictive modeling, computational phenotyping, and patient similarity. We also study big data analytic technology:

1. Scalable machine learning algorithms such as online learning and fast similarity search;
2. Big data analytic system such as Hadoop family (Hive, Pig, HBase), Spark and Graph DB

Table of Contents

01 Introduction to Big Data

0 Welcome to Big Data Analytics for Healthcare. I'm Jimeng Sun, associate professor in college computing at Georgia Tech. I'm here at Children's Healthcare, Atlanta to tell you a little bit about this course. First, let me tell you a little bit about me. I was born in Beijing, China. No? I'll skip ahead. My research area is in healthcare analytics and data mining. Before Georgia Tech, I worked at IBM TJ Watson Research Center in Healthcare Informatics. I also enjoy swimming. In this lesson, we'll discuss what you'll do for this course and what you will learn, and why you should care.

1 This course exists at the intersection of house care and big data. On the one hand, we have house care, we talk about house care applications as well as house care data. On the other hand, we have data science and big data analytics. We'll talk about different algorithms. And systems for processing and analyzing big data. We'll focus on the intersection between these two and how one is applied to the other.

2 Our learning goals for this class are understanding healthcare data, understanding different analytics algorithms, and understanding big data systems. What will I be able to do once I complete this course? You'll be able build models on healthcare data. For example, models for individual disease risk prediction, recommending treatments, cluster a patient into groups with common characteristics, and find similar patients. How will I be assessed? You do have few homework assignments, using big data tools. You do a project involving building a system to analyze healthcare data, writing a report, and giving a presentation.

3 Health care industry is huge, and there are a lot of data coming out of health care. U.S. health care is incredibly expensive. Overall spending is $3.8 trillion in U.S. per year. That's more than the value of ten biggest companies plus ten Beijing Olympics plus a Warren Buffet, and a Bill Gates. But does it have to be that way? There is massive waste in health care unfortunately. The estimated waste in U.S. healthcare alone is $765 billion. That's equivalent to the NASA's total budget for the past 50 years. Not only the cost is an issue, but also the quality of healthcare is poor. There are 200 to 400 thousand preventable deaths per year in the U.S. That's 1,000 people per day. Four preventable deaths occurred during this video. If we classify the preventable deaths against other causes of deaths, it will be the number three causes of deaths in the United States. After only heart disease and cancer. So, there are massive problems presented in modern health care. Including high costs, high waste, and low quality. How can big data help? The hope is big data can lead to better care and lower cost.

4 And big data for healthcare. People talk about the four v's. Volume. There's a massive amount of data, which gives analytics algorithms or systems a lot to act on. Variety. There's a variety of data that lets us connect lots of information sources together. Velocity, often times, this data, are coming in, in real time. Meaning that, data is coming in live, and needs to be processed, and analyzed, live. Veracity, there's a common problem with veracity. There's a lot of noise, a lot of missing data, a lot of errors, and a lot of false alarms.

5 Healthcare generates a large volume of date. For example, for genomic data each human genome requires 200 gigabytes of raw data or 125 megabytes, if we store just snipes. For medical imaging data, a single fMRI is about 300 gigabytes. Medical imaging data generated in the US, per year was estimated to be 100 petabyte. That's a lot of data. Healthcare also generates a lot of different kind of information. Such as clinical information, including patient's demographics, diagnosis, procedure, medication, lab results, and the clinical notes. And patient generated data such as information coming out of arm body

sensors and other devices that patients wear. And real time data sources such as blood pressure measures, temperature, heart rate, drug dispensing levels at intensive care units. Our main focus is on dealing with this wide variety of data. So, we'll talk a lot about this later in this course.

6 So that's why you should care about the health care side. What about the data science side? There's an article in Hartford Business Reviews named Data Scientist, The Sexiest Job in the 21st Century. Here are some main points in that article. Capitalizing on big data depends on hiring scarce data scientists. Scientists. What data scientists do is to make discoveries while swimming in big data. I already told you I enjoy swimming by the way. Data scientists realize that they face technical limitations, but they won't allow that to bog down their search for novel solutions. Data scientists actually created the systems that we will discuss in this course, like Hadoop and Spark.

7 Now it's time for a quiz. First, true, or false? Getting a graduate level degree is necessary for becoming a data scientist.

8 The answer is true.

9 Next, guess how much does a data scientist make, on average.

10 The answer is $120,000 per year. An experienced data scientist actually make over $150,000 year.

11 And finally, what skills do data scientists need to know?

12 The answer is: math and statistics, domain knowledge and skills, programming and databases, and communication and visualization.

02 Big Data Course Overview

13 In the first lesson, we talked about big data analytics for healthcare as a whole and what we expect from this course overall. Now, let's break down the actual topics we will cover. We'll start by talking about some of the healthcare applications for big data. Then we'll talk about some of the algorithms we will use in those applications. And then we'll talk about some of the software systems that we will use to implement those algorithms and to support those applications. If you look at the calendar, you'll see that throughout the semester, we alternate among those general topics. So, let's start by talking about healthcare applications.

14 To understand this course in general let's look at the big picture. So, this course we'll talk about three different things. We'll talk about big data systems. We also will introduce scalable mission learning algorithms. Then we'll talk about healthcare applications. And see how we can use those mission learning algorithms and big data system together to solve healthcare problems. So, let's start by talking about healthcare applications.

15 We'll talk about three types of healthcare applications in this course. Predictive modeling is about using historical data to view the model for predicting future outcome. Computational phenotyping is about turning messy electronic health records into meaningful clinical concepts. And patient similarity, it uses health data to identify groups of patients sharing similar characteristics. We'll begin with predictive modeling.

16 Predictive modeling is about using historical data to view the model for predicting future events. For example, we want to predict which treatment is likely to work for an epilepsy patient. Why do you we want to do predictive modelling? Let's motivate predictive modeling with. let's try to estimate which percentage patient was epilepsy in US responded to treatment Group A within first two years of treatment, Group B between two to five years of treatment, and Group C continued to suffer even after five years of treatment. So, write your number in those boxes, and they should probably add up to a hundred.

17 Here's the answer, group A, within the first two years of treatment, only 32% of patient in that group. Group B, between two and five years, there are 24% of patient, and group C continue to suffer after five years, there are 44% of patient. So clearly this is a problem, we like group A to be the majority. Predictive modeling would help in proof matching patients to the right treatment quickly. So that late responders in group B will be come early responder in group A. Also, this will help identify non-responder in group C quickly, so that new treatment can be developed for them.

18 So what makes predictive modeling difficult? We have millions of patients we want to analyze and their diagnosis information, medication information, and so on. So, all this data combined together, create a big challenge. The second challenge in predictive modeling is there's so many models to be built. Predictive modeling is not a single algorithm, it's a sequence of computational tasks. We'll introduce predictive modeling pipeline in more details in a later lecture. But every steps in this pipeline has many different options. All of those combined give us many, many pipelines to be evaluated and compared.

19 We just talked about predictive modeling. Next, let's talk about computational phenotyping. The input to computational phenotyping is the raw patient data. It consists of many different sources such as demographic information, diagnosis, medication, procedure, lab test, and clinical notes. And phenotyping is the process of turning the raw patient data into medical concepts or phenotypes.

20 To help us understand phenotyping better, let's do a quiz. Imagine you're trying to extract phenotypes from this raw data, so what are the waste products we should deal with? For example, missing data could be one. So, write down some of those wast product in this box.

21 Okay, here are some possible answers. Missing value, some important data may be missing from the raw data. We have to deal with that. Duplicates, some patient record may show up multiple times due to recording errors. Irrelevant data, not all the raw information are relevant for a specific task. We want to get rid of those irrelevant information. Redundant information, different data records can indicate the same underlying problems. For example, both diagnosis and medication records from a patient indicates underlying condition of Type 2 diabetes. So, we want to consolidate those redundant information.

22 Next let's see an example of phenotyping algorithm for type 2 diabetes. So, the input to the algorithm is EHR, Electronic Health Record of a patient. Then we'll first check whether the patient record indicate type 1 diabetes diagnosis. If the answer is no, then we continue checking with a Type 2 Diabetes diagnosis is present. If again no, would check whether Type 2 medication is given. Then if the answer is yes, we check whether any abnormal lapse is present. If yes, then we confirm this patient record, indicate this patient has Type 2 Diabetes. And this is not the only way to identifying Type Two Diabetes cases. There's a different path. For example, from here we can check Type 2 Diabetes diagnosis, if the answer is yes, we'll check medication for Type 1 Diabetes. If the answer is no, we'll check medication for Type 2 Diabetes. And if this answer is no, then we go back to check for abnormal labs. If the answer is yes, again this record indicates Type 2 Diabetes. If at this stage the Type 2 Diabetes medication is confirmed, then immediately we know this record indicates Type 2 Diabetes patient. And this is still not the complete algorithm. There's two other paths can lead to type 2 diabetes. Does the entire flow chart give us one example of phenotyping algorithm for type 2 diabetes? So, you may wonder why do we need such a complicated algorithm to determine whether patient have Type 2 Diabetes. Can we just ask whether patient have Type 2 Diabetes diagnoses present in the data? Shouldn't that be enough? The answer is no. The reason is because electronic house record data's very unreliable. There are missing data, redundant information, so sometimes for Type 2 Diabetes patient, the diagnosis is not present in the record. So, we still have other way to check whether our Type 2 Diabetes patient, for example, their medication, lab tests. So that's why it's not sufficient just checking one source of information. At the same time, even the Type 2 Diabetes diagnosis is present, it's not necessarily confirmed the patient has Type 2 Diabetes, because patient can come to the clinics for a checkup, for screening purpose, then this diagnosis code can still be present in the data. So, we have to check additional things, such as medication, and lab tests to really confirm this patient has Type 2 Diabetes. In this class, we'll learn how to develop phenotyping algorithm look like this from data, and also how to implement such algorithm efficiently using big data systems.

23 Okay, so we talked about predictive modeling application phenotyping. Next, we'll introduce patient similarity. To motivate patient similarity let's do another quiz. So, which of the following type of reason do doctors engage most often during patient encounters. Is it based on flowchart reasoning like what we have seen it phenotyping algorithm? Or is it based on her instinct and intuition? Or is it comparison to past individual patients?

24 So the correct answer is comparison to the past individual patients or case-based reasoning. Based on our anecdotal experiences, doctor often compared the current patient to the old patient they have seen.

25 So patient similarity is about simulating the doctor's case based with the computer algorithms. Instead of depending on one doctor's memory, wouldn't it be nice if we can leverage all the patient data in the entire database? So, the idea is when the patient comes in, the doctor does some examination on the patient. Then, based on that information, we can do a similarity search through the database. Find those potentially similar patients, then doctor can provide some supervision on that result to find those truly similar patients to the specific clinical context. Then we can group those patients, based on what treatment they are taking, and look at what outcome they're getting. Then recommend the treatment with the best outcome to the current patient. And that's what patient similarity does. So, in this course, we'll learn about different patient similarity algorithms and how to implement that in an efficient manner using big data systems

26 So far, we talked about health care applications. Next, we introduce what machine learning algorithms will be covered in this course. So, in this course, we'll cover big data algorithms. We'll talk about classification algorithms, clustering algorithms, dimensionality reduction algorithms and graph analysis. First, let's talk about classification algorithm. So given a matrix X, here every row represent a patient, every column represent a disease and every element here indicate whether a specific patient has a specific disease. Then, we learn a function f that map each patient to a target variable y. For example, here, the target could be whether a patient had a heart attack or not in next six months. And we'll also talk about clustering algorithms. So here the input of clustering algorithm is similar to classification. So, we have a matrix x, every row represent a patient, every column represent a disease, and we want to learn a function f that partition the set of patient into different clusters. For example, x1, x2, x3 represent different patient clusters. And the patient within a cluster are similar to each other and they're different from patient in different clusters. We'll also talk about dimensionality reduction algorithm. Here the input is a large matrix of the set of patients with large number of features. And the output of dimensionality reduction is a smaller matrix, x prime, that consists of the same set of patients with smaller number of features. There are different ways to construct those features. Sometimes those features are good summary of all the feature in the original matrix. Sometimes those are the only features we care about in order to predict a specific target. We also learned graph analysis. For example, we have two patient here, and we connect those patients to a set of diseases they have and also connect the diseases are related to each other. Given this network of patients and diseases, and we want to learn what are the important patients or disease in this network and also how they related to each other?

27 So far we're talking about healthcare application and machine learning algorithms. Next we'll talk about big data systems. In order to deal with big data set and implement your algorithm to process big data set, we need big data systems. So, in this course we'll introduce two popular big data systems. Hadoop and Spark. Hadoop is a distributed disk-based big data systems that all the data are stored in disks. Well, Spark is a distributed in-memory big data systems. That most data store in memory. So, Spark in general is much faster than Hadoop, but both are popular big data system that people are using. In this course, we'll talk about Hadoop and all the important building blocks in Hadoop. We'll talk about the core infrastructure of Hadoop, the MapReduce programming model and HDFS storage systems, and the high-level processing systems, such as Pig, Hive, and HBase. So, in this course, we will also talk about Spark, the core infrastructure of Spark, how do we store data and how do we process data. Using Spark

and the high-level abstractions such as Spark SQL and Spark streaming, MLLib for large scale machine learning library using Spark, and GraphX for processing graph data using Spark.

28 So today we talked about three big parts of this course. We talked about the healthcare applications, the machine learning algorithms, and the big data systems. We integrate all of this throughout the course. We'll move back and forth between application, algorithm, and systems. For example, we might build a scalable classifier using logistic regression on Hadoop for predicting heart failure. So, let's get started.

03 Predictive Modeling

29 This lesson is about predictive modeling. What is predictive modeling? Not quite. Predictive modeling is a process of modeling historical data for predicting future events. For example, we want to use electronic health record that we have available to view a model of heart failure. So that we can predict patients who are at risk of developing heart failure sooner. The key goal we want to answer in this lessons is how do we develop a good predictive model using electronic health record quickly?

30 In this lesson we'll focus on describing how to perform predictive modeling using electronic health records, or EHR. To demonstrate the importance of predictive modeling and EHR, here we're showing the number of publications with the keyword predictive model over times and the number of publications with the keyword EHR over time. Especially in the past few years, there is an explosion of interest in EHR as EHR become a major data sources for clinical predictive modeling research. Therefore, it's important to learn how to develop a good predictive model using EHR data.

31 Predictive modeling is not a single algorithm, but a computational pipeline that involves multiple steps. First, we decide the prediction target, for example, whether a patient will develop heart failure in the next few years. Second, we construct the cohort of relevant patients for the study. Third, we define all the potentially relevant features for the study. Fourth, we select which features are actually relevant for predicting the target. Fifth, we compute the predictive model, and sixth, we evaluate the predictive model. Then we iterate this process several times until we are satisfied with the resulting model. Now let's start with prediction target.

32 There are often many targets that an investigator want to predict using the data they have. However, only a subset of them are possible. So, we should choose the prediction target that addresses the primary question that is both interesting to the investigator and possible to be answered using the data. For this lessons, let's focus on predicting the onset of heart failure, which is an interesting and potentially possible target.

33 Here's a quiz question on heart failure. Make a guess. How many new cases of heart failure patients occurred each year in the U.S.? Is it a 17,000 patients? Or b, 260,000 patients? Or c, 550,000 patients? Or d, 1,250,000 patients?

34 The correct answer is 550 thousand patients, which is a huge health care problem.

35 So we want to develop a predictive model for heart failure. But what are the motivations for early detection of heart failure? First, heart failure is a complex disease. There is no widely accepted characterization and definition of heart failure. Probably because the complexity of the syndrome. It has many potential ideologies, diverse clinical features, and numerous clinical subsets. If we can detect heart failure earlier, we can potentially reduce the cost of hospitalization associated with heart failure. We can also potentially introduce new early intervention to try to slow down the progression of heart failure, improve the quality of life, and reduce mortality. In the long term we can improve existing clinical guidelines for heart failure prevention. So, in this class we'll show you how to develop a predictive model for predicting heart failure earlier.

36 So far, we're talked about how to define the prediction target. Next, we introduce the Cohort Construction step. Cohort construction is about defining the study population. For a given prediction target, there are only a subset of patient that are relevant among the whole patient-population. And they

are the Study Population. Be aware, often may not be possible to obtain data from everybody in the study population. As a result, the data set we studied is only a subset of those Study Population. So, the question is, how do we define the Study Population? There are two different axes to be considered. One the vertical axis, we have prospective study versus retrospective study. On the horizontal axis, we have cohort study verses case-control study. Depending on the combinations, we have four different options. Perspective Cohort study, Perspective Case-Control study, Retrospective Cohort study, and Retrospective Case-Control study. Now let's look at this two axis in more details.

37 Now, let's talk about prospective versus retrospective studies. In a prospective study, we first identify the cohort of patients, then decide what information to collect and how to collect them. Then start the data collection. In contrast, in a retrospective study, we first identify the patient cohort from existing data. For example, past electronic health records of patients, then retrieve all the data about the cohort. So, in prospective study, we identify the cohort and collect the data from scratch. But in the retrospective study, the data set already exists. We just need to identify the right subset and retrieve them.

38 Here's a quiz question to compare perspective and retrospective studies. Each row represents a particular property. Pick the study that has the corresponding property. More specifically, which one has more noise in their data? Which one is more expensive to conduct? Which one takes a longer time to conduct, and which one is more commonly done on larger dataset?

39 Here are the answers. So, retrospective study often work on data with more noise, because data are often created for other purpose, not research. In contrast, prospective study, because you design a dataset collection process specifically for this research, the quality of the data is often higher. As a result, less noise. Prospective study is often more expensive and takes longer time to conduct because the data has to be collected from scratch. Finally, because the cost and time constraints, the size of the data set prospective study used is often smaller and limited. On the other hand, retrospective study deal with historical data. It often can work with much larger data set.

40 Next let's talk about COHORT study. In a COHORT study, the goal is to select a group of patients who are exposed to a particular risk, for example, if we want to be in a predictive model for predicting heart failure readmission. Here heart failure readmission means, heart failure patient after discharged from the hospital, comes back again to the hospital due to heart failure. In this case, the COHORT should contain all the heart failure patients who discharged from the hospital, because they can potentially be readmitted after discharge. The key in COHORT study is to define the right inclusion and exclusion criteria to figure out what patient to include. Here's a visual illustration. We start with all patients, then we try to identify the relevant patient for a particular risk, for example, these three patients are relevant for this risk, such as heart failure readmission. Then we want to build a model to predict the target risk. In this case, the COHORT contains both positive and negative examples, for example, patient with heart failure readmission and patient without heart failure readmission. All this relevant patient is a COHORT in this study.

41 The other common study design is case-control study. In this design, we try to identify two sets of patient, namely cases and controls. And we put them together to construct the cohort. Cases are patients with positive outcome. For example, the patient who develop the disease. Controls are the patients with negative outcomes. That is, they're healthy patients, but otherwise similar to the cases. For example, they can have the same age, gender, and visit the same clinics. And the key here is to develop

the matching criteria between cases and controls. For example, we want you to predict a model of heart failure. Then we identify a study population of over 50,000 patients, and we have 4,644 case patients. Those are the patients who developed heart failure. And we matched them against a set of control patients on age, gender, and clinics. And we end up with 45,000 control patients. Notice that in this study, we have a lot fewer cases with heart failures than the controls without heart failures. This is pretty typical because in a real-world scenario, patients with the specific disease conditions are often harder to obtain, while there are a lot more patients without that disease are available to serve as a control. To summarize, in a case-control study, we first identify the cases, then try to match them to a set of control patients. In a cohort study, we'll identify all the patients who are exposed to the risk and the matching criterias are not involved.

42 So far we talked about how to define the prediction target, how to construct the patient cohort, next we introduce feature construction step. The goal of feature construction is to construct all potentially relevant features about patients in order to predict the target outcome. Next, we introduce a few key concepts that are related to feature construction. First, the raw patient data arriving as event sequences over time. Diagnosis date is the date that the target outcome happened. In the heart failure predictive modeling example, each patient is diagnosed with heart failure on this date. Since control patient does not have heart failure diagnosis, in theory, we can use any days from control patient as the diagnosis date. But commonly we choose to use the heart failure diagnosis date of the matching case patient as diagnosis date for the corresponding control. Before the diagnosis day, we have a time window, called prediction window. Before the predication window, we have the index day at which we want to use the learn predicted model to make a prediction about the target outcome. Before the index day, we have another time window called observation window. We use all the patient information happening during this observation window to construct features. There are many different ways to construct features. For instance, we can count the number of times an event happens. For example, if type two diabetes code happened three times during this observation window, the corresponding feature for type two diabetes equals three. Or sometimes we can take average of the even value. For example, if patient has two HBA1C measures during observation window, we can take the average of this two measurement as a feature for HBA1C. The length of prediction window and observation window are two important parameters that going to impact the model performance. Next, we illustrate their impact using some examples.

43 Here's Chris to help us understand the impact of prediction window and observation window. Which of the following timelines is easiest for modeling? Is it A, large observation window and small prediction window? Or B, small observation window and large prediction window? Or C, small observation window and small prediction window. Or D, large observation window and large prediction window.

44 The answer is A, large observation window and small prediction window. Because it is often easier to predict event in the near future, that is, small prediction window. On the other hand, large observation window means more information to be used to construct features, which is often better since we can model patient better with more data. Therefore, large observation window and small prediction window is easiest for modeling.

45 Here is another quiz. Which of these timeline is the most useful model? Large observation window, small prediction window. Or small observation window, large prediction window. Or small observation window and small prediction window. Or large observation window and large prediction window.

46 The answer is B. Small observation window, large prediction window. In this idea situation, if we can construct a good model, we want to predict far into future. Therefore, large prediction window, without much data about the patient. Therefore, small observation window. That's why B reflects idealistic timeline. If we compute a model in this setting, this will be the most useful model. However, the setting is often difficult to model, therefore unrealistic.

47 Here's another example illustrating the impact of prediction window. In this chart, the y axis is the accuracy of the model, the higher the better. The x axis is the size of the prediction window, which varies from zero days to 900 days. We can see the accuracy of the model drops as we increase the prediction window. Because it's easier to predict things happen in near future, than things happen far into the future.

48 Here's another quiz question on prediction window. Which of the following options is the most desirable prediction curve? Is it A, or B, or C, or D?

49 The answer is B because we can predict accurately for fairly long periods of times up to 450 days of prediction window. While the performance of all the other model jobs fairly quickly as the prediction window increases. You may notice that A has the maximum accuracy at the beginning. However, as the prediction window increases, the performance of A drops quite quickly. So, it's not that of useful model for predicting long term things.

50 Now let's consider the performance of different observation windows. Typically, as the observation window increases, the performance improves because you know more about the patient as the observation window increases.

51 Here's the quiz on observation window. Given the performance curve when we vary the observation window like this. What is the optimal observation window we should choose? Is it A, 90 days, or 270 days, or C, 630 days, or D, 900 days?

52 The answer is C, because the model performance plateaued after 630 days. It indicates a diminishing return as we go further beyond that point. Therefore 630 days is a good choice. So, you may wonder, choosing 900 days may also be a good choice, but it's a trade-off between how long is the observation window, and how many patients have that much data. So, if you chose 900 days as observation window for patients who do not have enough data up to 900 days, they will be excluded from the study.

53 We covered the first three steps. Next we'll talk about the feature selection step. In the feature selection step, we have talked about how construct features using patient event sequences from the HRC data. In particular, we construct feature from raw data in the observation window. If we look closely at the observation window, we see event sequence data, which are corresponding to different types of clinical events. For example, diagnosis, symptoms, medications, patient demographics, lab results, and vital signs. We can construct features from all those events. However, not all the events are relevant for predicting a specific target. The goal of feature selection is to find the truly predictive features to be included in the model. For example, here are two patient charts. We can see some features such as

demographics, including age, race, and gender, and vital signs such as blood pressures, and diagnosis such as diabetes and hypertension. However, in reality this patient chart is not that simple. In fact, we can construct a long list of features over 20,000 features, from a typical EHR data set. Not all of this are relevant for predicting a target. We need to select the ones that are relevant to the target condition. For example, if we want to predict heart failure, maybe those yellow features are relevant. However, for a different condition, such as, diabetes. Maybe those purple features are relevant. The goal of features selections is to identify those predictive features. Giving a specific target condition.

54 So far we've figured out what we want to predict and who we will use to make the prediction and what feature is to be used in this prediction. And next let's see how we make the prediction. Predictive model is the function that maps the input features of the patient to the output target. For example, if we know a patient's past diagnosis, medication, and lab result, if we also know this function, then we can assess how likely the patient will have heart failure. Depending on the value of the target, the model can be either regression problems or a classification problem. In regression problem, the target is continuous. For example, if we want to predict the cost that a patient will incur to the healthcare systems, then it's a regression problem, and y is the cost in dollars. And the popular method includes linear regression and generalized additive model. And if the target is categorical, for example, whether the patient has heart failure or not, then it's a classification problem. Popular method include logistic regression, support vector machine, decision tree, and random forest. You may have learned all those methods in other courses such as machine learning. And in this course, we'll utilize all those methods again in the context of building a predictive model for solving healthcare problems.

55 The final step of this pipeline is to assess how good our model is through performance evaluation. Evaluation of predictive models is one of the most crucial steps in the pipeline. The basic idea is to develop the model using some training samples but test this train model on some other unseen samples, ideally from future data. It is important to note that the training error is not very useful, because you can very easily over fit the training data by using complex models which do not generalize well to future samples. Testing error is the key metric because it's a better approximation of the true performance of the model on future samples. The classical approach for evaluation is through cross-validation process or CV.

56 Now, let's talk about cross-validation. The main idea behind cross-validation is to iteratively split a data set into training and validation sets. And we want to view the model on the training set, and test the model on the validation step, but do this iteratively, many times. Finally, the performance matrix are aggregated across this iterations often by taking the average. There are three common messes for cross-validations namely Leave-1-out cross-validation, k-fold cross-validation, and randomized cross-validation. In Leave-1-Out cross validation, we take one example at time as our validation set and use the remaining set as the training set. Then repeat this process many times, goes through the entire data set. The final performance is computed by averaging the prediction performance across all iterations. K-Fold's cross variation is very similar to leave-1-out validation. But instead of just using one example of validation set, we have multiple examples in the validation set. More specifically, we split the entire data set into K-Folds. And we iteratively choose each fold as set, validation set and use the remaining Folds as a trimming set. For example, the Fold 1 would be used as the validation set, and the remaining fold will be used as a trimming set to view the model. Then we use a Fold 2 as the validation set, the remaining fold as the training set to be build another model. And repeat this process K times, and the final performance is the average over this four different models. Finally, randomized cross validation will randomly split the data

set into training and testing. For each such split, the model is fit to the training data set, and the prediction accuracy is assessed using the validation set. The result are then averaged over all the splits. The advantage of this method over the K-fold cross validation is that the proportion of the training and validation set is not depends on the number of fold. The disadvantage of this method is that some observation may never be selected Into the validation set because there's randomization process. Whereas some other samples may be selected more than ones into the validation set. In other words, validation set may overlap.

57 To conclude, in this lesson, we introduced the key steps in building a predictive model. Which include define what is the prediction target and construct the right patient cohort, then construct all the possible relevant features from data, then find which features are relevant, and view the predictive model, and finally, evaluate the model performance. Now you should be able to design a high-level predictive modeling study using this pipeline on the HR data.

04 MapReduce

58 In the past couple of lessons, we have been talking about predictive modeling on big data. Today we're going to talk about a tool for processing big data called MapReduce. MapReduce is a powerful system that can perform some of the methods we have talked about so far. On big data set using distributed computation and distributed storage. We'll start by discussing what MapReduce is. Then we'll discuss how MapReduce takes care of fault tolerance ina distributed environment. Finally, we'll talk about some of the analytics that can be performed when snap produce and the limitations it carries.

59 Now let's talk about Hadoop and MapReduce. So, what is Hadoop or MapReduce? Is it a programming model for developer to specify parallel competition algorithms? Yup. We talk about MapReduce paradigm. Is it an execution environment? Hadoop is the Java implementation of MapReduce and Hadoop distributed file system. So, it is an execution environment. Is it a software package? Sure is! In fact, there are many software tools have been developed to facilitate development effort for data science tasks, such as data processing, extraction, transform and loading process, statistic computation, and analytic modeling using Hadoop. In summary, Hadoop and MapReduce enables a powerful big data ecosystem by providing the combination of all these things. So, in fact, MapReduce or Hadoop is a big data system that provides the following capability, distributed storage for large data set through Hadoop distributed file system, distributed computation through programming interface MapReduce, and fault tolerance systems in order to cope with constant system failures on large distributed systems that are built on top of commodity hardware.

60 Originally, MapReduce was proposed by Jeff Dean and Sanjay Ghemawat from Google in 2004 and were implemented inside Google as a proprietor software for supporting many of their search engine activities. Then later on, Apache Hadoop is developed as an open-source software that mimic original Google's MapReduce systems. It was written in Java for distributing storage and distribute processing of very large dataset to program. On Hadoop systems, we have to use the programming abstraction MapReduce, which provides a very limited, but powerful programming paradigm for parallel processing on a large dataset. All the algorithm running on MapReduce or Hadoop have to be specified as MapReduce programs. The reason for this very constrained programming model is to support super scalable, parallel and fault tolerance implementation of data processing algorithm that can run on large dataset. To utilize Hadoop and MapReduce for data analytics, we have to understand and master common patterns for computation using MapReduce. We explain this next.

61 There's a fundamental pattern for writing a data mining or mission learning algorithm using Hadoop is to specify machine learning algorithms as computing aggregation statistics. Say we want to implement in a machine learning algorithm for identifying the most common risk factors of heart failure. We need to decompose the algorithm into a set of smaller computation units. In particular, we need to specify a map function f, where f will be applied to all the heart failure patients in the large database. For example, we want to extract the list of risk factors related to heart failure appear in each patient's record. Then result from this map function will be aggregated by a reduce function. For example, instead of listing the risk factors for each patient, we want to compute the frequency of each risk factor over the entire population. Then in this case, the reduce function would do that by performing the aggregation statistic on the result from the map function. So, this process is quite abstract. Next we'll go into more details to explain why such abstraction is required and what are the benefit and limitation of this abstraction.

62 Here's an example to illustrate MapReduce in more details. Say we have a large database of patients stored in the Hadoop distributed file system. Each patient is stored as a separate record, and each record consists of the history of this patient encounter. For example, the diagnosis, medication, procedure, and clinical notes are all stored in those patient records. Our goal is to write up MapReduce programs to compute the number of cases in each disease. To do this in MapReduce, we first specify the map function that goes through each patient records, and extracting the disease mentions and output that. For example, we have two patient record over here, and when we apply the map function, we'll first find the disease mentioned in this record. For example, for the first record, we identify hypertension and diabetes are in the record. Then, for the list of disease mentioned, we identified inside this record, we emit the disease name and the value 1. And the output from each map function looks like this. For example, for this patient we have a hypertension and value 1, and diabetes with a value 1. For the second patient, he has heart disease with value 1 and a hypertension with value 1. And the disease name in this case is the key, and the value 1 is the value. So, map function would process input record, and output a set of key value pairs. You notice that the same key value pairs may appear as output from different map function. For example, hypertension. Value 1 happened in this output and also in this output. So, the next phase will combine them. All the output from map function will be processed internally by Hadoop. In particular, all those output will be shuffled and aggregated. For example, hypertension happened twice over here, and after the shuffling and combination phase, we have hypertension and all those associate value. Diabetes, on the other hand, only happened once in this record, so the corresponding lists of value only have one value, and similarly for heart failure, we only have one value here. And this intermediate result will be the input for the reduce function. To write a map reduce program we need to specify the map function, and the reduce function. So, here's one example for a reduce function. The reduce function will take the disease, key, and a list of disease values. For example, we have hypertension as the disease key, and we have a values 1 and 1. So in this reduce function, we'll take the disease and the list of value, and we sum up those value. So, the result of the reduce function would give us hypertension value 2, diabetes value 1, and heart disease value 1. So, in this very simple example, you may feel like this two-stage process is very strange. But if we're dealing with large data set, with billions of records, this two-faced process is very important. And next, we'll explain how internally macro view system works. And you understand why we have to specify the competition in these two phases.

63 So far, we illustrate the high-level ideas on how to write a MapReduce program. But how does MapReduce work? And why it requires such a strange two-phased process? We have to realize; the real word data set is often too big to be stored and processed on a single machine. So, we have to split the data into partitions so that each partition can be stored and processed in parallel on multiple machines. So, MapReduce systems has two components, mappers, and reducers. So, all those data will be partitioned and processed by multiple mappers. And each mapper deal with a partition, which is a subset of records in the entire dataset. For example, mapper 1 processes three records, mapper 2 processes another three records and mapper 3 process the remaining records. Then, we have reducers, in this case, we have reducer 1 and reducer 2. So, they are also divided the work by processing intermediate result, in certain ranges. For example, reducer 1 will be in charge of heart disease, and reducer 2 will be in charge of cancers. So, in the map phase, each mapper will iteratively apply the map function on the data that they're in charge of. And also, they will prepare the output that will be sent to the corresponding reducer. For example, we have a set of intermediate result prepared for reducer one. And we have a set of intermediate result prepared to be sent to reducer 2. And across different mappers, this process are

happening in parallel. For example, mapper 1 is processing the first record by applying the map function. So, identify two diseases mentioned. One Is to be sent to reducer 1, is about heart disease and second disease mentioned is about cancer. It will be sent to the reducer 2 and the same process is happening concurrently on different mappers. For mapper 2, the first record would emit one disease mentioned for reducer 2. And for mapper 3, we admit one record for reducer 1. And this process is iteratively going through all the records. And all those intermediate results will be processed internally through a combination and shuffling process. So, we'll combine the values for the same disease at each mapper, then stand out all those intermediate results to the corresponding reducers. So, at the reducer side, once they receive those records, they can start generating the final output by applying the reduce function. For example, this R2 hypertension mentions, and we obtain the final com, that's 3. And we're going through the other disease and compute the final com. Then output the result, so these are the final output from two different reducers. For example, for hypertension, it happened 3 times. For another heart disease, it happened 2 times, for these two types of cancers, they both happen 3 times. So, to summarize, this MapReduce system has three different phases. The map stage, where we performed map function and the pre-aggregation and combination functions. Then we'll have a shuffle stage, all of the intermediate results will be sent to the corresponding reducers. And we have the final reduce stage, where the final output are generated.

64 So one of the key functionality that MapReduce or Hadoop system provide is fault tolerance. On a large cluster environment, many things can fail at any given times. So, failure recovery is very costly, and often times, data scientists do not want to deal with failure recovery when they implement their algorithms. They want to assume the system will always work. It's really on the shorter of the system to take care of all the potential failures that can happen in this distributed environment. For example, this mapper two can fail during execution of map reduce program. And this time, the map reduce system will restart mapper two then goes through the same workload again, by generating the intermediate result. Then all the intermediate results will be sent to the corresponding reducers and the final output will be generated. Similarly, reducer can also fail. For example, if reducer two failed, then Map Reduce system will automatically restart reducer two and extract the corresponding intermediate result again back from the mappers and re compute all those reduce functions. But keep in mind, the system of Map Reduce is designed in a way that they want to minimize the re-computation. Ideally, when a component failed, only that component should be re-computed. For example, when reducer two failed, reducer one should not be impacted at all. So, Map Reduce systems is designed in a way that such optimization is taken into place, and all those re-computation, is minimized.

65 The MapReduce system is just part of the Hadoop systems. There is another part that's very crucial as well, that is the distributed file systems. Here, we're going to illustrate the ideas behind the Hadoop Distributed File System or HDFS. Given a large file, it's impossible and impractical to store the whole file on a single machine. Oftentimes, we split this large file into different partition, for example, here we have four partition, A, B, C, D. Then we store those different partition on different machines where we call workers. Here we have four workers, and the other thing we do is, we store multiple copy of the same partition on different workers. For example, for partition A, we store it on three workers, on 2, 3, and 4. Same for B, C, and D. So, this way, we distribute a large file across multiple machines. In order to access these large files, we can retrieve those different partitions many different ways. There are two benefits for doing this. One is now we can access the multiple partitions in the single file concurrently. So, the root speed is actually faster than accessing the single file on a single machine. But more

importantly, if any of those workers failed, we still can't retrieve a large file. Say we have two worker failed, worker 1 and worker 3. We can still access the entire file by retrieving all of those partitions from worker 2 and 4. So HDFS is the backhand file system to store all the data you want to process using MapReduce program.

66 So the key design philosophy behind MapReduce is not to include many functionalities that people may want. The key design decision is, what functionality can we remove so the system is more reliable and more scalable? At the same time, we want to make sure even with the minimum functionality that provided by MapReduce, so we can still enable powerful computational algorithms such as machine learning. So given the machine learning algorithms, there's a lot of limitation to utilize MapReduce. For example, we can't really direct access data. Instead, we have to specify map and reduce function and compute in a very restricted forms this aggregation query. So, all those map function are those function applied to the individual data records. And then this aggregation query in this example, which is taking the sum and averaging that, is the reduce function. So, this seems to be a very restricted computation paradigm, but surprisingly, many of the machine learning algorithms can still be supported with MapReduce. On the other hand, Hadoop systems will provide distributed file systems and distributed computation. And more importantly they will ensure fault tolerance and straggler mitigation. So here, straggler mitigation is really an extension of fault tolerance. For example, if we have one mapper that runs very slowly, we'll correctively start the same mapper on a different machine. Then leave both mapper runs, and whichever finish first will take the result and stop the other one.

67 Next we'll talk about some analytic example using MapReduce. So, in particular, we'll cover a k-nearest neighbor classifier and linear regression.

68 Now let's talk about how to use MapReduce to write a K nearest neighbor classifier. Given a set of patients, say we want to predict whether a given patient will have a heart failure or not, by finding a similar patient to this query patient. So here we have a set of patients, and we plot them in this two-dimensional space. X axis is the cholesterol level, and y axis the blood pressure. And every black point over here indicate a patient, and the value on those points are whether they had heart failure or not. For example, a patient over here as heart failure, and a patient over here does not. And the goal is, given a query patient, for example, this patient over here. We want to find the nearest neighbors, then do the majority vote and to predict whether this patient will have heart failure or not based on those similar patients. In the map reduce setting, all those patients will be split into multiple partition. In this case we have a two different partition and this green partition for mapper one is brown partition for mapper two. And this red point over here is our query data point. And we want to find three nearest neighbor. Now we need to specify the map and reduce functions. So, the input to the map function are all those data points and this particular query point. To output are K nearest neighbors for each partition, and the algorithm is quite simple. For each partition we'll go through all those data points, and you need the K closest point. For example, in these three points over here are the local nearest neighbor in partition one and this three points over here are the local nearest neighbor in partition two. And those will be the intermediate results sending to the reduced phase. So, in the reduced phase we need the local nearest neighbors and the query point, and the output is a final global nearest neighbor, and the algorithm is almost the same as the map function, so reduced function will go through all those local nearest neighbor to identify the global nearest neighbor, which are the three points over here.

69 Now let's talk about how to use to implement linear regression. For example, we want to view the linear regression that map patients information to heart disease risk. And we want to find out the coefficients associated with all those patient features. So, in this particular case, we have input feature matrix which is n by d and n is number of patients, d is number of features and the output target variable which is n by 1. Every row here is the heart disease risk associated with that individual. Finally, we have a d by 1 vector and every element here tells us about the coefficients in their linear regression model for that corresponding features. For example, the first features may be the coefficient for age, the second feature may be the coefficient for height, and so on. So now let's see how we compute such a model using MapReduce. So, from statistic class, we know that there's many way to solve linear regression, and one of the popular way to do that is using this normal equation. That is, the optimal coefficient in maximum likely cosines can be computed by taking X transpose X inverse times X transpose Y. If we have a small data set, this can be easily computed on a single machine using your favorite statistical tools, such as R or MadLab. However, if we have a very large data set with billions of patients, this computation cannot be done on a single machine. And next we'll see how we can use MapReduce to help us to do this. To write this as a MapReduce program, we have to understand this equation a little bit better. There are two steps involved here. One is to compute X transpose X. The other part is to compute X transpose Y. And both can be further decomposed into aggregation statistics. For example, here X transpose X becomes summation from a one to n over xi times xi transpose. And, similarly, x transpose y becomes summation over i from 1 to n, xi times yi. So here, immediately, we can see the patterns we're looking for in term of aggregation statistics. This can easily be mapped to MapReduce computation. For example, this xi times xi transpose becomes the map function, and this xi times yi becomes another map function, f2. For example, to implement the first one as a MapReduce function, here are the pseudo code. For the map function, the input is those x and y pairs, and x is the patient feature and y is the heart disease risk. In this particular case, we only need the x feature vector, and we want to compute x times x transpose. In the reduce phase, we take all the output of those xi times xi transpose and compute the sum

70 Now we understand how to decompose this normal equation into two map reduce formulation. We already shown you how to specify this part, x transpose x, as map reduce computation. And now let's do a quiz. Can you specify the pseudo code for map and reduce function for computing x transpose y?

71 So here are the answers. So, in the map phase, we're taking our patient record and their heart disease risk, and we just simply compute x times y. So, in this case, this corresponding to Xi times the heart disease risk, Yi. And the reduce function is the same as before. So, we take all those Xi times yi from each patients and just compute the sum.

72 So far we've talked about MapReduce and have demonstrate some example using MapReduce for machine learning. And what are the limitation of MapReduce? Let's illustrate that through an example. One of the popular machine learning algorithm is logistic regression. So, it's the classification model, and the formulation is very similar to linear regression, but It's not very easy to implement using map reduce. Here's why. For example, we have a set of data points and x are the set of patients who are likely to have heart failure. Yellows are the set of patients who do not have heart failure. And we want to learn a classifier to separate them. And this line is optimal separation, and we want to find that. So, recall we have talked about gradient descent and stochastic gradient descent in an earlier lecture. And here, if we want to use gradient descent to find optimal classification line, how would that work? So, for simple demonstration we have a very clear separation between this two groups of patients. And this line is what we want to learn. So, we may start with a random initial points, then start computing the gradient, then

update the gradients, then iteratively it will converge to the optimal line. So that's how reading design works, and that's also how many machine learning algorithms work. They all require this iterative computation. So, if we want to map this using MapReduce paradigms, how would this look like? Next, we'll see how we can map this into a MapReduce computation. Then you will realize why it's not efficient to use MapReduce for this type of workload. In order to implement the machine learning algorithms, we have to formulate this as a set of aggregation statistic computation. So, after specified the map functions, then it will be applied to the entire data set and then we aggregate the result and then that will be the reduced function. To compute logistic regression using map reduce, we have to specify a map function which corresponding to computing the greeting for a single patient. Then this function will be applied to a large data set with millions of patient. Then, after we apply this math function over all the patients, then we aggregate them together. That give us the gradient. Then we update the parameters of the logistic regression using gradient descent and iterate. To use a MapReduce to compute logistic regression requires this iterative process. Every iteration requires loading the data two times, one for compute the map functions, one for compute the re use function. And we have to do this iteration many, many times, so it's not efficient. So, in summary, MapReduce is not optimized for iteration or this multistage computation.

73 Now let's do a quiz about MapReduce. So, which of the following would be the best for a MapReduce job? Single pass over the data versus multiple pass over the data. The key distributions are skewed versus uniformly-distributed keys. So here what I mean by that is, for example, if the age is the key, and age is concentrated only on the age range of twenty to thirty, then it's skewed distribution. While if the age is uniformly distributed from one to eighty, then it's uniform distributed keys. Third, no synchronization needed between different part of computation, versus a lot of synchronization is required.

74 The answer is MapReduce is best for single pass, such as computing histograms. But it's not good for multi pass, for example, computing iterative optimization algorithms, such as logistic regression. So, MapReduce is good for uniformly distributed keys. And also good for Skewed distribution of Keys. If we have a Skewed distribution of Keys, then only one reducer has to do all the jobs. As opposed to Uniformly-distributed across all reducers. Finally, map reduce is good for No synchronization is required. It's not so good when a lot of synchronization is required. In fact, MapReduce has very little synchronization. The only synchronization in the MapReduce drop, is between map and reduce phase. So, during map phase, everything is done in parallel independently. And everything inside the reduce phase, is done independently in parallel. And the only synchronization is between map and reduce phase.

05 Classification Methods Metrics

75 Last time, we talked about the overall process of predicted modeling. Now, we'll talk about how to evaluate the performance of predicted models. When we are doing predicted modeling, one of our fundamental concerns is the quality of the models we developed and so we need metrics to evaluate the quality of the models. And there are many evaluation metrics, such as accuracy, sensitivity, and specificity, which we'll talk about in this lesson. A big part of working with big data is developing multiple mottos and comparing them against one another. An evaluation metrics are the language we use to make these comparisons.

76 In the last lesson we talked about predictive modeling pipeline. It involved multiple computational steps. In this lesson, we'll focus on performance evaluation, and go into details, what are those different performance evaluation metrics. Predictive model is a function that map feature x to predict output target y. The most common predictive models are either classification problems or regression problem. For classifications the target variable y is either binary or categorical. There are many performance metrics associated with classification problems such as, true positive rate, false positive rate, positive predictive values, F1 score, area under the ROC curve, and many, many more. On the other hand, if the target variable Y is continuous we also have a different performance metrics. Such as mean absolute error, mean squared error, and R squared. In this lesson, we'll go through all those different performance metrics and explain the definitions and how they're related to each other.

77 Now let's start with performance metrics for a classification problem. In this lesson, we focus on describing metrics for evaluating binary classification problems, for example predicting whether patients would develop heart failure or not. These metrics can be generalized to the setting with multiple classes as well. But in this lesson, we'll focus on the binary setting. There are many evaluation metrics for classification model. Next we'll illustrate how they're all connected. To depending on the prediction and the Ground Truth value, we have four different possible outcomes. True Positive, False Positive, False Negative and True Negative. Next, we explain their definition and relationship to each other. So, the output of a binary predictive model, can be either Positive or Negative. So here you can see a zoom in version of this. To differentiate with the Ground Truth value, we call this Prediction Outcome Positive, and Prediction Outcome Negative. Similarly, we have the Ground Truth value can be either Positive or Negative. Again, to differentiate with the prediction value, we call this Condition Positive and Condition Negative. If the Prediction Outcome is positive, and the Ground Truth's value is also positive, then we call this, True Positive. However, if the Prediction Outcome is positive, but the Ground Truth condition is negative, then we call this False Positive, or Type I error. Conversely, if the Prediction Outcome is negative, and the Ground Truth Condition is positive, we call this False Negative, or Type II error. Finally, a True Negative occur when Prediction Outcome and Ground Truth Condition are both Negative. In fact, this 2x2 matrix is called a Confusion Matrix, or a Contingency table. Now, we understand all the definitions of this basic matrix. Let's look at the relationship among this matrix. First, each row and column of this matrix sum up to the marginal. For example, True Positive plus False Positive equals Prediction Outcome Positive. And the True Positive plus the False Negative equals the Condition Positive. And the Total Population equals the Prediction Outcome Positive plus Prediction Outcome Negative. Or Condition Positive plus Condition Negative.

78 Now let's do a quiz to better understand the relationship of all this metric in this confusion matrix. Go ahead, fill in the true positive, true negative, condition positive, prediction outcome negative and the total population.

79 I hope this quiz is not too confusing for you. Here are the answers. The key insight for solving this quiz is leverage. The row sum and the column sums equals to the marginal. For example, true positive equals prediction outcome positive subtract false positive. For example, 155 minus 100 equal to 55. And the true negative equals the condition negative, subtract the false positive. 935 minus 100 equals to 835. Now you have all those four numbers, and the condition positive can be calculated by taking the sum between true positive and false negative. 65 equals to 55 plus 10. And similarly, prediction outcome negative equals the false negative plus true negative. 10 plus 835 equal to 845. And finally, the total population can be calculated easier by sum up, prediction outcome positive, and prediction outcome negative. Or you can sum up the condition positive with condition negative, so the total population equal to 1000.

80 Knowing this two-by-two confusion matrix is just the beginning. There are many different metrics can be derived from those basic metrics. In particular, many metrics can be derived by taking ratios of different numbers from this basic metrics. First, let's introduce a set of metrics that are derived by normalizing some terms with the ground truth value, either the total population or the condition positive or the condition negative. So, all these metrics are defined by some conative divided by those normalization terms. For example, accuracy is the ratio from the sum of true positive and true negative divided by the total population. Accuracy is the most basic metric that many people have intuitive understanding, but accuracy is not always the best measure when the class label are very imbalanced. If only 1% of the total population have heart failure, models can be achieved 99% accuracy by simply predicting everyone would not have heart failure. So, it is important to use the appropriate metric for measuring the performance of a predictive model. Another important metric is true positive rate or sensitivity or recall. So, they have different names, because the same metric has been developed by different communities, and true positive rate equals the true positive divided by the condition positive in the ground truth. To explain the intuition, let's assume positive means heart failure and negative means without heart failure. The intuition of true positive rate is to measure among all people with heart failure, what percentage is correctly identified by the predictive model? A very related metric can be derived called false negative rate, which equals 1 minus true positive rate. The intuition of false negative rate is among all heart failure patients, what percentage of those patients is missed by the predicting model? False positive rate is the ratio between false positive and the condition negative. The intuition of false positive rate is to measure among all people without heart failure, what percentage of them is incorrectly predicted by the model as to have heart failure? Finally, true negative rate or specificity is the number of true negative divided by the ground truth condition negative, which also equals 1 minus false positive rate. The intuition behind true negative rate is to find among all people without heart failure, what percentage of them is correctly classified as non-heart failure patients? Note that accuracy measures the combination performance of true positive and true negative, while true positive rate measures the performance only about true positive, and true negative rate measures the performance only about true negative. It is often possible to obtain high performance on a single metrics such as accuracy, while it's difficult to obtain high performance on all metrics. That's why we often test a predictive model against multiple metrics. First of all, different metrics are designed for different situations. It is important to choose the proper metric for your prediction study.

81 So now let's do another quiz on those new metric we just learned. Fill in the box based on the number already provided here. Let's calculate the true positive rate, false positive rate, false negative rate, and true negative rate. The detailed formula of all this metrics are provided in the instructor notes.

82 I hope your answers are accurate. The true positive rate can be calculated by taking the true positive divided by the condition positive and that give us 85%, and the true negative rate can be calculated by taking the true negative divided by the condition negative. That give us 89%. Next, for false positive rate, we can calculate that by either taking the false positive divided by the condition negative, or just simply taking one minus the true negative rate, which is 11%. Finally, the false negative rate can be calculated by taking the false negative divided by the condition positive, or simply one minus the true positive rate. That'd give us 15%.

83 So far we have learned all the accuracy metrics which are normalization by the ground truth values. Next, we'll learn another set of metrics that are defined by some commodity divided by the prediction outcomes. Either the prediction outcome positive or prediction outcome negative. First prevalence. Prevalence is the ratio between condition positive and total population. And prevalence measures how likely the disease occurs in the total population. For example, for different disease condition, the prevalence can be very different. For heart failure. Among older population, the prevalence might be quite high such as 20%. For a rare type of cancer, the prevalence of that disease can be quite low maybe 0.001%. Now let's look at positive predictive value or precision. Positive predictor value is the true positive divided by predictive outcome positive. Positive predictive value measures among all patients that are predicted to have heart failure. What percentage of them would actually have heart failure? A relative metrics is called false discovery rate. Which measures the number of False Positives divided by the Prediction Outcome Positives, or 1 minus positive predicted value. The intuition behind False Discovery Rate is that, among all patients that are predicted to have heart failure, what percentage of those predictions is incorrect? Similarly, we can have negative predictive value which defines as the true negative divide it by prediction outcome negative. The intuition behind negative predictive value is among all patient who are predicted to not have heart failure by the model. What percentage of the prediction is correct? And finally, we can also define false omission rate, which is the false negative divided by the prediction outcome negative. The intuition behind false omission rate is among all people who are predictive not to have heart failure by the model, what percentage of them will actually develop heart failure? That means the prediction is inaccurate there.

84 Now let's do another quiz. Please fill in the numbers for Prevalence, Positive Predictive Value, and False Discovery Rate.

85 so here are the answers. Prevalence can be computed by taking the condition positive divided by the total population. So, it's close to seven percent, and the positive predictive value equals the true positive, divided by the prediction outcome positive. We have 35% here. And the false discovery rate can be calculated by taking false positive divided by prediction outcome positive, or simply, one minus positive predictive value, which is 65%. I hope you predicted correctly, all those numbers. [SOUND]

86 Another popular metric is called F1 score. It combines the true positive rate and the positive predictive value. And the F1 score is two times positive predict value times true positive rate divided by the sum of those two measures. In fact, this fancy formulation is really a harmonic mean of those two measures, positive predictive value, and the true positive rate.

87 Let's do another quiz about F1 score. Based on the information in this table, please calculate F1 score.

88 And the answer is close to 0.5. You can calculate the F1 score by taking the true positive rate times the positive predictive value, then divide it by the sum of those two measures. And scale that by 2, which will give you the number 0.5.

89 Now we have learned many different classification metrics. Let's use that to decide which one of this is the best classifier. Here are the different confusion matrix of these three classifiers and the corresponding performance metrics. This is the true positive rate, this is the false positive rate and this false negative rate, and this is true negative rate, and the numbers on the side are the marginal. This are the ground truth's condition positive and the ground truth's condition negative. And this is predicted outcome positive and this predictive outcome negative. And this is the total population.

90 And the answer is C because it has higher performance matrix in all the three measures, positive predict values, F1 score, and accuracy.

91 Now we change the performance measures on classifier C. Can you tell me which one of this is the best classifier?

92 The answer is still C. This might be surprising to many of you. Although A seems to have a higher performance measure than B and C, C can be easily improved by reversing the prediction. We can change the positive to negative, and negative to positive. This way the performance measure will become highest, the same as before. So, in this manner the C classifier will still perform the best.

93 In general, predictor models output continuous prediction score. For example, in binary classification, it will be between 0 and 1. The value closer to 1 means prediction outcome positive, and the value closer to 0 means prediction outcome negative. In this case, we need a threshold as the prediction boundary. This threshold has a significant impact on all the performance metric we have discussed so far. And receiver operating characteristic, or ROC curve, provide a way to compare different classifiers as a prediction boundary is varied, and this curve is created by plotting the true positive rate against the false positive rate at various threshold values. So, such a curve can be generated by going through all the data points in the descending order of their prediction score and using those prediction scores as threshold values. For example, we have 20 patients. Ten of them have positive outcomes indicated by letter p, and ten of them have a negative outcome indicated by letter n. We sort them based on the prediction score. Then we start using those prediction score as threshold points. We start from the point where the true positive rate and the false positive rate are both 0. Next, we pick the threshold value to be 0.9, which means the prediction score greater than or equal to 0.9 will be predicted as positive, and otherwise predicted as negative. In this case, the true positive rate will be 0.1, because only one out of the ten patients with positive outcome are predicted correctly. The false positive rate in this case is 0, because no one has been mis-classified. Next, we change the threshold to 0.8. In this case, two positive patients are classified correctly out of ten. Therefore, the true positive rate improved to 0.2. Still, the false positive rate remained 0 because no mis-classification has happened yet. Now let's change the threshold value to 0.7. Now we actually mis-classified this instance to be positive, but the actual outcome is negative. Therefore, the false positive rate become 0.1, since one out of ten negative examples are mis-classified as positive. The true positive rate remains the same. We can go through all those data points by changing the special values to complete this ROC curve. The ROC curve illustrates overall performance

of a classifier when we're varying the threshold value. One important metric is the area under this ROC curve, or AUC. Since AUC doesn't depend on the choice of the threshold, it becomes the most popular performance metric for classification problems.

94 Now we understand ROC curve, and the concept of AUC. We can use that to choose the best classifier. However, to utilize that classifier to make a prediction, we still have to choose a threshold. Now which of the following would be a good threshold for this classifier? Is this A, 0.8, or B, 0.54, or C, 0.38, or D, 0.3?

95 The answer is it really depends. If you're really curious about false positive rate to be low, then A is a good choice. If you really want to have a high true positive rate, then C and D might be good choices. If you care true positive rate and the false positive rate equally, then B will be the good choice. So, the bottom line is, the optimal classification threshold may vary depending on your preferences.

96 So far we introduced many classification metrics. Next, we'll present some popular regression performance metrics. The most popular regression metrics are mean absolute error, MAE, or mean squared error, MSE. The mean absolute error, MAE, measures the average of the absolute errors, that is the difference between the prediction and the ground truth value, and the mean squared error MSE, on the other hand, measures the average of the squared error. MSE is easier to work with because the derivative of the square term is linear, but MSE will greatly be affected by outliers because of the square term as well. On the other hand, MAE is more robust against the outliers, but it's harder to work with because this absolute value is not differentiable. Here are some visual illustrations. The X axis is the grand truth value, and the Y axis is the prediction. As the amount of noise increases from left to right, both MAE and MSE increase. You can also notice that MSE increased a lot faster because the square of error term.

97 Both mean absolute error and mean squared error, a popular metrics. But they're not bonded in a fixed range, so it's not possible to compare across data sets. Next, we'll introduce another regression metrics called r squared which has a fixed maximum score of 1. So formally r squared, or coefficient of determination is one minus the ratio between MSE and variance. For example, if we have a linear regression model looking like this, the mean square error can be equal to .86, while the variance equals 4.9 and r squared for this particular example is around 0.82, which is considered to be very good. In fact, r squared of 1 indicate the regression is perfectly fits data while r squared of 0 indicate the line does not fit the data at all. It is important to notice that by this definition, it's possible to have negative values of r squared. Which means the predictive model performed worse than a simple average over the original data. Again, the same visual illustration as we increase the noise level, we can see the r squared value also decreases.

06 Ensemble Methods

98 Last time we talked about metrics to evaluate predictive models. Now that we know what we're looking for in our models, let's talk about how we actually develop those models using big data. We'll start by talking about some of the algorithms used in big data analytics like stochastic gradient descent. We'll then talk about ensemble method. A powerful class of machinery algorithms that often lead to the best performing models. In particular, we'll talk about two criteria, bias and variance, and the tradeoff between them. And finally, we'll talk about bagging and boosting, two ensemble methods.

99 Now let's start with gradient descent method for classification. Gradient descent is a basic optimization method that has been widely used in machine learning and data mining applications. Next, let's go to the high-level procedure about how to apply gradient descent method for classification and regression problem. So, remember, the input to any classification or regression problem is a training data set consisting of end pairs of data points, x, and the corresponding target, y. And the final output of the model is usually involving some parameters set. For example, to see the, in linear regression are the linear coefficient beta. And the first step of gradient descent is to specify the likelihood function of input data D given parameter Theta. The likelihood function is really the joint distribution of the data points, given the model parameter theta. In these steps, sometimes it's easier to use log likelihood function instead of likelihood function since log transformation is monotonically increasing and leads to the same optimal as the original likelihood function. But the resulting terms are often a lot easier to manipulate. After we specify the likelihood function, next, we want to find the derivative, also called gradient, of the likelihood function, given theta. This step is a crucial step. Most of the computation really happens in this step, and depending on how likelihood function is specified, finding the derivative can be easy or can be very hard. Once we completed gradient, we'll update the parameter theta by moving the old parameter towards the direction of the gradient. Finally, we repeat this process until it converges which means the theta is no longer changing. And this process illustrates the high-level idea of gradient descent. But the algorithm actually requires some additional tuning parameters. For example, the learning rate or the step size, which controls how far we update the theta based on the gradient. And the step size can be learned through cross-validation. And also note that gradient descent is the simplest gradient based optimization algorithm. There are many other algorithms, are more advanced but still based on gradient computation, such as conjugate gradient, and method. So, in fact, as long as you can specify the formula for computing the gradient, and you can use more advanced optimization method as black box by just simply plugging those gradient computation as a function.

100 Now let's illustrate the gradient descent method using linear regression. Imagine we're giving a data set D, every row corresponding to a patient and every patient is characterized by a set of features, specified in X and we also have an outcome variable Y. For example, in this case, it could be the cost of this patient, and the goal of the model is to map the input feature X from the patient to the output cost Y that this patient incurs. So first, we want to specify the log likely to function for linear regression. In this case, linear regression assumes Gaussian distribution. If you carry out all the calculation based on Gaussian probabilistic distribution, then you will see this log likelihood function involves some constants, subtract a sum over a set of squared terms. And these summation terms is the sum over all the patients. And for each patient, we find the true cost this patient I incurred, subtract what the model predicts. Here this beta are the coefficient from this regression model, and xi are all the features from patient I. So, the intuition is, we want to learn a linear model that convert the input features such as age, gender,

demographics, from this patient to the cost this patient incurred at the hospital. Once we have the log likelihood, we can take the derivative of this log likelihood to find the gradient, which is specified over here. This notation means partial derivative of beta on its base element. The entire gradient is also a vector over all the beta j's. So, the intuition is, we have a set of parameters, one for each feature and we want to create a gradient for each one of those parameters as well. If we look closely at this gridding computation, we notice that it involve a summation over all the patients. And then for each patient, we compute this linear term and scale that by a scaler. So, the intuition is if we want to compute the gradient of a specific feature, for example age, want to know how important is age in this linear regression model, then we have to go through the entire data set to compute this sum and also for each of this term with scaled by the H feature for a specific patient. Once we have this gradient, then we can just update each beta coefficient by moving towards this grading direction. Here, we added the step size, eta, as additional parameter so that you realize, in the real implementation, you have to specify the step size as well. Then again, this process has to be repeated many times until all the betas are converged. So, note that the gridding computation involve going through the entire data set and we have compute this gridding many times. So, this algorithm can be very expensive given a large data set. So next we'll see how we can address that challenge using a slightly different method.

101 So, Stochastic Gradient Descent, or SGD Method, is a variant of gradient descent for handling big data set. So, in traditional Gradient Descent Method, we have to compute the likelihood of the entire data set, then compute the gradient of the entire data set, then repeat, this, many, many times. Which can be too expensive to do on a large data set. The idea of SGD is actually quite simple. Instead of computing the likelihood over the entire data set, the idea is to compute the likelihood function and the gradient on a random subset of data points. For example, B data points. Sometimes, SGD referring specifically to when we update one data point at a time, while for larger B it referred as a mini batch update. But the intuition are the same. We take a subset of data points, compute the likelihood on that subset, then compute the gradient on that subset, then perform that date and repeat. And sample another set of random data points then repeat this process. So compared to traditional gradient descent, SGD method can compute this update much more quickly.

102 Next let's see how we can run stochastic gradient descent for linear regression. Here, we're using the same dataset as before. Every row is a patient, and we have a set of features associated with each patient, such as age, gender, what disease they have and what medication they're taking and the goal is to predict a target Y, which is the cost they will incur at the hospital. So next, if we want to compute SGD on this data set, we randomly sample one patient and compute a log likelihood of this patient. If we look closely at this log likelihood term, we notice that this only involve a single patient, patient i. And intuition here is, it measures how well the model performed for this specific patient. Then we compute the gradient on this patient and the gradient in this case is this linear term, scaled by this corresponding feature j. Then we can perform the update just like before. Using the gradient, and then repeat the process by sampling another patient. Then go through this update. So, this process is very similar to gradient descent for linear regression. So, the main difference is, when we compute the log likelihood and the gradient, only single patient is involved. We don't have to do this for the entire dataset. And this is much more efficient.

103 So far, we've talked about stochastic gradient descent as an efficient method for dealing with large dataset. Next, we introduce ensemble method which is another popular method for classification. Ensemble method combines multiple models to form a better model. Traditionally ensemble method

refers to combining models using the same base learning algorithms. Such as random forest. However, more generally it can also refer to combining models using different algorithms. Ensemble model often outperform other classification method. For example, Netflix price an open competition for the best recommendation algorithm to predict user's rating for movies, Netflix provided a training data set over 100 million ratings that close to half a million user give to 17,000 movies. The goal is to improve 10% in root mean square errors, or RMSE, over the existing Netflix internal algorithm. The competition began in October 2006, and ended in September 2009. Which lasted almost three years. And here are the leaderboards. Notice that the top two teams had the same exact performance gain over the baseline method. And the number one team won simply because they submit earlier. In fact, all the top ranked teams are based on ensemble methods. If we look closely, we can even see the ensemble in their team names. So initially, number 12 BellKor merged with number 10, BigChaos and they became number seven, Bellkor in BigChaos. Then later on BellKor in BigChaos combine with Pragmatic Theory, it becomes the final winner, BellKor's Pragmatic Chaos, and likewise the number two teams is also named The Ensemble. So, as you can see, method really work in practice. Now let's learn some of the most popular ensemble method together.

104 In general ensemble method involves three steps. First given an input data set, we need to generate a set of data set, D1 to DT for subsequent model training. In this step, those data set can be easier generate independently like in bagging, or sequentially like in boosting. And bagging and busting are two different methods we will talk about in this lesson. Second, each data set will be used to train a separate model. M1 to MT. Those models can be independently trained from the input data or there can be dependency among those models as well. Finally, we need to construct an aggregation function F to combine the result from all those t models. This aggregation function can be as simple as taking simple average or taking a weighted average, and the weight are determined by the algorithm. Depending on how we generate the data's and how we trim those models and what aggregation function to use. Different ensemble algorithms can be developed.

105 In order to explain why ensemble method work, we need to understand bias and variance tradeoff. Here's the visual illustration of the concept of bias versus variance. Bias refers to the prediction error due to the wrong modeling assumption. For example, if the model assumes linear relationship between the targets. However, in reality the data do not follow the linear trend, then the difference will be due to the bias of the model. The variance on the other hand, refers to the error from the sensitivity to small fluctuation in the training data set. Ideally, we want a model to have both low variance and low bias. Intuitively, it can be illustrated by the following example. Here we have a two-by-two example, and x-axis shows the low variance versus high variance. While the y-axis shows the low bias versus high bias. The red center in every figure Indicate the ground truths target, and all those blue dots are prediction from the model. In this first example, all the model predictions, those blue dots, are concentrated on the red target. This means the model has low bias and low variance. And this is the optimal scenario for modeling. In the second example, the model prediction are scattered around the center, but not really concentrated in the center. So, this means the model has low bias, but high variance. In this third example the model prediction are clustered at one position, but not on the target. So, this means the model has high bias but low variance. In this case the model can be easily fixed by shifting the prediction towards the center. In the fourth example, the model predictions are scattered and away from the target. This means the model has high bias and high variance, and this is the worst possible situation. Now we understand the intuition behind the bias and variance. Now let's look at the relationship between bias variances. I want

to explore the relationship using this plot. The x-axis is model complexity, and the y-axis is the error from the model. In general, there's a tradeoff between bias and variance. As we can see from this curve, as we increase the complexity of the model, bias decreases. This is because a flexible or complex model can often fit the data better. Therefore, low bias. However, the error coming from variance will increase as we increase the complexity of the model. This is because a complex model is more susceptible to over fitting, hence high variance. So, the total error is error due to the bias, plus the error coming from the variance. Our goal is to find optimal model complexity that balance the right amount of bias and variance that lead to the minimal error. And note that the best model will change from task to task, and from data set to data set. So, it's not possible to expect the same algorithm that always perform well in all cases. Therefore, it's important to understand the trade off and search for the best model for your task.

106 Now we understand the bias and variance tradeoff. Let's do a quiz about model complexity. Here are four regression models, and x axis is input feature and y axis is the output target. Rank all those four models from lowest to highest model complexity. And we have four models here. A is this linear line. B is this curved line. And C, is this wiggly line and D is this flat line. So, rank these four models from lowest to the highest model complexity.

107 And here are the answers. The flat line, D, is the model with the lowest complexity because it has smallest variance, with just a single parameter to specify. That is the height of this line. And this linear line, A, is the second because it is also pretty simple model and can be specified with just two parameters. And the variance of this model is also quite low. And the smooth curve, B, ranked as third. Because it is more complex than the linear line and the flat line. And finally, the wiggly line C is the most complex one, due to the complex shape of this function.

108 Let’s do another quiz, using the same examples. Note that, all those four models are trying to approximate under line data, which are those loop dots. So, which of the models is the best for approximating the underlying data?

109 The answer is B, because B fits the data better than A and D, which means the modeling assumption of B is closer to the underlying data than those linear models. At the same times, the complexity of B is much better than this wiggly line that's specified by C. So, B actually balanced the bias and variance trade off and it's the best model.

110 Now let's talk about some popular ensemble method. Let's start with bagging. Bagging is a simple, yet powerful ensemble strategy, which is named by Leo Brayman. Given an input data set, the idea behind bagging is to take repeated bootstrap samples from input data set. To generate all those sample data set, D1 to DT. Here bootstrap sampling means sampling with replacement from the original data set, then based on the sample data set, we separate models, M1 to MT. Then, finally we classify a new data point by taking the majority vote or average of all those models.

111 Bagging is one general strategy for ensemble. Random forest is a classical bagging algorithm, where the underlying models are decision trees. So, now let's talk about random forest, how it works. Imagine we have a large patient database. Every dot's here corresponding to a patient, and each patient is represented by a high dimensional feature vectors, such as age, gender, diagnosis, medication. Then we can represent those patients by a matrix, for example, we have N patients and D features. Every row is a patient, every column is the features. That's our entire data set. Then Random Forest will randomly sample subset of patients in every row corresponding to a patient in the original database and every

column corresponding to a feature. Then Random Forest will randomly sample a subset of patient and a subset of features to construct a sub-matrix, like D1 or D2. Then Random Forest will use those sub-matrices as their input data to build separate decision tree. And the way how the tree are constructed is simply recursively select a feature from the sub-matrix and split based on that feature and repeat this process until all the features in this sub-matrix are used. In this manner we would generate multiple trees. Once we have all those models, when a new patient comes we can score that patient against all those models, then taking an average, and use that average as our final prediction. Note that the algorithm for constructing those trees in Random Forest is much simpler than a traditional decision tree algorithm, such as C5. This choice is intentional. In fact, there are theoretical studies showing that it's actually important to use those simple methods, so that we generate a diverse set of models in bagging. The other benefit for using those simple algorithm is computational cost will be dramatically reduced.

112 So why does bagging work? The reason is because, bagging reduces the variance of the final model without increasing the bias. Since the final model is just take the mean over multiple model, the mean of the final model is the same as the individual model. Therefore, bias is the same. However, the variance of the model can be reduced by averaging. So, the infusion come from simple statistic. That the variance of a random variable X, can be reduced by a factor of T, if we measure the variance of the mean over T independent identically distributed samples of x. So, in backing, all the models are viewed along both trap samples, which can be considered close to IID. Therefore, the variance of the final model can be greatly reduced by averaging.

113 So boosting involve incrementally building those models one at a time and emphasized the later model to the training instances that the previous model misclassified. For example, so we start with building model M1 on the first data set, D1. Then we test the performance of model M1 on data set D2. Then based on what mistakes, what misclassification has happened on D2, we train another model M2 to really focus on correct those misclassification has happened because of M1. Then we repeat this process by combining M1 and M2 together and test it against the third data set, D3. Again, these three is trying to correct the mistakes, the combined model of M1 and M2 misclassified, then repeat this process again on D4, and to get the final model M4. And the final model become a weighted average of all the past model we have trained. So, in some cases, boosting has shown to be yield better accuracy than bagging, but it also tend to be more likely to over fit the training data. By far the most popular boosting algorithm is Eta-Boost. But there are many other boosting algorithms out there as well. So, more details will be provided in the instructor note.

114 Here's a quiz to compare bagging and boosting. Here, every rows are some characteristic of the method. First, how do we combine those different models, it says by taking simple average, or weighted average, or can the method be done in parallel? Is it easy or hard? And how sensitive is the method towards noise? It's more sensitive or less sensitive? And finally, what about the accuracy?

115 So here are the answers. In bagging, we use simple average to combine all those models, because the models are developed independently. And in boosting, we take weighted average, because there is a sequential dependency among all those models. In terms of parallel computing, bagging is very easy because all those models are independent. Then you can use those models separately on different course, or different machines. Parallel computing for bagging is easy. But, for boosting, parallel computing is hard because there is a sequential dependency between the early model to the later model. We can't really compute the later model in parallel with early model. In term of noise, bagging is less sensitive to noise

because the model being combined are oftentimes much more diverse, since they're viewed independently on the separate samples. Well, for boosting, it's more sensitive to noise because the sequential dependency and later model, hard to correct errors that made by the earlier models, and it can over fit the data, therefore sensitive to noise. Finally, in term of accuracy, bagging often achieve good performance in all cases, while boosting, in most of the cases, give a better result, but because of over fitting problem, sometimes they perform much worse in some cases. So, the knowledge is bagging is like a Japanese car. It always gets the job done. It's not always the fastest car, but it's a pretty reliable car. And boosting is like a sports car. It's much faster when it works, but when it has a problem, it can cost you a lot. [SOUND]

116 So in this lecture, we talked about ensemble method. So now, let's summarize the overall pros and cons of ensemble method. In term of advantages, ensemble method are often simple. Almost have no parameters, except how many models to be combined. And it's quite flexible, you can combine many algorithms together. And there's many different ways how to combine them. And there are also theoretical guarantee wide works. In term of disadvantages, it's mostly coming from computational challenges. Because now, we have to build multiple models, not only at the training time, but also at the scoring time. Every time we want to score a patient instead of scoring that patient against single model. Now we have to score that against three different models. The other disadvantage is, is lack of interpretation. Because the final model is a combination of multiple models. The interpretation oftentimes can be very difficult.

07 Computational Phenotyping

117 In this lesson, we're going to discuss a healthcare application of clustering called phenotyping. Phenotypes are medical concepts such as diseases or conditions. We know many phenotypes of patients based on existing medical knowledge such as major diseases. But there are many more phenotypes and their subtypes out there that we haven't discovered. Computational phenotyping is a way to use data available to us to discover those novel phenotypes. Phenotypes aren't just for disease diagnosis, though. We can also use those phenotypes for predicting healthcare cost, readmission risk, and supporting genomic studies.

118 Now let's talk about computational phenotyping. So, recall, computational phenotyping is about converting raw electronic health record through phenotyping algorithms into a set of meaningful medical concepts, or phenotypes. For example, a specific disease can be a phenotype. Such as type 2 diabetes. And the raw data, in this case, consists of many different sources. Such as demographics about patients, diagnosis code, medication information, clinical procedure, lab result, and clinical notes. There are many reasons why phenotypes are not represented consistently or reliably in the raw data. First, the data are noisy, there are missing data and raw information in the raw data. And second, the main usage of this data is to support clinical operations, such as billing. And it's not designed directly for supporting research. Third, there are many overlapping and redundant information. For example, diagnosis information can be found in the structure field corresponding to diagnosis code. But the same information can also be present as end structure information in the clinical notes. This information is overlapping and redundant in the raw data. And phenotyping is this process of deriving research grade phenotypes from clinical data, using computer algorithms.

119 Here's a phenotyping algorithm for Type 2 diabetes. And the goal here is, we want to determine whether patient has Type 2 diabetes based on her electronic health records. For example, we can first check whether the patient has Type 1 diabetes code in her record. If no, then we check whether Type 2 diabetes diagnosis are present in her record. If still no, then we check medication for Type 2 diabetes. Then, check abnormal lab result related to diabetes. If these two steps are confirmed then we confirm she has Type 2 diabetes. And there are many different paths that can lead to the confirmation of Type 2 diabetes, and this decision flow is a phenotyping algorithm. And this was developed manually by clinical experts. More details about this algorithm can be found in the instructor note.

120 There are many different applications that require phenotyping. For example, genomic study, which is about finding relationship between genomic data and phenotypic data. Clinical predictive modeling, which is about building an accurate, robust, and interpretable prediction model about disease onset and other related targets, such as hospitalization. And pragmatic clinical trials, which is about comparing treatment effectiveness in the real-world clinical environment using observational data, like electronic health records. And healthcare quality measurement, which is about measuring efficiency and quality of care across different hospitals. All those applications depends on phenotyping algorithms. We'll show them in more details next.

121 Phenotyping algorithm is very important in supporting genome-wide association study. What is a genome-wide association study? It's an approach that involves scanning biomarkers such as single nucleotide polymorphism. Or SNP's from DNA of many people, in order to find genetic variation, associated with a particular disease field phenotypes. Once new genetic associations are identified, researchers can use that information to develop better strategies. To detect and treat and prevent

diseases. So, how are genomic wide-association studies conducted? To run a genomic wide-association study, or GWAS, we first identify the disease phenotypes. Then group the participants into these two groups, cases and these are people with disease phenotypes. And controls, those are similar patient without the disease phenotype. Then we need to obtain DNA samples from all these participants. >From DNA samples, we can use lab machines to quickly survey each participant's genomes for genetic variation. Which are called single-nucleotide polymorphism or SNP. If certain genetic variation have found to be significantly more frequent in the people with the disease phenotypes compared to people without the disease phenotype. These variations are said to be associated with the disease. We do this by computing the frequencies of SNPs on the cases and on the controls. Then based on the frequency, we calculate the odds ratio. >From there, we can calculate the corresponding p-value for the odds ratio. If the p-value is small, then we conclude this variation is significant. The associated genetic variations can serve as powerful pointers to the region of the human genome that may cause the disease. Here's an example showing more details on how the GWAS is computed. We first identified the cases and controls. That is, the people with the disease phenotype and the people without the disease phenotype. In this case, we have 4000 patients with the disease phenotype and 6000 patients without the disease phenotype. Then we iterate over all the SNPs to compare the relevant frequencies. For instance, for SNP1 for the control group, we have 2676 out of 6000 has the corresponding variation G, at this location. And the frequency of G in this case is 44.6%. in the case group, we have 2104 out of 4000 with the corresponding variation G at this location. So, the frequency is 52.6%. If we go through the calculation, now we'll find the P-value is 5 times 10 to the minus 15. Which means, this is extremely significant. We can conduct the same calculation on SNP2 and find out the P-value here is 0.33 which is not significant. And there's support GWAS study, we need to know high quality phenotypes on the cases and controls. In order to perform this calculation, that's why phenotyping algorithm, is very important.

122 As we have shown in the genome-wide association studies, we need phenotypic data in order to analyze genomic data. But in general, why do we care about phenotyping algorithms in genomic study? In fact, many people argue that we need rich and deep phenotypic data in order to analyze genomic data. Especially as sequencing technology improves, the cost of generating genomic data is dropping so fast over time. While the cost of computing or Moore's Law cannot keep up with the improvement of sequencing technology, which means we'll have more and more genomic data in near future about many individuals. However, due to the complexity of the electronic health record, the cost for generating high quality phenotypic data is actually increasing, while the cost of genomic data is dropping drastically. That is why we really need to invent better phenotyping algorithms to reduce the cost of acquiring high quality phenotypic data and to support genomic studies.

123 Finger typing algorithms can also help with clinical predictive modeling. We have talked about predictive modeling in other lectures. Clinical predictive modeling starts with the raw EHR data as the input, then goes through the predictive modeling phase. Then we come up with accurate predictive model, such as predicting whether a patient will develop heart failure or not in the next six months. There are many problems with predictive modeling directly on the raw data. First, as we all know, the raw data are very noisy. The resulting model may not perform as well because of the noise in the data. The raw data are also very complex and height dimensional. The resulting model may be difficult to interpret. Third, because the model is tied directly to the raw data, it can be difficult to adapt the model from one hospital to another, because their input data format can be different. So, instead of directly modeling over the raw data, we can first convert the raw data through phenotyping to high quality, low dimensional

medical concept, or phenotypes. Then use the phenotypes as input to the predictive modeling process to get the accurate model. So, this way we can remove a lot of noise from the raw data thanks to the phenotyping algorithm. We can also get better interpretive model since the input to the model are meaningful phenotypes instead of complex raw data from EHR. The resulting model can be applied across hospitals because the input are general phenotypes, as opposed to a specific data format from a hospital.

124 Another application of phenotyping algorithms is to support pragmatic clinical trials. So clinical trials can be described as either traditional trials or pragmatic trials. So traditional clinical trials generally measure efficacy, which means the benefit that treatment produces under ideal conditions. So, it deals with one condition. The pragmatic trial deals with real-world patients, often have multiple conditions simultaneously. In the traditional trials, we only test one drug at a time. In the pragmatic trials, patients potentially can take multiple drugs at the same time. In the traditional trial, randomization is required, which means some patient will be given the drug, some patient will be given a placebo. And this randomization is very important because it deal with the bias in the clinical research. However, in the pragmatic trials, randomization is often not possible because it's real-world environment. The other difference is traditional clinical trials recruits homogeneous population. In the traditional trials, the patients are carefully selected, often with very strict inclusion and exclusion criteria. So, for pragmatic trials, there's no patient selection criterias introduced, any patient can be potentially included. So, in summary, traditional trial really is designed with a very well controlled environment, while pragmatic trials deal with real world environment. High quality phenotyping algorithms are very important for pragmatic trials because we need to know what disease condition patient has and what medication they're on. Those are all can be derived as phenotypes.

125 Phenotyping algorithm is also very important for house care quality measures. It is important to compare house quality measure across hospitals. One way for doing that is to have all those hospital sending their raw EHR data to the central side. The central side can be an insurance company or public health agency such as Centers for Disease Control. Then the central side has to aggregate all those raw information in order to compute all those health care quality measure. And this become very difficult task because all those hospital can use very different format to represent their raw data and this center side has to figure it out how to process them differently. In more scalable way for dealing with this problem was to process all those raw EHR data through phenotyping first, then obtain the high-quality phenotypic information then share that with the central site. With those consistent phenotypic information sending from different hospitals, now the central side can aggregate those information to compute the healthcare quality measures then compare them across hospitals. So, in this case, high quality and consistent phenotypic data are crucial to enable this house care quality measure comparison across hospitals.

126 Now understand phenotyping is a very important process, then what are the phenotyping method? There are two main categories of phenotyping method, supervised learning, and unsupervised learning. They're actually corresponding to two important topics in machinery. I'll have Charles and Michael to explain what they are from Machine Learning. So, what do you think supervised learning is? I think of supervised learning as being the problem of taking labeled data sets, gleaning information from it, so that you can label new data sets. That's fair. I call that function approximation. So, here's an example of supervised learning. I'm going to give you an input and an output. And I'm going to give them to you as pairs, and I want you to guess what the function is. Sure. Okay, okay. 1 1, 2 4. Wait, hang on, is one the input, and one the output? Yes. And 2 the input, 4 the output? Correct. Okay, I think I'm on to you. 3 9,

4 16, 5 25, 6 36, 7 49. Nice, this is very hip data set. It is. What's the function? It's hip to be squared. Exactly, maybe. Now if you believe that's true, then tell me if the input is ten, what's the output? A hundred. And that's right if it turns out in fact that the function is x squared. But the truth is we have no idea whether the function is x squared or not, not really. I have a pretty good idea. You do? Where does that idea come from? And it comes from having spoken with you over a long period of time and plus math. And plus, math. Well, I'm going to- You can't say I'm wrong. You're wrong. You just said I was wrong. Yeah, I did. No, you've talked to me for a long time and plus math, I agree with that. Okay. But I'm going to claim that you're making a leap of faith, despite being a scientist, by deciding that the input is 10, and the output is 100. Sure, I would agree with that. What's that leap of faith? Well, I mean, from what you told me, it's still consistent with lots of other mappings from input to output. Like 10 gets mapped to 11. Right, or everything's x squared except ten. Sure. Where everything's x squared up to ten. Right, that would be mean. That would be mean. But it's not logically impossible. Or would it be the median? Ha. Thank you very much, man. I was saving that one up. What about unsupervised learning? Right, so unsupervised learning, we don't get those examples. We have just essentially something like input. And we have to derive some structure from them just by looking at the relationship between the inputs themselves. Right, so give me an example of that. So, when you're studying different kinds of animals, say, even as a kid, you might start to say, there's these animals that all look kind of the same. They're all four-legged. I'm going to call of them dogs, even if they happen to be horses or cows or whatever. But I have developed, without anyone telling me, this sort of notion that all these belong in the same class, and it's different from things like trees. Which don't have four legs. Well, some do, but I mean, they both bark, is all I'm saying. Did I really set you up for that? Not on purpose. I'm sorry, I want to apologize to each and every one of you for that. But that was pretty good. Michael's very good at word play, which I guess is often unsupervised as well. No, I get a lot of supervision. You certainly get a lot of feedback. Yeah, that's right, please stop with that. So, if supervised learning is about function approximation, then unsupervised learning is about description. It's about taking a set of data and figuring out how you might divide it up in one way or the other. Or maybe even summarization. It's not just a description, but it's a shorter description. Yeah, it's usually a concise, compact- Compression. Description. So, I might take a bunch of pixels like I have here and might say male. Wait, wait, wait, wait, I'm pixels now? As far as we can tell. 100 That's fine. 101 I however am not pixels. 102 I know I'm not pixels. 103 I'm pretty sure the rest of you are pixels. 104 That's right. So, I have a bunch of pixels and 105 I might say male, or I might say female, or I might say dog, or I might say tree, 106 but the point is I don't have a bunch of labels that say, dog, tree, 107 male, or female, I just decide that pixels like this belong with pixels like 108 this as opposed to pixels like something else that I'm pointing to behind me. 109 Yeah, we're living in a world right now that is devoid of any other object. 110 Chairs. 111 Chairs, right. 112 We got chairs. 113 So these pixels are very different from those pixels, 114 because of where they are relative to the other pixels. 115 Exactly, right? 116 So if you were- I'm 117 not sure that's helping me understand unsupervised learning. 118 Go outside and look at a crowd of people and 119 try to decide how you might divide them up. 120 Maybe you'll divide them up by ethnicity. 121 Maybe you'll divide them up whether they have purposely shaven their hair in 122 order to mock the bald, or whether they have curly hair. 123 Maybe you'll divide them up by whether they have goatees- 124 Facial hair. 125 Or whether they have gray hair. 126 There are lots of things that you might do in order- 127 Did you just point at me and 128 say gray hair? 129 I was pointing, and your head happened to be there. 130 Come on. Where's the gray hair? 131 Right there. 132 It's right where your split curl is. 133 All right. 134 Okay, so imagine you're dividing a world up that way. 135 You can divide it up male and female. 136 You can divide it up short and tall, wears hats, doesn't

wear hats, 137 all kinds of ways you can divide it up, and 138 no one's telling you the right way to divide it up, at least not directly. 139 That's unsupervised learning. 140 That's description, because now, rather than having to send pixels of 141 everyone or having to do a complete description of this crowd, 142 you can say there were 57 males and 23 females exactly. 143 Or there are mostly people with beards or whatever. 144 I like summarization for that. 145 I like summarization for that, it's a nice concise description. 146 Good. That's unsupervised learning. 147 Very good. 148 And that's different from supervised learning. 149 It's different from supervised learning, and 150 it's different in a couple of ways. 151 One way that it's different is all of those ways that we could have divided up 152 the world? 153 In some sense, they're all equally good. 154 So I can divide it by sex, or I can divide it by height, or 155 I can divide it by clothing or whatever, and they're all equally good absence 156 some other signal later telling you how you should be dividing up the world. 157 But supervised learning directly tells you, here's a signal, 158 this is what it ought to be, and that's how you train. 159 They're very different. 160 But I can see ways that unsupervised learning could be helpful in 161 the supervised setting. 162 Right, so if I do get a nice description, and it's the right kind of 163 description, it might help me do the function approximation better. 164 Right, so instead of taking pixels as input and labels like male or female, 165 I could just simply take summarization of you, like, how much hair, relative, 166 the height to weight, and various things like that might help me do it, 167 that's right. 168 And by the way, in practice, 169 this turns out to be things like tensity estimation. 170 We do end up turning it into statics at the end of the day. 171 Often. 172 It was statics from the beginning, but when you say density estimation- 173 Yes. 174 Are you saying I'm stupid? 175 No. 176 All right, so what is density estimation? 177 Well, they'll have to take the class to find out. 178 I see. 179 Okay.

127 One way to defining Supervised Learning is to develop expert-defined rules like the ones we have seen in the early slide for type 2 diabetes. And this is probably the most widely adopted method for phenotyping. And this approach begins with manually develop the algorithm, often use Boolean logic or scoring threshold or decision tree based on domain expertise. Then the logic is iteratively enhanced through validation and chart review on EHR data. So, the advantage of this approach is, it provides a human interpretable algorithm. The number of chart review to validate this algorithm can be low because often times the expert can come up with a pretty good algorithm to start with. However, the effort and time for developing such an algorithm can be significant because it requires clinical and informatic knowledge. And this approach cannot be used to identify phenotypes that are not well understood by the clinical experts. We can also use supervised machine learning to train a classifier to differentiate the cases and the controls. Depending on the algorithm classification models sometimes can be difficult to interpret. And it requires significant amount of training data, and it may not transfer well from one hospital to another. As the model may learn features that are unique to a specific hospital. Unsupervised learning provide approaches to cluster EHR data into patient groups corresponding to phenotypes or subtypes. Unsupervised learning does not require expert labels which tremendously reduce the time needed for manual chart review. However, the validation of the resulting phenotypes can be challenging because there's no ground on what those phenotypes are. While this method often require very large amount of treating data, they do not carry the cost of manually labeling individuals as cases or controls, as what is required in the supervised method. So, example of unsupervised learning for phenotyping include dimensionality reduction, such as modeling and tensor factorization. We'll talk more about them in other lectures.

128 So here a quiz of phenotyping missile. Which of the phenotyping approach require more human effort during evaluation? Is it expert-defined rules or classification models? And which phenotyping approach is easier to interpret? Is it expert-defined rules or classification models?

129 And the answer is classification models often require more human effort during evaluation, because they require a large amount of label training data in order to be a good model. However, the expert-defined rules, because the quality of those rules are often very good, so, during evaluation phase, it doesn't require a lot of human effort. The expert-defined rules are easier to interpret because they're designed directly by clinical experts, follows clinical intuition and knowledge. While the classification models sometimes can be difficult to interpret because they're derived directly from data, may or may not follow the clinical intuition.

08 Clustering

130 Now we're going to move on to a different method called clustering. In classification, we group data by label or categories that we already knew. But in clustering, we want to discover those labels or categories form raw data. We'll start by defining clustering. Then we'll describe some algorithms for clustering, such as K-means and Gaussian mixture models. Then, we'll describe scalable classroom algorithms for handling big data sets. Finally, we'll preview some of the healthcare applications of clustering.

131 There are many healthcare applications for clustering. For example, patient stratification is about grouping patients into clusters such that patient within the same cluster share many common characteristics, and patients across cluster share very few common characteristics. And disease hierarchy discovery is about learning a hierarchical structure about all the diseases and how they relate to each other from data. And phenotyping is about converting raw data into meaningful clinical concepts or phenotypes. The phenotypes is actually a cluster of patients that share some commonalities.

132 So what is clustering? Let's illustrate clustering using the following example. Imagine we have a patient by disease matrix, where the columns are all the different diseases, and rows are different patients. And other elements indicate whether a specific patient has a specific disease. If we want to apply clustering on the rows of this matrix, which means you want to learn a function f that partitions this matrix into a set of groups. Each group corresponding to a set of patients, P1, P2 and P3. Once we have all those patient groups or patient clusters, we can utilize those to support application such as phenotyping and patient stratification. And similarly, we can apply clustering on the columns of the matrix to partition all the diseases into different disease group, such as D1,D2, and D3. And we can further group all those disease groups into a hierarchy, and this will support the disease discovery application.

133 In this class we'll introduce two sets of clustering algorithms. The classical clustering algorithms and the scalable clustering algorithms. For classical algorithms, we'll discuss K-means, hierarchical clustering, and Gaussian mixture model. For scalable algorithms, we'll introduce mini batch K-means and DBScan.

134 Now let's talk about K means clustering. So, the input to the K means algorithm is a set of data points X1 X2 to Xn and the perimeter K indicate the number of clusters. And the output of K means clustering are K clusters S1, S2 to SK and each cluster contains a set of data points and the union of all those clusters give us all the data points. The objective of K-means clustering is it minimize the sum of all the data points, x, to its corresponding center, mu i. Here mu i is the center for cluster Si. In order to minimize this objective, we want to find the optimal assignment of all this data points into K cluster, such that this term is minimized. Now let's talk about the algorithm for K means. First we specified K as the number of clusters. Then we initialized K centers. Then we'll assign each data point to its closest center. Once we have all the assignments we update the K centers. For this step we update the new center mu i by averaging all the data points within the cluster Si. Then we check whether the algorithm converged or not. Which means we can check whether any cluster assignment has changed from the previous iteration. Or some pre-defined maximum number of iteration has been reached. Then we iterate into the convergence criteria as math then we output the clusters. And this is the K means algorithm. Now, let's visually illustrate how K means works using this example. We want to run K means algorithm on this set of points with K = 2. To start, let's initialize the two-cluster center with the blue cross and red cross. Then we'll assign all the data points to as close the center. And all those blue points are assigned to the blue

cluster, and all the red points are assigned to the red cluster. Then we find the new cluster center by averaging all the blue points and all the red points. So, this blue and red cross are the two new centers. Then we iterate this process over three iterations, and two had converged. This is the final result when we want K means with setup points, with K equal to two.

135 Now let's do a quiz on K-means. Given n is the number of data points, k is the number of clusters, and d is dimensionality of each data points, and i is the number of iteration of the K-means algorithm, what is the computational complexity of K-means? This is the algorithmic illustration to help you find the answer and put your answer in this box.

136 And the answer is big O of n times k times d times i. Now let's see how we got this answer. When we run k-means algorithm most of the computation goes to clustering assignment and center update. For clustering assignment, we need to compare end data points to K centers. So that takes n times k comparisons. Since each comparison is order d operation, because each datapoint has d dimensions and the total complexity for each iteration is n times k times d. Similarly, we can derive the center update of the same complexity and the total complexity of k means algorithms becomes n times k times d times i.

137 Next let's introduce hierarchical clustering. The goal of hierarchical clustering is to learn hierarchy over a set of data points. For example, given five different diseases, we want to construct such a hierarchy so that similar diseases are grouped together into this tree structure. There are two main ways for doing this. One is the bottom-up approach called agglomerative method. We'll start with individual disease, d1 to d5, as their own clusters, then interactively group those smaller clusters into bigger clusters, until only one cluster remains. For example, d1 and d2 will first be grouped together, then d4, d5 will group together. Then subsequently, d3 and d4, d5 will be grouped together and finally, all five diseases will be merged into one cluster. The other approach is a top-down approach called a divisive method. We'll start with a single cluster with all the diseases in it, then iteratively divide the bigger cluster into smaller clusters, and thus, the divisive method. In general, this agglomerative method bottom-up approach is more efficient, therefore, more popular in practice.

138 Now let's talk about agglomerative clustering algorithm. We'll start with a set of data points x1 to xn. First we compute all paired distance matrix. This will be a n-by-n matrix, and every element in this matrix corresponding to the distance between the pair of points. Then we initialize each data points as their own cluster. So, we have n cluster here. Then we check how many clusters are left. If we only have one cluster left, that's it, that's the final result. We output the hierarchy. If we have more than one cluster, we merge the closest clusters. Then we update the distance matrix. We will run this algorithm for the first times, we'll start with a n-by-n distance matrix, then we merge two closest clusters, and update the distance matrix, here the distance matrix will be of size n- 1 by n- 1. Then we iterate, continue to merge to a closest cluster together and also update the distance matrix until we have only one cluster left, and that's the final result.

139 Next we'll talk about Gaussian mixture model. So far we have talked about two clustering algorithm, K-means, and hierarchical clustering. They are both hard clustering method, which means every data points only belong to a single cluster, and Gaussian mixture model is a soft clustering method. Soft? Yes, soft clustering. So, every data points can belong to multiple clusters with a different degree. So, first, what is a Gaussian? Gaussian distribution is the most popular continuous probability distribution, and it's also known as the normal distribution. The shape of Gaussian is a bell curve, like this, and Gaussian distribution has two parameters, mu, and sigma. And the mu variable corresponding to the

center of the bell curve, and the variance sigma capture the spread of this bell curve. When the variance is small, the spread will be small, which means all the probability will be concentrated around the mean. When the variance is large, the probability will be scattered, which means the events that are far away from the mean can still happen with large probability. Now let's introduce the mathematical definition of Gaussian Mixture Model. The probability of a data point X is the weighted sum over k Gaussian distribution. Here at the pi k is the mixing coefficient for cluster k. Intuitively, it tells us how big the cluster k is, and mu k is the mean of cluster k or the cluster center for cluster k, and sigma k is the co-variance for cluster k. So, here's an example when we have a two Gaussian distribution and data points x will be generated from this two underlying Gaussians. For a Gaussian mixture model, the goal is to figure out the parameters of the two underlying Gaussians, namely finding out pi k, mu k, and sigma k, given all the observed data points.

140 To compute a Gaussian mixture model, we utilize a popular optimization strategy called expectation maximization, or Algorithm. Next, let's see how Algorithm works for Gaussian mixture model. We first initialize all the parameters for Gaussian mixture model and could mix in coefficient pi k. The center mu k and variance sigma k. Then in the E Step, we assign each data point Xn with a score gamma nk for each cluster K. So, in this case, data points Xn will have K scores, one for each cluster. In particular, gamma nk tells us how likely Xn is generated from cluster K. Once we have all the assignments, then in the M Steps we update all those model parameter, pi k, mu k, sigma k, using the scores we have learned from those assignments. Then we check if convergence criteria are met. For example, likelihood is no longer changing, or parameters are stabilized. If no, we continue this Iterations into convergence. If yes, we return all the model parameters for Gaussian mixture model. Next, let's look at those steps in more details.

141 In order to run Algorithms for Gaussian mixture model, we have to initialize all those parameters. In particular, we have to find the mixing coefficients, the centers, and variance for each clusters. And this initialization step is very important. A bad initialization can cause many subsequent iterations into convergence. A good initialization can save a lot of iteration so that the convergence can happen very quickly. For example, in this picture, if we initialized these two clusters here and here as the center and also give equal weights for the mixing coefficients, this will be a pretty bad initialization, because the starting point of these two-cluster center do not coincide with any data points. Which means subsequently the Algorithm has to iterate many times to correct this bad initialization. So, what will be a better initialization? We can actually use K-means result to initialize for Gaussian mixture model. For example, the center in the Gaussian mixture model will be the center from K-means result. The covariance matrix sigma K can be computed by using all the data points from the corresponding clusters. And finally, the mixing coefficient pi k can be simply computed as the size of the cluster K divided by the total number of data points. And usually this initialization with K-means result will be much better than a random initialization like this. Next, let's talk about the E step. In this step, we'll assign each data point a score gamma nk for each cluster K. And the gamma nk can be calculated with this equation. And the numerator has two terms, pi k, the mixing coefficient for cluster K, the probability of xn in cluster K. And the denominator has to normalize this assignment score to between zero and one. For example, we have one blue Gaussian over here and one orange Gaussian over here. The assignment score for this point is 0.5 for the blue Gaussian and 0.5 for the orange Gaussian. Which means this point is equally likely to belong to the blue cluster or the orange cluster. The assignment score for this point is 0.8 for blue Gaussian, and 0.2 for orange Gaussian. Which means this coin is more likely to come from this blue cluster.

And the E step is to go through all these data points by assigning all these assignment scores. Now let's talk about the M step of Gaussian mixture model. In this step, we'll update all the model parameter pi k, mu k, sigma k, using all the assignment score we have learned from the E step for each cluster K. First let's define an auxiliary term Nk, which equals the sum of all gamma n k for a specific k. And intuitively, Nk is the size of cluster k. The we calculate mu k equals the weighted sum of the data points in cluster k divided by Nk, the size of cluster k. And intuitively, mu k is the center for cluster k. Then the covariance sigma k can be computed with this equation. These particular terms corresponding to the covariance from the data points xn. Intuitively, sigma k is the covariance of all data points in cluster k. And finally, the mixing coefficient pi k is proportional to the size of the cluster Nk. In other words, pi k is a prior probability for data points to belong to cluster k.

142 Now let's visually illustrate how Gaussian mixture model works. The goal is to run Gaussian mixture model over this set of points. We first initialize two Gaussians indicated by the blue and red circles. Then we compute assignment scores for each data points. The bluer points indicate the assignment score for the blue cluster are higher. Likewise, the redder points indicate the assignment score for the red cluster are higher. Then we update the Gaussian mixture model's parameters based on the new assignment scores. In particular we update mu, sigma, and pi. Then we iterate the E step and N steps until converges. And here are the final result of Gaussian mixture model after 20 iterations

143 Now let's compare two clustering method. K-means and Gaussian Mixture Model. In term of clustering algorithm itself, K-mean is a hard clustering algorithm. Each data points only belong to one cluster. Gaussian Mixture is a soft clustering algorithm. Each data points can belong to multiple clusters with different weights. Or K-means the only parameter is the center, mu k, for each cluster, but for Gaussian mixture, we have three parameters. In addition to the center, mu k, we also have mixing coefficient, pi k, and covariance, sigma k. Both K-means and Gauss mixture model utilize Algorithms where they iteratively update the clustering assignment and model parameters.

144 So far, we'll have talked about three classical clustering algorithm. K-mean, hierarchical clustering, and Gaussian mixture model. Next, we'll describe two scalable clustering algorithm. Mini batch K-means and DBScan. So, one problem with K-mean's algorithm is that it needs to assign all the data points to cluster centers at each iteration. When we have a billion data points, this can be very expensive. Mini-batch K-means avoids doing these global assignments by working with smaller batches. And here's the high-level algorithm. To start, we initialize K centers as a set C. And there are many different ways for this initialization. For example, we can randomly sample K data points as the centers. Then we update those centers t times, as the following. We first sample b data points to form a batch M. Then we assign the data points in M to the closest center in C. Then we update the centers based on the assignments in M. Next let's look at step b and c and more details. In step b, for each data points x in mini batch M, we'll first find it closest center C to x, then cache this result in the hash map d[x] so that later we can retrieve this center quickly. Next, in step C we update the center based on the assignments in this mini-batch M. In this step, again we go through all the data points x in this mini-batch. First, we retrieve the corresponding center, then we increment the corresponding center count by one. Then we set the step size as the inverse of the center count. Finally, we update the center by 1- eta times the old center c + eta times this data points x. So intuitively, we move the old center towards this data point x by the step size eta. So, note that as the center count increases, the step size becomes smaller and smaller. As a result, the center update becomes smaller and smaller over time. Therefore, the center will eventually be stabilized.

145 Now let's do a quiz on our mini batch k-means. Given k is the number of cluster centers, t is number of iterations, and b is the batch size, and d is dimensionality of the data points. What is the computational complexity of mini batch k-means? And here is the pseudo code of the algorithm to help you answer the question and put your answer in this box.

146 And the answer is, big olive, t time b times k times d. The reason is the most expensive step in this algorithm is to assign data points to its closest center. Since we have b data points in each batch and K centered. So, we have to compute k times b comparisons. And each data point is of dimensionality d, so each iteration, the total cost is b times k times d. Since we iterate this algorithm t times, the total cost becomes t times b times k times d.

147 Now let's talk about DBSCAN algorithm. DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. The main intuition behind DBSCAN is to define clusters as areas of high densities separated by areas of low density. As a result, oftentimes the cluster found by DBSCAN can be of any shape. Here's an example. If we run DBSCAN on this data set, we'll have two clusters. The blue areas is one cluster, and the red area is another cluster. Both are defined by high-density regions, and they are separated by low-density regions. More details about the algorithm can be found in the instructor notes.

148 In order to explain DBSCAN algorithm we have to introduce some key concepts. First, how do we measure the density? In DBSCAN, the density at a point p is defined as the number of points within absolute distance to p. For example, here are a set of points in two-dimensional space. This point A was equal to one, has density three, because there are three points in this circle. Then what is the dense region? At data point P, in a dense region, of the density of P is greater than some threshold, the mean data points. For example, this data point B is in the dense region, whereas the mean point equal to four. Because there are more than four points within radius one to data point B, but A is not in a dense region, because there are only three points within radius one to A, which is less than the mean points four. Now we understand the density and what is the dense region. Next we can define a set of key concepts. Core, border, and noise. Core points are points in the dense region. For example, B is a core point because it's in the dense region. And the border points, are points was in absolute distance to a core point. For example, A is a border point because A was within absolute distance with B and B is a core point. And all the other points outside absolute distance to the core points are noise.

149 Now we understand the key concept of core point, border points, and noise. We can use that to explain the algorithm. To start, for each core point c, we create edge to the points of absolute distance. For example, B is a core point. Then we connect B, to all the points within absolute distance to B, and those are the edges we created here. Next, we define N as a set of nodes in the graph, which are really all the data points in the data set. If no core points in the set N then we're done. If there are still core points in the set N we pick a random core point C. Then we find the connected components from C. For example, if we pick this point over here as the core point C, then the connected component can look like the following. Then we update the remaining set of points N by removing all the points in x from N. So, this operator indicate the set difference by removing x from N. Then we go back to step three and continue the iteration.

150 Now let's look at some examples of DBSCAN giving a set of two-dimensional data points. Look like this. We want to find a set of clusters using DBSCAN. The first step of DBSCAN will try to classify all those data points into this three categories. The core points are those gold color points. And the border points are those blue color points. And the noises are the orange color points. Then we iterate through

DB scan algorithms, we'll find this one, two, three, four five, six clusters. And you notice that there are points not belong to any of those clusters, they are the noises.

151 Here's a quiz for DBscan. So, how many cluster can a datapoint belong to, using DBscan algorithm? Put your answer in this orange box.

152 And the answer is 0 or 1. If the data point is a noise, it won't belong to any cluster, so it will be 0. If the data point is a core points or a border point, then it will belong to 1 cluster. So, the key here is all the noises are not assigned to any cluster.

153 So far we have talked about clustering algorithms. Next we introduce a set of evaluation metrics for clustering. In particular, we'll talk about rand index, mutual information, and silhouette coefficient. And rand index and mutual information requires we know the ground truths of the clustering result. And silhouette coefficient does not require any ground truths.

154 Now let's talk about Rand Index or RI. Given n data points, let's say X is the clustering assignment from the algorithm and y is the ground truth. In here, we illustrate a cluster X1 that comes from the algorithm and cluster Y1 come from the ground truth. Then we compute the term a, which is the number of pairs that belong to the same cluster in X and in Y. For example, these two data points, P1 and P2 belong to the same cluster in X because they both belong to X1. And also, they belong to the same cluster in Y because they both belong to Y1. And this pair will be counted towards this term. We want to find all such pairs that belong to the same cluster in both X and Y. Then we compute another term, b, corresponding to the number of pair that belong to different cluster in X and Y. For example, these two points P3 and P4. The P3 belong to X1, and P4 belong to Y1. They belong to different clusters. And we want to find all such pairs. So, once we know A and B, the Rand Index is defined as a plus b divided by the total number of pairs. In this case, the total number of pairs is n times n minus 1 divided by 2. The Rand Index is between 0 and 1. 0 means bad clustering assignment, and 1 means perfect clustering assignment. So, in general, we want to have an algorithm with high Rand Index.

155 Like grand index, mutual information is another way to measure clustering quality. And mutual information is a concept from information theory which measures the mutual dependency of two random variables. In this case, the two random variables are clustering assignment x and the ground truth assignment y. More specifically, X has K cluster, X1, X2 to XK and Y has R cluster, Y1, Y2 to Yr, and then the entropy of X is defined as sum of probability PX times log of PX. And this entropy term measures uncertainty of x. And the entropy term is between 0 and 1. And 0 means the variable is deterministic. And 1 means the variable is completely random. Then we define the mutual information between x and y. As sum over both x and y of the joint probability p (x, y) times log of p (x, y) divided by the marginal probability p (x) times marginal probability of p (y). And the range of mutual information is between 0 and entropy of x. So, when mutual information equals 0. Means x and y are independent. When mutual information equals H(x), then it means y is completely determined by x. If we apply mutual information as cluster and evaluation metric, then higher value means good clustering assignment. Sometimes, people want the metric to be normalized between 0 and 1. So in this case, we can define this normalized mutual information as the mutual information between x and y, divided by the square root of entropy of x times entropy of y.

156 There are a lot of commonality between Rand index and mutual information. So, what are the pros and cons of this Q evaluation metric. They both provided bounded ranges when the metric is close

to 0, which means bad clustering assignment. When it's close to one, means good cluster assignment. And there's no assumption of clustering shape, so you work with arbitrary cluster algorithms, but a disadvantage is it will require ground truth cluster assignment. In many cases we don't know the ground truth, even the data set, so this will be a big limitation for both rand index and mutual information.

157 Next let's introduce silhouette coefficient, which is another popular clustering evaluation metric. In this case, we do not require any ground truce information. Here's the main idea behind silhouette coefficient. Given the data point x, we first want to find the cluster containing x, then we want to find the next nearest cluster to x. So visually, it's illustrated as the following. So, we have these two clusters, C contains x and C prime is another cluster that's very close to x. Then we compute this measure a, which is average distance of x to all the data points in cluster C, which means we compute the distance of x to all the other data points in C and find the average. Similarly, we define the term b, which is average distance from x to all the other points in C prime. For example, in this case, we'll compute the distance from x to all the other points in C prime and then take the average. And the silhouette coefficient on x = b- a divided by max(a,b). The intuition is if the assignment of x is good, then the difference between b and a should be large, then we'll have a large value for this coefficient, which means good clustering assignment. If x is assigned to a cluster which is so close to another cluster, then the difference will be small. In that case, the silhouette coefficient will be small. In that case, it will be a bad clustering assignment.

158 In summary, here are the pros and cons for a silhouette coefficient. Again, it provided a bounded range between minus 1 and 1. And minus 1 means very bad clustering assignment, 1 means very good clustering assignment, and 0, in this case, means overlapping cluster. The limitation here is silhouette coefficient assumes spherical clusters. For the clustering algorithms that generated non spherical clusters, such as DBScan, silhouette coefficient would not be a good evaluation metric.

09 Spark

159 Previously, we have talked about MapReduce, a distributed fault tolerance system, for processing large data set. However, MapReduce, is not efficient for supporting iterative workload as many machine learning algorithm require. Today, we'll introduce Spark, another big data system that provides better performance, by using distributed memory, across many machines. We'll talk about the key concept behind Spark. Namely, Resilient Distributed Dataset or RDD. We'll explain how Spark can better support iterative algorithms. We also provide some example of house care applications using Spark.

160 Before we introduce the big data system Spark, let's first remind ourselves of the computing environment we're using here. For big data analytics, we usually need to perform all the analytics in a data center look like this. Many racks of servers that are interconnected through Internet. A lot of time we access this environment through cloud computing services such as, Amazon Web Services, Google Cloud Platform and Microsoft Azure. With this environment in mind, next we'll see why we need to design another big data system like Spark

161 In the previous lecture we have introduced Hadoop and MapReduce. Hadoop is a big data system that operates on acyclic data flow graphs from stable storage to stable storage. So, the input and output of Hadoop jobs are from the stable storage system. For example, hard disks in the distributed environment. The computing jobs follows the MapReduce paradigm, which forms acyclic data flow graph that look like this. And all the input and output from each Map and Reduce function are in this stable storage system. So, Hadoop system is based on acyclic data flow graph and stable storage to ensure fault tolerance. So, the benefit of such a design is that at one time we can decide where to run different tasks. So, the map and reduce function can be run on different machine in the distributed environment. And we can automatically recover from failures because the input and output of those MapReduce functions are stored in a stable storage. For example, disk. So now we know Hadoop can work with big data using MapReduce computation. So, what are the limitations of Hadoop? Hadoop often does not perform well when the workload involves cycles. More precisely, Hadoop is inefficient for application that repeatedly reuse a working set of data. So, what are those application that require such workload? There are some major applications such as iterative algorithms and interactive data mining tools that fall into this category. Iterative algorithm contains many machine learning algorithms, such as clustering, classification. They often times need to be repeatedly computed on the same data set. Then we have graph analysis. Many graph algorithms, such as page rank computation, spectral clustering, they also fall into this iterative algorithm category. Also, more and more data mining practice need interactive response. This days, data scientists using interactive tools such as R and Python to explore data, form hypotheses, and validate those hypotheses and repeat on the same data set. It will be great to provide more efficient big data platform that can support all this.

162 Next let's illustrate the inefficiency of Hadoop using one concrete example. Say we want to a machine learning model on this training set. And the model is specified by the model parameter w. Typically, a machine learning algorithm start with some initial model. Then they run the algorithm for one iteration. For example, this can be implemented as the map-reduce job. And the resulting model, W (1) is also going to be stored on the disk. Then we repeat this process with this new model parameter against the same training data to get the next iteration of model. And this process repeated many times until convergence. If you look at this computation pattern, you quickly realize that we repeatedly load the same data from disk to memory in order to perform the computation. At the same time, we have to

write the result, in this case, are those model parameter repeatedly to disk from iteration to iteration. And this repeated reading and writing to disk are the inefficiencies of Hadoop.

163 Next let's illustrate these two types of workload, iterative algorithm, and interactive computation. For iterative algorithm, we apply the same computation logic on the same input data set again and again to generate different iteration of result set. And those result are often corresponding to the model parameters. Interactive computation is slightly different. In this case, we iteratively perform some computation. Across the iteration, those computation can be very different. For example, we start with some raw data, then perform some data cleaning, get a cleaner data, then perform modeling algorithms to get the model. And the requirement is, we want this iteration to iteration to be fast so that we can perform this work in an interactive manner. So, the key objective for supporting iterative algorithms and interactive computation is to keep the working set in memory so that we can perform all those operations fast. So here we illustrate that ideas using this diagram. For iterative algorithm, what we want to do here is load the entire data set into a distributed memory across many machines. So that when we perform all of this iterative computation, all the data are in memory, so we don't have to read and write to disk. For the interactive computation, the idea is also very similar. We want to keep the intermediate result all in memory so that the next iteration can perform immediately once the first iteration is done. There is no read and write to disk anymore.

164 So what's the challenge about keeping everything in memory? The key challenge is how do we design a distributed memory abstraction that is both fault-tolerant and efficient. Hadoop guarantee fault-tolerance by using disk so that they can keep data and intermediate result in a stable storage. But memory is not a stable storage. It's, by definition, efficient, but it's not fault-tolerant. To guarantee both fault-tolerance and efficiency is the challenge for such a system. Next, let's look at some options here. A lot of existing distributed storage systems depends on fine-grained updates. For example, given the table look like this, we can re-write from any place in this table. So those dots indicate a place we want to read and write a specific cell from that table. Example of such abstraction include database systems, key-value stores, and distributed memory. So, the way to make this type of storage abstraction fault-tolerance is we have to replicate data like what we have done in the Hadoop setting where we replicate data several times and store on different machines. Or we have to keep track what operation has happened using logs. But if we have this very fine-grained updates any place on this table, then keeping track of what has been changed or replicating all those data become very expensive.

165 To solve this problem, spark decides to strike a balance between granularity of the computation and the efficiency for enabling fall tolerance. In this case, RDD provide an interface based on coarse-grained transformations of the entire data set. For example, a map function can be performed on every record in this dataset. Similarly for group-by or join or filter. So, this operation are course-grained operations because it's not focusing on a specific part of the dataset. They are being applied on the entire dataset. If we only have to keep track of the course-grained transformations, all operations can be efficiently tracked. In this case, efficient fault recovery can be done using lineage. So, lineage here looks like a family tree. It keeps track of all the transformation across different RDD. It may start with the root RDD, and some transformation being applied to those RDD and derived RDD are generated. So, if we only support coarse-grained transformation, we can log those operations that apply to many elements in this RDD. And we can re compute the failure happened because we have the operation being logged. And more importantly, since everything is in memory, there's no cause if nothing fails. So, this is very efficient mechanism to enable fault tolerance.

166 Next let's illustrate how RDD can recover from failure. Again, we have these two different scenarios. This is for iterative algorithms, and this is for interactive computation. It's quite obvious for the interactive computation. For example, the result from the second iteration failed. In this case, we only need to repeat the second iteration to regenerate the same result. If both results are failed in memory, we can repeat iteration one and two to recompute the result. For iterative algorithms, if only part of data has failed in memory, we can load the corresponding chunk from the stable storage to refresh that memory. Also, those results from each iteration are stored in memory. So, if part of that is filled, like the latest one, then we only need to repeat that from the previous one to regenerate the latest one. And this is the key idea behind spark, so spark is really a big data systems built on top of RDD.

167 Software stack for Spark already become quite rich. The Spark core contains the basic functionality of Spark including components for scheduling, memory management, fault recovery, and interacting with storage systems and more. And Spark Core is a home API for RDD and this RDD, as we just illustrated, is the main programming abstraction. And RDD are represented as a collection of data points distributed across many computer nodes and can be manipulated in parallel. Spark Core provides many APIs for building and manipulating this RDD. Spark SQL is Spark package for working with structure data. It allows querying data via SQL like syntax. Spark streaming is a spark component that enable processing real-time data such as weblogs. Spark also has a machine learning library called MLib. MLib provide many machine learning algorithms including classification, regression, clustering, collaborative futuring, and so on. All this machine learning algorithm in MLib are designed to scale to a cluster of computers. GraphX is a graph processing engine for Spark. It can manipulate large graph such as social networks of friends, citation network of papers, publications, and patient and disease graph as well as all those medical ontologies. GraphX provide various operators on graphs, such as extracting sub graphs and map vertices. It also provide a library of common graph algorithms such as PageRank. Under the hood, Spark is designed to efficiently scale up from one machine to many machines. To achieve this flexibility, Spark can run over a variety of cluster managers such as Standalone Scheduler on a single machine, and Hadoop YARN, and Apache Mesos.

168 Next let's talk about programming interface for Spark. The core interface is written in Scala. It has several other different languages being supported such as Python and Java. There are three different types of API. There's one set of API is about creating and manipulating RDD. Then we have these different operations on RDD, such as transformation of one RDD to another, an action that has to operate on an RDD in order to compute some results. Spark also provide a way to share global variables across machines through this restricted share variable mechanism, such as broadcast and accumulator.

169 Next let's look at some example of RDD transformation. Here's example illustrate map versus flatMap, given an RDD look like this we have three different type of symptoms, sprained ankles, fractured ankle, and severe sprained knee. Then we want to apply this map function with this particular operation tokenize. For example, when we tokenize this strain we'll get the list of words. For example, the input is "sprained ankle" then output is two words, "sprained" and "ankle". And that's a list so if we applied this with the map function we'll have a set of list one for each element here. For example, sprained ankle ill become a list of two word, sprained and ankle. And fractured ankle will be another list, fractured and ankle. And the severe sprained knee becomes a list of three words severe, sprained, knee. So, the result of map function become a list of lists. In many cases, we don't want this two-level structure. We want one level list with all those words in the same list. So, in that case we can apply flatMap. The result of flatMap is a list of all those individual words. So essentially we're flattening the RDD from map function

into this one level list or another way to say that is, we concatenate all those lists together. Make them a single list. Now some more examples of RDD transformation. Now, given an RDD of words, if we apply distinct transformation, the result of that will be the distinct word in the original RDD. So sprained only show up once in the resulting RDD. The result on this operation is relatively cheap. Another commonly used operation is union. So, union works with two RDDs. So now we have this RDD1, with the set of words. RDD2 with another set of words. Now, if we apply union, so RDD1.union(RDD2) So this will give us a resulting RDD that merged this two RDD together. In this operation is also relatively cheap because it only involves these two already together. Notice that this is really a simple concatenate of two RDD together. The redundancy across this RDD still remains in this resulting RDD. Other popular operations, such as intersection, is also supported. Now given these two RDDs, we want to find intersection between these two, and that will be the sprain and knee because they are in both RDD. This operation is more expensive because it involve sorting, refining out distinct values and finding out overlapping across two RDD. So, this is more expensive to do another transformation such as subtract. That's also a set operation. Given two RDDs, we want to remove the elements in RDD2 from RDD1. So RDD1.subtract(RDD2). So, this gives us the resulting RDD, which is the remaining element in RDD1 that are not in RDD2. This operation can be expensive because we have to find the distinct elements in both RDDs, then perform a set difference operation.

171 Now let's do a crisp on RDD transformation. So, what is the output of each given transformation for this input RDD. This is the map function applying to x and return x times x. Then will have this filter function apply to x which was x equal to one. So, write your result in this red boxes.

172 When we apply mapped function, this is the result. 1, 4, 9, 16. When we apply filtered function, in this case, we only return the value when they are not equal to one. We have value 2, 3 and 4.

173 Spark provides many operations that categorize into these two groups, transformations, which define a new RDD from an input RDD, and actions, that return a result to the driver program. For example, for transformations, we talked about map, filter, and there are many more, such as sample, groupByKey, reduceByKey, sortByKey, flatMap, union, join, cogroup, Cross, mapValues. So, it's a very rich set up of transformations, way beyond what MapReduce provide us in Hadoop. Same for Actions. Action is like the reduce phase in MapReduce. So, it can collect the values from RDD. We can perform a reduce function, we can count how many records, we can save it somewhere else, we can look up by key to find a subset. These are example set of transformation and actions, but there are many more operations for Spark, and more examples will be in the instructor notes.

174 Another important concept in Spark is this shared variable. The way to create a shared variable is to use this broadcast variable that allows us to efficiently send a large, read-only values to all the worker nodes. Next, let's illustrate a Spark job using this example. Say we have a large volume of clinical nodes. We want to find certain patterns. So, Spark job runs on a clusters. You have two sets of nodes. The driver which is like MapReduce master. And workers, which just like MapReduce slaves. The drivers coordinate the entire task, and the workers perform all those individual computation. Here are some example of a Spark code. First step, we want to load this clinical node from hdfs. So, we load all the lines in this clinical nodes. So that's our base RDD. Next we want to filter all those lines to find the line start with keyword SYMPTOM. This will become the transformed RDD. Then we want to split the symptom lines with deliminator tab. Then find the third element with this parameter 2. For example, in this case we'll assume the third element corresponding to the symptom name. So that's what we want. Since we know we're

going to find various patterns on those symptom names, so we cache this result. So far everything runs on driver and no computation has really happened, until we reach an action. Here is an action. We want to filter all these symptom names to find the keyword, fever, and we want to count those lines. So, we want to know how many times symptom fever occurs in this case. So, this is an action, and they will happen. When we reach this action line, the driver will send those tasks to individual workers to work on their own block. Then the worker will go through the steps to try to find fevers and count the number of times they occurred. And the result will be sent back to the driver node, and that's the total count for fever. At the same times, the cache will be created on all those worker nodes. So next time we don't have to do all the beginning operations, because the result are cached, so the next line is another action. Here we'll look for a different symptom, cough, and we'll want to count how many times they occur in the clinical nodes. In this case we send the task, again, to the workers, and they already have the cache. So, they don't have to go through the raw data again. So, what they need to do is continue from this cached line to compute the number of times cough occurred. And the result will be returned back to the driver and the process continued. So, this is an illustration of how Spark job works. Spark can give quite amazing results because this in-memory operation. For example, full-text search on Wikipedia takes less than 1 second versus 20 seconds if it's read from disk. With Spark, we can scale to 1 TB data in 5-7 seconds versus 170 seconds just to read from disk. So, the reason we can get this 5-7 second near real time response on a huge data set is because now we can read and processing data in parallel. And also cache the result in memory whenever needed.

175 Now, let's illustrate how fault tolerance is enabled using RDD. So, RDD tracks that lineage information of all those different transformations, and they can recompute efficiently if lost partition happen. In the previous example, we have seen that this sequence of transformation help us to find the symptom names. So, we first filter to find the line contain symptom then split and extract the individual elements which is the symptom name. So, visually, what happens is we're first read a file from HDFS, perform filter operation, then we have this filtered RDD. Then, the filtered RDD will be the input to the map function to generate the mapped RDD. For example, if part of the result from the filtered RDD is lost, we can recompute very quickly from the previous step. By the way, all those transformation are tracked because they're the lineage information for us to reconstruct the RDDs we need if lost partition happen.

176 With Spark, now we can efficiently support iterative algorithms such as machine learning algorithm like logistic regression. So, giving logistic regression as a classification algorithm, trying to find the best line to separate these two sets of points. And this is the target we want to learn. We start with this random initial line. First, we compute a greeting over all the data points, then we update this initial line by moving towards the greeting. Then we recompute the greeting again on all the data points, update the line, and do this update again, again, and again, until it converges. That's how we get the final target line. Because the input and output are all capped in memory. This update can be done very quickly. Here's example of spark code for writing logistic regression. We start by loading the data from disk into memory and cache that. Then we initialize the model parameters w. Then we perform the following iterations. Then we complete the gradient over the entire data set, which is really involve a map function over all the data points. And for each data points, we perform this calculation, then perform the reduce function. This is really to sum them up, that give us the gradient. Notice that with this one simple operation, we performed a MapReduce function to compute the gradient. Once we have the gradient, we can update the motto parameters then repeat this process. Finally, we have the final parameters once it's done the all the iterations. Here is a performance comparison between Hadoop and Spark for computing logistic

regression. So, X axis indicates the number of iterations from first iteration to 30th iteration. So, Y axis is total running time. And in this case, the lower the better. Notice that every iteration of Hadoop takes about 127 second. Notice that for Spark the first iteration is even longer than Hadoop because the caching operation. Then the further iterations for Spark only take six seconds as opposed to over 100 seconds. That's why Spark's in total running time is much lower than Hadoop.

177 Next let's illustrate Spark with another house care example. Say we have matrix R which is the patient by disease. Every row corresponding to a patient, every column corresponding to a disease. And the goal here is to predict the disease risk based on existing disease diagnosis. For example, we know for a patient what disease they already have, but for the ones they don't have, we want to assess the risk. So those are all those question marks. So how do we model that problem? This is really a collaborative filtering problems. We can model a large matrix R with many missing values with a product of two matrices, A and B. A corresponding to all the patient features, and the B corresponding to all the disease features. So, the way to do that is through this alternating least squares operation, ALS. So, we'll fix one of this matrix. For example, we'll fix B then update A, then we can fix A update B. So, this is called alternating least squares. So, the algorithm goes as follows, we start with a random initialization of A and B. They'll start to optimize the patient feature vectors, A, based on the disease feature vector, B. Then we do that for the disease feature vectors based on the patient feature vectors. Then we repeat this process until convergence. So, if we want to write this using Spark, it's actually quite easy to do.

178 Next let's talk about how do we do this using Spark. So, we can right this simple 0 implementation of alternating lee square. First, we can read the data matrix R. Then, we can initialize A and B randomly. After that, we can start this iterative operation. Here we want to update A and B alternatively. For example, we have U patients, for each patient I will update that corresponding patient based on the matrix B and input matrix R. Say we have a function for doing that. This is actually quite straight forward because it's just simple B square problem can be solved with a linear regression operation. That is just to update for patient I. We want to update A and B alternatively. So, these are the range values. So, for A we update each patient from 0 to U minus 1, and for B we update each disease from 0 to M minus 1. So, for each patient we want to update the patient based on the input matrix, R. And the disease matrix speed. So, this really becomes really a regression problems. And we want to do this for every patient. So that's where this map function comes in. For each patient I, from zero to U minus 1 we apply this function. Similarly, we can do that for disease. For each disease, we update the disease based on input data matrix R, and patient matrix A. So next, let's see how we can do this in parallel.

179 Again, we read the input data, and we initialize A and B, then here, we can run all of those update for each patient in parallel. Then, collect those result to the driver node. So, spark.parallelize provide the way for us to perform this parallel computation. And similarly, we can perform this parallelized operation on diseases. For each disease, we want to update that as well. And the problem here is the big data matrix R has to be sent to all the node in each iteration. This is very inefficient if we have a large data matrix. So, this is where we use broadcast. Instead of reading data directly, we can use spark.broadcast to create this read only object. Then when we want to use this matrix, we can do R.value to access that particular variable. It has been shown that we can achieve three times performance improvement by simply doing broadcast.

10 Medical Ontology

180 One of the things that make healthcare a unique domain for big data analytics is the existence of structured medical knowledge, which are often represented as ontologies or knowledge graphs. For historical reason, healthcare and medicine domain have already developed many ontologies for organizing diseases, medical procedures, medications, lab tests, and more. These ontologies give us great resources to understand house care data and to enhance and validate all of the models developed using big data analytics tools. In this lesson we discuss several just ontologies and illustrate how they can help us in our analysis.

181 Now let's talk about health data standards. There are many different health data standards. In this lecture, we'll present several important ones. Each has its own unique focus. Let's illustrate all those data standards through an example of patient encounter. For example, this patient is coming to hospital to get a lab test, and the result of the lab test is stored using LOINC, or Logical Observation Identifiers Names and Codes. LOINC typically represents all different kinds of lab tests. So, the lab test result goes to the doctor, and the doctor diagnoses patient with different disease code, or ICD. It stands for International Classification of Disease. That represents different diagnoses and different diseases. Once we have the diagnosis on this given patient, we may want to treat this patient with a medical procedure that's represented by CPT code, or Current Procedural Terminology. So, CPT represents all different procedures that can happen, to give an individual. Of course, the patient can also take some medication, and that's represented by NDC code, or National Drug Code. So, for a given patient, during a typical medical encounter, all different types of information are coded with different health data standards. LOINC code for labs, ICD code for diagnosis, CPT code for procedures, and NDC code for medications, and all those codes interacting with each other can be represented by a medical ontology. The most popular medical ontology is called SNOMED, it stands for Systemized Nomenclature of Medicine. It's a huge graph of medical concepts and their relations. Of course, to utilize all this information in a huge oncology like SNOMED, you need software systems to interact with all those concepts and relations. The most popular software for accessing this medical knowledge is UMLS. It stands for Unified Medical Language System. UMLS is a set of software tools that provide integration of multiple sources of medical knowledge and medical ontologies. So next we'll talk about all these different health data standards. We'll first cover all those individual set of data standards for different types of medical data. Then we'll talk about SNOMED and UMLS as a way to integrate all those different medical concepts together. All these health data standards are very important to support common healthcare operations, such as efficient insurance claim processing. For example, when a healthcare encounter has happened and all this information has been recorded through electronic health records, and that information will be sent to an insurance company in order to process those claims so that the doctor can get paid. And all those health data standards support efficient insurance processing. The secondary use of these health data standards are to support research and development. Because house care's data are encoded largely in structured forms, so that it can be easily analyzed and standardized. Next we'll introduce all of them in more details.

182 First let's talk about ICD code for diagnosis. ICD stands for International Classification of Diseases which was developed by World Health Organization, and the focus of ICD codes is to categorize diseases. In US currently, we're using the version nine of ICD and we're already moving towards ICD-10, and most of the rest of the world are currently using version 10. And version-9 of ICD code has over 17,000 individual codes. It covers both diagnosis and procedures. And ICD-10 code is the next generation of ICD

code. It has over 141,000 unique code. Next let's talk about ICD-9 and ICD-10 in more details. First, let's introduce ICD-9 code. ICD-9 code has three to five digit with three different levels. Namely, the categories which cover the first three digit, the subcategories which is the fourth digit. And the subclassification, which correspond to the last digit. There are 17 categories plus several supplementary categories in ICD-9. The 17 categories correspond to major disease categories, such as infectious disease, neoplasia, and et cetera. ICD-9 code in the 17 categories have only numerical digits. For example, 250 is a category code for diabetes. And 250.01 is the subclassification for type one diabetes without complications. In addition to the 17 categories, there are supplementary categories starting with letter E or letter V. For example, V85, corresponding to body mass index. And V85.0 indicate BMI less than 19. V85.1 indicate BMI between 19 and 24. And V85.2 indicate BMI between 25 and 29. Now let's talk about ICD-10 code. ICD-10 code can have seven alphanumerical characters. So, it's longer than ICD-9 code, which only have five digits. First three characters indicate the disease category. The fourth character indicates Etiology of the disease. That is the cause of the disease. The fifth character indicate body part affected. And the six characters indicate the severity of the eonis. And the character seven is the place holder for extension of the code to increase specificity. For example, E110, indicate disease category with type one diabetes. And nine indicate there's no complication. Here's an example of a more complicated ICD-10 code. M1A indicates the disease, chronic gout disease. And a three indicate the Etiology of renal impairment. And one indicate the body part, in this case is the shoulder. And two indicate the severity or vital details, so in this case two means the left shoulder. And the last one, zero for extension. Here zero means without tophus. So, as you can see ICD-10 code can be quite complicated.

183 Because of the practical need, there are a huge effort involved in mapping from ICD-9 code to ICD-10 code. In general, a mapping from ICD-9 code to ICD-10 code is a one-to-many relationship, because ICD-10 code is just more specific than ICD-9. So, in some cases, ICD-9 is already pretty specific. In this case, they will have one-to-one mapping. For example, for this Tietze's Syndrome, they are both having unique code in ICD-9 and ICD-10. So, the mapping is one-to-one. But in most cases, we'll see one-to-many mappings. For example, 649.51, spotting complications during pregnancy, maps to three codes in ICD-10. So, in this case, complications during each trimester has its separate code in ICD-10. Sometimes the mapping from ICD-9 to ICD-10 can be quite complicated. For example, 733.82 in ICD-9 has a mapping of 2,530 in ICD-10. So, ICD 10 is just going through a lot more details about the disease. So, the mapping can be quite tricky.

184 Here's a quiz question. Which of the following are not an ICD-9 code? Is this 501 or U80.1, 802.3, V70, E820.0, and 5A0.01?

185 Let's go through them, one by one. ICD-9 code is between three and five digits. So, 501 is valid. And U80.1 is not valid, as the initial letter can only be E or V. And 802.3 is valid. And V70 and E820.0 are both valid because they corresponding to valid supplementary code. It start with V or E. And 5A0.01 is not a valid ICD-9 code because ICD-9 cannot have letter after the first position. So, the answers are U80.1 and 5A0.01. These two are not valid ICD-9 code.

186 Next let's do another quiz. Please search online to find the corresponding ICD-9 and ICD-10 codes for Influenza.

187 Here's the answer for ICD-9, 487.1 stands for Influenza. And for ICD-10, the answer is J11.1.

188 CPT code is another important coding system used in US healthcare. CPT stands for Current Procedure Terminology. The focus of CPT is to describe medical, surgical, and diagnostic services. It's a US standard for coding medical procedures. CPT is maintained by American Medical Association. And CPT is mainly used by insurance company to determine how much to pay for a medical service. In general, reimbursement rate will be associated with each CPT code has been used in the claims. So, CPT is very important in the US, since it tied to how much money doctor will make. Now let's talk about CPT code in more details. CPT is a five-digit code, has three different categories. Category one, corresponding to widely performed procedures. And Category two, corresponding to quality metrics performed in healthcare organization. Then we have Category three, it's again four digits, follows with letter T for experimental use. Now let's talk about Category one, CPT code. We can take quick look at the different sections of Category one CPT code. They're arranged in their numerical ranges. They are divided into six sections, Evaluation and Management, Anesthesia, Surgery, Radiology, Pathology and Laboratory, Medicine. And here are the Category two CPT code. There are supplementary code for tracking the performance measure. And the different ranges of the code map to different types of services. It include, Composite measures, Patient management, Patient history, Physical examination, Diagnostic/screening process and result, Therapeutic and preventive or other interventions, Follow up or other outcome, Patient safety and Structural measures. For example, blood pressure measured is one of the Composite measures, and there are other Composite measures, as well, they're all in this range.

189 Here's a quiz on CPT code. Search online, try to find a CPT code for detailed office visit.

190 The office visit is in the range of 99201 to 99205. And all those different codes represent an office visit. However, the distinction is mainly on how much time you spent face to face with the patient. So, 99201 is for ten minutes time, while 99205 corresponding to an hour time. So, you can see that which code you use to document the service will make a big difference. However, the underlying service can be very similar.

191 Next let's talk about LOINC. LOINC is a standard for lab and clinical observation, and it's created by Regenstrief Institute which is a non-profit organization in Indiana. LOINC has been created mainly to capturing lab test. And a LOINC code contains digits like 2865-4, and this is a LOINC code. Of course, LOINC code itself has a number and also has attributes associated with this lab test. And different attributes are separated by a column. Let us talk about loinc attribute and loinc number. Here's the attribute of a specific loinc code. They're separated by colon. The first part is the component name, and this specify the specific lab test. The second part is the property of the lab measurement. And the third part is a time aspect whether it's measured at a point of time or over some durations specified here. Here Pt means point of time. The fourth part is a type of sample. For example, serum or plasma in this example. Next is the scale. The scale may be quantitative, ordinal, or nominal or narrative. Finally, we have the method, that has been used to conduct this lab test. And this particular attribute corresponding to a LOINC number of 2865-4.

192 Here's the quiz on LOINC code. Try to search online to find LOINC code for lab test Creatinine.

193 And the answer is 2160-0. You may find Azure lab tests also contains the term creatinine, and this is one of the most popular one, a test for creatinine.

194 Next let's talk about NDC code. NDC stands for National Drug Code. NDC is the standard for medications, and NDC is registered and maintained by FDA, the Food and Drug Administration. And FDA

maintains a searchable database of all the NDC code on their website. NDC codes are used throughout the entire drug supply chain, from pharmaceutical company to drug distribution companies, to medical community, and to insurance company, and government. They all use NDC code to track medications. NDC have three parts. The first part is the four to five digits that indicates the labeler that is a company produce the drug. 0777 is a labeler code for Dista Products, and the second part is the product code to indicate what drug it is. For example, 3105 corresponds to Prozac Capsules of 20mg. And then the final one or two digit corresponding to the package code. In this case, 02 indicate there are hundred pills in this package. NDC code exists in different ways of grouping those three segments. It has a four digit here as labeler, four digit here as the product code, and two digit as a package code. Or it could have a five-digit labeler, three-digit product code and two-digit package. Or it could have five-digit labeler, four-digit product code and one digit package code. But overall, all NDC code have ten digits.

195 Here's a quiz question on NDC code. Search online, find the NDC code for metformin hydrochloride, 500 milligram.

196 One of the answer could be, 0093-7214-01. Your answer could be different from this depending on who is the labeler, and what's the product code and this can look different, even for the same drug.

197 Next let's talk about SNOMED. SNOMED is one of the most comprehensive multilingual medical ontology that describes different clinical and healthcare terminologies, and their relationships. And SNOMED stands for systematized nomenclature of medicine. And SNOMED is maintained by another non-profit standard organization called IHT SDO, which is based in Denmark. And the objective of SNOMED is to encode all kinds of health information, and to support effective clinical recording of data was the aim of improving patient care. Next, let's look into more details of SNOMED. First, let's talk about SNOMED Development Cycle. It starts from the central organization, IHTSDO. It maintains the international version of SNOMED ontology. More specifically, it maintains the SNOMED's development, release, distribution, maintenance, and education. And this organization released the SNOMED CT international which is the international version of SNOMED. Then there are different members, those are different countries. Each country can have their own national release. For example, US can have their own SNOMED version. Those different versions are called reference set. There could be SNOMED CT US National Edition and released in 2005. That could be one reference set. Then there's different implementation which may only cover a subset of the entire reference set within a country. Then once we have the implementation of SNOMED. And the users of SNOMED are quite broad. They could be clinicians, researchers, or data analysts. And the purpose of SNOMED is to help improve clinical documentation and understand semantic interoperability of medical concepts, and to enable clinical decision support, as well as data retrieval, analytics, statistics, information management purpose.

198 Now let’s talk about the Logical Model of Snomed CT. The logical model of Snomed CT is quite simple. It asks three types of components, concepts, descriptions about concepts, and the relationship between concepts. Every concept has a unique identifier, our SNOMED CT identifier. This is a machine-readable identifier. Then each concept can be associated with one or more descriptions. And the descriptions provide human readable forms of the concept. There are two types of descriptions. One is the fully specified names, FSN. That is the most precise explanation of that concept. And there are also other synonym that provide different version, or different ways of describing the same concept. And relationship captures interactions between multiple concepts. Usually two concepts. For example, the most important relationship is the is a relationship, or subtype relationship. It gives you a way to

generalize a concept from more specific level to more general level. Then there's the attribute relationship. Each concept can have multiple different attribute.

199 Now let's see an example of SNOMED concept. The concept ID is the following, 22298006. It is definitely in a machine-readable form. This concept ID is associated with different descriptions, and among which myocardial infarction disorder is a fully specified name for this concept. Then there are many synonyms include myocardial infarction, infarction of the heart, MI, heart attack, and so on. Some of those description are considered preferred. For example, myocardial infarction disorder and myocardial infarction. And some are considered acceptable. So, the rest of those descriptions are considered acceptable. So, note that the concept of preferred or acceptable, they're implemented in the US English Language Reference Set, which may not be in the reference set from other countries.

200 Now let's talk about IS a relationship. IS a relationship indicate generalization from a specific concept to a more general concept. For example, cellulitis of foot is a cellulitis, so this is a IS a relationship. At the same times, cellulitis of foot is also a disorder of foot. So, there's two different paths to generalize this concept. And you notice that IS a relationship is directional. So, two concept are directly linked by IS a relationship. The source concept is said to be the subtype. And the destination concept is said to be the supertype. If we generalize all those concepts to the most general forms, we have a single root of the entire hierarchy. Besides IS a relationship, there are other relationship as well. For example, abscesses of heart Is associated morphology to abscesses. And it has a finding site to heart structure.

201 Here's a summary of SNOMED. SNOMED Ontology consist of different type of concepts, and those concepts are organized in a hierarchy. For example, from top down, there is 19 level of hierarchy, start from body structure go down to the substance. For example, arthritis of knee is the is arthropathy of knee joint is an arthropathy, is a joint finding, and so on, so you can see all those different levels of concepts can be mapped through the hierarchy of the entire SNOMED concept. In the top level are the most general or low granularity concept, and the lowest level here are the high granular or the most specific concepts. Concepts and hierarchies, we also have relationship and their attribute. For example, here are two different relationships. Arthropathy is a joint finding, and another one Appendicitis is associated morphology to inflammation. So here associated morphology is an attribute in this relation, and every concept has a unique machine-readable identifier, and each concept also has associated descriptions. One of those is fully specified name.

202 Here's the quiz on SNOMED code. Search online, try to find a SNOMED code for chronic gouty arthritis.

203 And the answer is 68451005.

204 Here's another quiz. What is the resulting structure of all the ISA relationship, in SNOMED? Is this an undirected graph? Is this a tree? Is it directed graph without cycles? Or is it undirected graph with cycles?

205 And the answer is C, it's a directed graph without cycles. It's a directed graph because all those ISA relationship has direction, so it's a directed graph. The reason it's a directed graph without cycle is because every relationship has direction, from the subtype to the super type. And given a concept, it can have multiple parents or multiple super types, so it's not a tree. And there's no cycle in this graph because it always start from specific concept to more general concept. It won't come back, there's no cycle in this graph.

206 Next let's talk about UMLS. UMLS stands for Unified Medical Language System. It is a set of software tools that maintained by National Library of Medicine in the US. It's a comprehensive thesaurus and ontology of all biomedical concepts. It integrates all those existing data standards we just talked about. It also provides software tools to map data to those clinical concepts. So, what are the different components in UMLS? So UMLS recognizes that there are many existing ontologies and terminologies. They want to integrate all of them together, you see one system, so that people can access all of those medical concepts through the systems. There are three knowledge sources in UMLS, the metathesaurus, semantic network, and specialist, lexicon, and tools. There's over 1 million biomedical concepts from over 100 different sources that constitute this medical storage. It includes all the ones we have talked about such as ICD and Snomed. It covers 135 broad categories and 54 different type of relationship between all those concepts, and the semantic type and relationship provide a consistent categorization of concept and their relationship represented in UMLS metathesaurus. Third is the lexicon information and tools that can help processing medical texts. Next, let's provide more details on each one of them.

207 Metathesaurus Concepts. The idea is very similar to SNOMED. Each concept has a specific identifier and organizes into a hierarchy. At the lowest level we have atom, at over 7.4 million atoms or AUI. Those are concept name in a specific source. For example, all this different AUIs mapped to something related to headache. Then there are strings, they're are distinct concept names. So, you notice that on the atom level even the same mention can have different AUI because they come from different ontology, different sources. One from MeSH, one from ICD-10. But they have the same string, so string or SUI will be the same. Then we have terms, that's a set of normalized names. For example, here, headache and headaches all map to the same term, or LUI. Then on the highest level, we have concept, or CUI. That's a set of synonyms. And all of this together, for example, correspond to a single CUI, or cooey.

208 Now let's talk about Semantic Network. So, there are over a 135 semantic types such as, disease, syndromes, clinical drugs, all those are different semantic types. Semantic Network organize all those different types into And then organize those into hierarchies. And there are 54 different type of semantic relationships, such as cause, treat, all those are different type of relationship. And combine them together, the semantic types plus the semantic relationship. That give us the semantic network, and the concept of semantic network in UMLS is very similar to SNOWMED. The only difference here is here we have much richer information because multiple data sources, a different thesaurus, has to be integrated into UMLS, into the same semantic network.

209 Finally, we have the specialist lexicon, which is English language lexicon of common words and biomedical terms. So, we have over 300,000 biomedical terms, and their syntax, how the words are put together, morphology, such as inflection, derivation, compounding, all those language expressions, and orthography, that is really the spelling of those terms. And lexicon is used with lexicon tools in a variety of ways for natural language processing For example, MMTX and MetaMap are two software tools provided as part of UMIS to parse medical text

11 Graph Analysis

210 In this lesson we'll talk about graph analysis. Graph analysis is a set of methods, that are commonly used in search engines, and social networks. And in fact, we'll start by describing graph analysis examples, in search engine tasks. We describe some core algorithms, used likely by search engines. However, just as graph analysis can find a cluster of web pages, that are related to one another, it can also find a cluster of patients, or conditions, that are related to one another. We turn to house care applications of graph analysis, while discussing similarity graph, and spectral clustering.

211 Today we'll talk about two important graph-based algorithms. First, we'll talk about PageRank. Given a large directive graph, PageRank ranks all the node on this graph based on their importance. Second, we'll talk about spectral clustering, which is clustering algorithm based on graph partitioning. Let's start with PageRank.

212 PageRank is an algorithm that was originally developed by Google's co-founder to rank webpages. Traditional way to assess the importance of webpages is based on the content. However, content-based analysis is very susceptible to spend. And Google's co-founders, Larry, and Sergey, figured out a very smart way to rank webpages based on the link structures between the pages, instead of using the content in the page. For example, even this small directive graph, we can rank those four pages based on the link structure instead of the content inside those web pages. And the intuition behind PageRank is if more high-quality page link to you, then you're consider higher ranked. Now let's illustrate the effect of PageRank using this example. PageRank operates on directed network, for example, given a directed graph like this, we can run PageRank algorithms to figure out what nodes are important and what nodes are not important. For example, those red nodes are considered important or higher ranked because there are many incoming edges pointing to them. And those smaller nodes are considered less important because they don't have many node connect directly to them. And this is intuition behind PageRank algorithm. Next, let's see how can we formulate this intuition into a mathematical algorithm. Now let's come back to this toy example. In order to describe PageRank algorithm, we have to represent graph as a matrix. For example, this small graph will be converted to adjacency matrix to look like this. Every row represents a source, and every column represents a destination. For example, for page Google, there are three outgoing links, and you can see in the adjacency matrix, we have three entries with value one. And there are two outgoing links from Wikipedia, and we have two entries here corresponding to those two edges. And similarly, we can construct the rows for YouTube and Facebook that completes the adjacency matrix. And we call this matrix, A. Once we have the adjacency matrix, we can further normalize each row to make them sum to one. So, every non-zero element will be corresponding to a probability of jumping from the source page to the destination page. Once we have the normalized adjacency matrix A, PageRank can be described with this simple recursion. In this recursion, the q vector corresponding to all of the PageRank, and the PageRank can be computed as the sum of these two parts, browsing and teleporting. For the browsing part, we just redistribute the old PageRank using the source destination adjacency matrix. C is the weight assigned to the browsing part, and c is a value between 0 and 1. For example, we can assign c equal to 0.85 meaning that 85% of the weight will be given to the browsing, the rest will be given to the teleporting part. The teleporting part is just randomly jump to a page. In this case, e is all one vector and N is the number of node in the entire graph. This is mathematical definition of PageRank. Next, let's see how we can implement this efficiently using big data systems, such as MapReduce in Hadoop.

213 In order to implement PageRank using MapReduce, we have to partition the computation into map phase and reduce phase. So, in the map phase, we'll distribute the existing PageRank from the source to destination, following the outgoing links. Here's a pseudocode for the map function. The input is a key value pair, where key is a webpage, and the value is the current PageRank for this page qx and outgoing links y1 to ym. And output is another set of key value pairs where key is another page and value is the partial sum of the PageRank. And here's the algorithm. We'll first emit the page with a valid 0 to make sure all the pages are emitted. Then we follow the outgoing links. For each outgoing links we'll emit that corresponding page, yi and distribute a portion of existing PageRank to that page. In this particular case it will be qx/m, where m is the number of outgoing links from page x. Next, let's illustrate this map function using an example over here. In this case, we're working with the same graph. We just reassigned the ID to each page, 1, 2, 3, 4. So, when we run the map function for each page it has its current page rank, q1, q2, q3, and q4. Now let's look at page 1 as an example. So, we know page one has three outgoing links. You can see the PageRank for q1 will be equally divided into three parts and assigned to the outgoing pages. Of course, we also emit the page itself with 0 value, and we do the same for the second page. If we look at the second page as an example, it has the two outgoing links. So, we partition the existing PageRank for a second page equally and assign to those two pages. And we do the same for the third page and the fourth page. This illustrates the map phase of the algorithm for PageRank. Now let's illustrate the reduce phase of PageRank algorithm. Here's the pseudocode for the reduce function. The input is a key value pair, and the key is a page x, and the value is a list of partial sum of the PageRank for x. The output is another key value pair, and the key is the page x and the value is the updated PageRank for x. The algorithm is quite simple, we neutralize the page rank to be 0. We sum up the partial value from the value list. That give us the PageRank from the browsing part. Next, we re-normalize this to take into account of the teleporting parts. Finally, we emit this page x and its updated PageRank, qx. So, what's happening here is the Hadoop system will perform the shuffling operation by grouping all the partial sum for each page together to get this list of partial sum of PageRank. Then the reduce function will be applying to the list of partial sums computer updated PageRank. Now we understand the map and reduce function. To compute the final page rank we have to iteratively running this map reduce job many times in order to compute the final page rank.

214 Here's the quiz for PageRank. Give a directed graph look like this, mentally compute PageRank, and rank the following sites from highest to the lowest based on PageRank. So, in the events of tie, you can assign both side with the same number.

215 Let's figure out the answer together. The top ranked page is YouTube because it has three incoming links. Likewise, you can see Google ranked the last because there's no incoming links to this page, so it ranks number six. And similarly, Twitter ranked the number fifth, second to the last because it only has one incoming link. So, the rest of the pages, Wikipedia, Amazon, and Facebook, they all have two incoming links. You may think they will have the same PageRank, but in fact they don't. If you actually carry out the recursive algorithm PageRank does, you will realize Amazon has a higher rank than Wikipedia, than Facebook. So, because of the recursive nature of the algorithm, even the page with very similar local structure still has a very different ranking using PageRank.

216 Next let's talk about spectral clustering. In traditional clustering algorithms, given the input data and matrix, for example this disease by patient matrix. Every row corresponding to a patient, every column corresponding to a disease. We want to learn a function, f, that partition this matrix into P1, P2, P3. Each partition corresponding to a patient cluster. In the traditional clustering setting, this function is

directly applied to this matrix. While in spectral clustering this function is more involved. So next, let's talk about how do we construct this function in the setting of spectral clustering. The first step of spectral clustering is to construct the graph. The input to the spectral clustering are patient vectors. The first step we want to connect all those patients together based on their similarity. In other words, we want to construct this similarity graph. So, every node on this graph is a patient, and every edge indicates the similarity between two patients. Once we have this graph representation, we can store that efficiently using a matrix. Just like what we described in the PageRank, we can use the same adjacency matrix representation to store the similarity graph. The second step of spectral clustering is to find the top k eigen value of this graph. For example, here w represent the top k eigenvectors, and the middle matrix is the diagonal matrix with eigenvalue on the diagonal. This this third step is we want to group those patients into k groups using the eigenvectors. This is the high-level algorithm for spectral clustering. It depends on how do you implement each steps, there are many different variations of spectral clustering. Next let's look at some of the variations.

217 The first thing is, how do we construct the similarity graph? On high level, we want to view the similarity graph, based on the local relationship between patients. So similar patient will have a stronger relationship. There are several common ways for building such graph. We can base on epsilon neighborhood. We can use k-nearest neighbors. We can also fully connect the graph but assign a different way to the address. Next, let's look at them in more details.

218 Let's start with Epsilon Neighborhood Graph. Let's illustrate the idea using this example. Every node here indicate a patient, and in this example we have ten patients. For Epsilon Neighborhood Graph, what we are going to do here is, we'll connect patients if they are within epsilon distance to each other. For example, this two patients are within epsilon distance, so an edge formed between them. But the distance between these two patients is greater than epsilon, so there's no edge between them. In this case, the epsilon is indicated by this length.

219 Another way to compute similarity graph is based on K-nearest Neighbor Search. For example, we have this 10 patient over here. We want to perform K-nearest neighbor search from each patient and the resulting graph is this directed graph indicate the two nearest neighbor from each node. For example, the two nearest neighbor of this patient are this one and this one. So, one benefit of this k-nearest neighbor graph is the graph can be very sparse when the k is small, and there is several different variation of such graph. For example, we can have the edge to be binary, just zero and one, or we can actually assign different ways based on the distance between those two neighbors.

220 Another way to construct the similarity graph is to just use the fully connected graph, but parameterize the edges differently, based on similarity. For example, for this ten patients, we can connect everybody to everybody, have this fully connected graph. Then the edge wave will be determined by this Gaussian kernel or this Radial based function. For example, the edge wave, Wij between those two patients, can be computed using this formula, which indicate the Gaussian kernel.

221 Now let's do a quiz on absolute neighborhood graph. Given a set of patients and their two-dimensional position in this space, what is the optimal value for epsilon? Is this this long, or this long, or this long, or this long?

222 The correct answer should lead to class string structure on the graph. So, when the epsilon is too small, the graph will be highly disconnected. That will lead to many clusters. When the epsilon is too large

then everybody is connected to everybody else. The entire graph becomes a single cluster. So good epsilon should review the clustering structure. For example, when we choose B, then we'll see this four clusters. Or we can choose C that give us this two bigger clusters. So, both B and C are correct answers.

223 So far, we explained the high-level ideas of spectral clustering. Depending on how you implement each steps, there are many different variations. If you want to learn more about different variation of spectral clustering, please refer to this tutorial, and to learn about all those different variations. For example, we can have this very simple unnormalized spectral clustering, just involve building the graph compute eigenvectors, then perform k-mean clusters on those eigenvectors. This is probably the simplest spectral clustering algorithms out there. Then there are different enhancement on the original algorithms, by normalizing the graph differently. For example, this normalized spectral clustering published in 2000 and another different normalized spectral clustering published in 2002. Spectral clustering perform really well in practice even when the data are high dimensional or noisy. In fact, there are very good theoretical foundation behind spectral clustering. If you are interested in learning more, you can refer to this paper. In that paper, they actually illustrate theoretically why spectral clustering works. Intuitively speaking if the data are not really clusterable in the original space. By performing the spectral clustering, you can transform the original data into the space where the clusters are well defined. So, for example here, the color indicate different clusters, in the original space, it's very difficult to carve out these three clusters. But when you perform spectral clustering, in the eigenspace you can find all those three clusters are well separated.

224 Congratulations. You made to the end of the videos. Thank you for joining me. I'm Jimmy Son, signing off.

12 Dimensionality ReductionTensor Factorization

225 So now we're going to discuss dimensionality reduction. This is our method for distilling very complex and noisy raw data into robust core components. We'll start by talking about dimensionality reduction. Including singular value decomposition, principal component analysis, and CUR decomposition. Then, we'll discuss a more advanced method called Tensor Factorization. That handles higher order interaction among data. Finally, we'll see how this method applies to predictive modeling and phenotyping in healthcare.

226 First, a recap of clustering algorithms. In previous lectures, we talked about hard clustering, such as k-means, hierarchical clustering. We talked about soft clustering algorithms such as Gaussian mixture model. We also talked about scalable clustering algorithms. Such as, mini batch k-means and DBScan. In this lecture, we'll talk about dimensionality reduction. The reason for dimensionality reduction is because, it's often more robust and efficient to work with low dimensional data. Such as, a data set with 10 to 100 dimensions. But the original data, or the raw data often contains of much higher original data set. Say, in order of tens of thousands to even a million dimensions. So, the dimensionality reduction is a set up method to reduce dimensionality of original data. While still maintaining the underlying data characteristics. So ,we talk about Singular Value Decomposition, SVD, and Principal Component Analysis, PCA. Those are two classical method that summarize original set of features as linear combinations. Then we also talk about CUR decomposition. Instead of using linear combination of original features. CUR samples actual columns and rows from the original dataset. So, it's more intuitive and often leads to sparse result. Then we'll talk about tensor factorization, as another set of method to reduce dimensionality. When we're dealing with higher order tensor instead of matrixes.

227 First, let's talk about singular value decomposition, or SVD. SVD vectorized the input matrix X as product of three matrices. U, sigma, V transpose. Where the U and V matrices are which means U transpose times U equals to V transposed times V equal to identity matrix. Visually, given a large matrix X. And here, every columns represent a disease. Every row represent a patient. For example, X1 corresponding to the disease indicator of all the patient for the first disease and X2 represent the disease indicator for all the patient for the second disease, and so on. Then we can vectorize X as matrix U times a diagonal matrix sigma times V transpose. Here X consists of all the input data, the patient by disease matrix, and the U matrix consists of left singular vectors. So, every column here in U represent left singular vector. And sigma is a diagonal matrix, whereas diagonal elements are non-zeros, and off-diagonal elements are all zeros. And all the sigmas are the singular values. They're assorted in descending order. So, sigma one is greater than or equal to sigma two, which is greater than or equal to sigma three, and so on. Then the columns in V corresponding to the right singular vectors. And this is the mathematical definition of singular vector decomposition.

228 Now let's see an example of SVD. Given SVD over X, if we represent SVD as the matrix factorization, then we have this U times sigma times V transpose. However, we can also separate all those columns of U and V separately to obtain a different view we call spectral view. Here, sigma 1 times U1 times V1 transpose is a rank one approximation of the original matrix X, and sigma 2 U2 V2 transpose is another rank one approximation of the original matrix. If you sum up all those rank one approximations, you get original matrix X. So visually we have matrix X represent a set of documents and a set of terms used in those documents. For example, here we have two set of documents. The red ones are computer science document, and the purple ones are medical documents. And you can imagine the vocabulary used in

these two sets of documents are very different. So, the terms used in CS documents are very different from the terms used in the medical documents. If we want to summarize this big matrix using SVD, then we can summarize this input matrix as product of three matrices, U sigma V transpose. And if we look at the spectral view, what it means is, it's a set of rank one approximation. For example, this red part corresponding to this first rank one approximation, so this is sigma 1, U1, V1 transpose. Putting them together gives us the CS documents and the corresponding terms. Then we have another rank one approximation. This purple one's sigma 2, U2, V2 transpose. Multiply them together. Gives us the medical documents and terms. So, by looking at the spectral view you can see that we reduce the of this huge matrix Into this two-dimensional space. One for CS document, one for medical document.

229 Next let's talk about the property of SVDs. SVD is the matrix factorization, factorize input X as the product of three smaller matrices. U sigma, V transpose. And V are the eigenvectors of the covariance matrix X transpose X. So, if we multiply X transpose X together and you carry out this calculation, you'll find out V times sigma square times V transpose equals X transpose X, which means V is the eigenvector of X transpose X. With eigenvalues Sigma square. And similarly, U are the eigenvectors of the Gram matrix, or inner-product matrix X X transpose. So here, given X X transpose, if you carry out this calculation, you find it equals U times Sigma square times U transpose. Which means U is the eigenvectors of X, X transpose with the eigenvalues sigma squared.

230 Let's do a quiz to understand SVD better. Given a document-by-term matrix like the one we have shown you in the previous slide, what is A transpose A? Is it a document-to-document similarity matrix? Or is it term-to-term similarity matrix? Or is term-to-document similarity matrix?

231 The answer is, it's actually a term-to-term similarity matrix, because when you compute A transpose A, the number of rows and columns of this resulting matrix equals the number of terms. And every element represents the similarity between two pairs of terms.

232 Next let’s talk about principal component analysis, PCA. PCA is very related to SVD. So, we already know, using SVD, we can factorize the matrix x as a product of three matrixes U sigma V transpose. Now if we group u and sigma together Into one matrix and we call that principal components and we transpose, we call that loading. This essentially give us the principal component analysis. So visually, given this large input matrix, n by m, and every row represent a patient, every column represent a disease then we can factorize this into U sigma and V transpose, and if we group U and sigma together and the result of U times sigma become the principal components, and V transpose become the loading matrix. And in particular, this U1 times Sigma1 give us the first principle component. The direction of the first principle component is specified by the corresponding vectors in the loading matrix.

233 Now let's illustrate PCA with a visual representation. Given a set of two-dimensional points scattered like this. If we want to reduce this set of two-dimensional points into one dimensional space, we need to find a direction to project those points to. For example, the first principle component direction is pointing this way, and if we project all those points onto this first principle components, the corresponding offset along this direction will become the coefficient to represent those points. And they are the first principle components. For example, if x axis is representing weight and y axis represent height, every data point represents a specific individual, then if we want to project those individuals into a one-dimensional space, and this one dimension is a linear combination of weight and height and the offset on this one dimension is the first principle component.

234 So what's the problem with SVD or PCA? It has a sparsity problem. For example, most of the input dataset for analytics are often sparse, meaning that only a small number of elements in the large matrix are non zeros, majority of these elements are zeros. For example, every row represent a patient, every column represent a possible disease, then for a given patient, there's only a few disease this patient has, so this input matrix is very sparse. So SVD is one of the best dimensionality reduction approach, because it gives the best low rank approximation. However, the result of SVD, in particular the U and V matrices, are large and dense, so SVD destroys the sparsity in the original data. As a consequence, the result from SVD would take a lot more storage space and competition with time for the subsequent analysis. Alternatively, we may want to use those factorizations that preserves the sparsity. For example, we can use actual columns and actually rows from the original matrix to form the factorization, and this is called CUR decomposition. C represent the sample columns from the original matrix and R represent the sample row from the original matrix. So, CUR, in this case, maintains the sparsity of the original data. As a result, the storage cost of CUR decomposition is much smaller than SVD.

235 So CUR decomposition uses actual rows and columns to form the factorizations. So, here's the definition for CUR. Given an input matrix A, find a set of columns in C and a set of row in R and a matrix U, such that the norm of A minus CUR is small. This norm must indicate the errors of approximating our original matrix A using C times U times R. So visually given the input matrix A, which is m by n, we want to approximate A with a smaller matrix C which is m by c, times a small matrix c by r, and a matrix R, which is n by r. Here, C come from the columns of A, and R come from row in A. Let's use this two-dimensional example to compare CUR to SVD. So SVD will try to find the best direction to project all those data points to. And this direction often is a linear combination of all these data points. However, CUR will try to find an actual data point. And to project the rest of data points towards that direction. For example, CUR may sample this data point over here, then try to project all the other data point to this direction.

236 Next, let's talk about the actual algorithm for CUR. In the input to CUR algorithms is the matrix A and the number of columns we want to sample, C and the number of row we want to sample, R. And the output of CUR is these three matrices. C, U, and R. There are many different algorithms that achieve CUR the composition. There's different computational complexity and different approximation error guarantees. And this lecture will show you an algorithm for CUR that based on SV. So, in this specific algorithm, the first step is we do SVD decomposition. Find a top rank-k approximation that's the SVD. Then we sample c columns to form matrix C, and sample R rows to form matrix R. Finally, the U matrix can be computed as C pseudo inverse times A times R pseudo inverse. So, this symbol represents pseudo-inverse. Pseudo-inverse of matrix X can be computed with SVD. So, if we have SVD of X which is U times Sigma times V transpose. And the pseudo-inverse of X is actually V times sigma-inverse, times U Transpose. And the property of pseudo-inverse is X times X-plus, equal to identity. Next, we talk about these two key steps. How do we sample columns and rows based on SVD. So, from the previous step, step one, we have this rank here approximation using SVD. Approximate matrix A by U times sigma times V transpose. Here we have K columns in U and the same for V. Sigma is K by K matrix, were actually sampled based on the row lengths of U and V matrix. If the row length of this corresponding patient is large, then we're more likely to sample this patient to form R. Similarly, if the row lengths in this main matrix is large, then we're more likely to sample the corresponding columns to form C. So, the probability of sampling is proportional to the row lengths of U and V matrix.

237 Here's a quiz on CUR decomposition. If we apply CUR on this patient-by-diagnosis matrix A, to get the CUR decomposition from this patient-by-diagnosis matrix, then what are the columns in C? Are they

actual diagnoses, combination of all diagnoses, or combination of a subset of diagnoses? Similarly, what are the rows in R? Are they actual patients, combination of all patients, or a combination of a subset of patients?

238 And the answers are actual diagnoses and actual patients. So that's a key characteristic of CUR decomposition. You sample actual columns and row from the input matrix. In this case, we'll actually get diagnosis and patient from the original matrix to form the C and R matrices. So, they are more intuitive, and also preserves the sparsity in the original data.

239 Next, we want to talk about tensor factorization as another dimensionality reduction method, but first, what is a tensor? Tensor is a generalization of a matrix. A matrix is actually a special case of tensor of the second order, so we call matrix a second order tensor. Tensor can better capture interactions across concepts. For example, we can have a patient by diagnosis, by medication tensor, a third order tensor, and we call those patient diagnosis medication or the tensor mode. This element indicate, for this given patient, what diagnosis she has, and for treating this diagnosis, what medication has been given. This tensor representation can capture diverse data types, so the element of this tensor can be quite diverse. It can be binary, indicate whether patient has been taking this drug treating this disease, or it could be count, or integer, at counting the number of times such diagnoses has been given, and for treating this diagnosis, this number of medication has been given. Or it can be some numerical continuous values, so it's quite general.

240 Now let's talk about some basic operation on tensor. Tensor slicing is an operation to extract a subtensor, in this case, a matrix. So, the idea is to set all the mode except two modes the same. By doing so, we'll get a matrix. For example, given this third alter tensor, patient, diagnosis, medications, we can get a diagnosis medication matrix for a specific patient. So that's tensor slicing along this particular patient dimension, for example, this healthy patient we can extract a specific matrix for her. Then for a sick patient, we can extract another slice of this tensor that corresponding to this patient that is sick. Similarly, we can extract patients associated with medication for treating a specific disease, say hypertension. We can also extract all the patients and their corresponding disease, by a particular medication, say beta blocker. So, these are all different ways for slicing a tensor to get a matrix.

241 We can multiple tensors from electronic health records. For example, we can construct this patient by medication matrix, or second order tensor. We can construct this patient-diagnosis-medication tensor, a third order tensor. We can construct a patient-lab result tensors and we can construct a patient-symptom tensor, and finally, we can construct a patient-diagnosis by procedure tensor. There's many different ways to construct those tensors from electronic health records. Another important tensor concept is rank-1 tensor. So, rank-1 tensor is altering product of a set of vectors, one from each mode. For example, given this third order tensor x, if x is rank-1, then it equals to this other product of three vector, a, b, and c. So, the mathematical notation of this rank-1 tensor is represented as this auto product operations of the three-vector a, b, c. And the corresponding element in this tensors, this ijk element in this tensors is simply multiplication of the ith element from vector a, the jth element from vector b, and kth element from vector c.

242 Now let's try to connect tensor factorization and rank one tensor to phenotyping. Here's one example of phenotype we want to discover from data. And those phenotypes actually corresponding to a rank one tensor. In this particular case, we have a patient vectors, diagnosis vector, and a medication vector. And this give us a rank one tensor, which correspond to a phenotype. So, in this particular

phenotypes, 40% of patients has this phenotype, meaning that 40% of entries in this patient vectors are non-zeros. The rest are zeros. The corresponding diagnosis in this diagnosis vectors is hypertension. And for treating hypertension, three medication has been commonly prescribed. Beta blocker, diuretic, and so on. So, in this case, phenotypes is a group of patient that share common characteristic. They share some common diagnoses and common medications. Next, we'll show you how to use tensor factorization to discover such phenotypes.

243 Now we want to talk about how do we extract phenotypes using tensor factorization. Given this patient by diagnosis by medication tensor, we can factorize this tensor as the sum of R rank 1 tensor, each one of this rank 1 tensor corresponding to a specific phenotype. Each rank 1 tensors has three factors, patient factors, diagnosis factor, and medication factor. An ultraproduct of this three factors give us a rank 1 approximation of the input tensor, and we have R of those. And combine all of them together, give us approximation of the original tensor, and the lambda corresponding to the importance of these phenotypes. You can imagine different phenotypes may have different importance for representing this input population. So, lambda captures that. And this intuitively, how do we use tensor factorization result for phenotyping? Next, let's see what's the computational method for conducting tensor factorization.

244 Simply the composition factorized in X as sum of a set of rank one testers. Mathematically, it can be represented as tester X, approximated by a sum from one to R and a set of rank one testers. Each rank one tester is represented by a scalar lambda i, and set of vectors, one for each mode, ai1, ai2, and ai3. And this whole thing is the ice rank one tensor to approximate the input tensor. Of course, we can reorganize all this vectors into matrices. And that give us the model. So visually, what it does is, all those corresponding vectors, coming from the same mode, will be put together, become a matrix A(1). And similarly for all those vectors coming from the second mode, will be put together so we get A(2). And finally, we get A(3). All those three matrices are part of the model, and that's the output of CP decomposition. Again, all those corresponding lambdas scalar values tells us how important each rank one tensors are also being put together into this vector lambda. And this whole thing give us the model for CP. And this is the high-level intuition of CP decomposition. Like SVD, another matrix factorization, tensor factorization start to become a standard operations. Depending on the loss function and different constraint put onto those components, there is different algorithm to perform CP like decomposition. For phenotyping application, we use a variant of CP Decomposition with non-negative constraints. However, overall, as a data analyst, you can probably treat this tensor factorization as a black box, just like SVD and PCA.

245 So what is the process for using tensor factorization for phenotyping? So, you already have input tensors. Using tensor factorization, you have already learned a set of phenotypes. In particular, you have learned the phenotype definition based on the combination of diagnosis and medication. Next a new set of patient comes in. How do you apply those existing phenotype definitions to this new patient? In fact, just like PCA, you can project this new patient's data towards the direction, as specified by the phenotype definition. Then you will get a low dimensional representation, which is every row represent a patient, and every column represent a phenotype. For example, this would tell us, for a given patient, which phenotypes she has.

246 Next let's see how we can construct tensor vectorizations for phenotyping and use it for predictive modelling. So patient EHR data are often represented as event sequences. For a specific patient, we have five records, from t0 to t4. Each record contains one or more diagnosis and medication pairs. For example,

diabetes and sulfonylureas at t0. Then at t1, three different diagnosis and medication pairs happen in this patient record. And t2, two diagnosis and medication pairs happen. And t3, another two pairs. And t4, heart failure happened, and loop diuretic has been used. And this is the event sequence for a given patient. And note that the goal of this predictive modeling is to predict heart failure, which is event happen at this time, t4. To construct tensor, we need to find the index day and observation window, then aggregate all this diagnosis medication pairs within the observation window to populate this tensor. For example, this patient diagnosed with heart failure at t4, which is the index date. Then we look back two years to construct observation window, then we count the number of occurrences of all diagnosis and medication pairs. For example, this Diabetes and Sulfonylureas happened twice within the observation window. The corresponding element in this tensor will be two. Then we go through all the patient to populate the entire tensors, we'll have this patient by diagnosis by medication tensor. Based on the information from the observation window prior to the index date. For example, in this specific case, we can construct a tensor of size 31,000 patients by 170 disease diagnosis by 470 medications. And 15% of patients in this tensor had heart failure. And we want to apply tensor factorization on this tensor to extract phenotypes, then use those phenotypes as features to predict whether patient will have heart failure or not.

247 So here's a predictive performance. Here, we're trying to compare different dimensionality reduction method. And using those low dimensional representation as features. To predict whether the patient will have heart failure or not. In this case, the baseline performance is indicated by this line. Is AUC close to 0.7? Then we're comparing three different methods. a tensor vectorization, non-active-matrix vectorization, and principle component analysis. And this is the baseline performance using all the raw data, 640 features. And the classifier we're using, are all the same, it's logistic regression with L1 regularization. Then, as we increase the number of phenotypes or increase the load emission of representation. You can see the performance improves for PCA. Similar trend has been observed for non-negative matrix factorization, NMF. And also, for the tensor method, which is based on non-aggregative tensor factorization. And here, you notice that, with only a small number of phenotypes, in this case, 30. All those dimensionality reduction methods outperforms the baseline method, which use 640 features. This really shows all those dimensionality reduction method really works. So, as we increase the number of phenotypes, you can see tensor method and NMF perform better than PCA. But between those two, they are quite similar in term of predictive performance.

248 Next, I want to show you intuitively how those phenotypes look like. Using tensor factorization, we're able to extract different disease phenotypes, some corresponding to major disease such as diabetes without complications, so we call this uncomplicated diabetes phenotypes that cover 17.6% of the population. Then we have another phenotypes corresponding to mild hypertension which covers 31.1% of patient that has hypertension, and two other medications commonly used for treating hypertension. In this particular example, the results from the tensor factorization method are presented to a cardiologist, and he thinks those result make sense, and assign label as attacks. And the tensor factorization mess not only can discover major disease phenotypes, but also can discover disease subtypes. Here are three different phenotypes discovered by tensor factorization. They all shared a common diagnosis, which is hypertension. But the medication for treating those patients with hypertension are very different. Because of that, the cardiologists give the labels of these three phenotypes as mild hypertension, moderate hypertension, and severe hypertension.

249 So you may wonder how does tensor factorization compare to this nonnegative matrix factorization? So, tensor factorization can provide much concise phenotype representation. For example, here's one phenotype coming out of tensor factorization. It consists of two diagnosis and a set of medications. Well for the corresponding phenotypes, in NMF, it looks a lot more complicated because it captured all interaction between diagnosis and medication and the list is much longer. In fact, there is over 1000 different combination of these medication has to be specified in order to summarize these phenotypes. So that the tensor phenotypes is much more concise. As a result, it's much more intuitive to present a tensor phenotype to clinicians.

250 So in summary, we talk about tensor factorization as a way for doing phenotypic, and it represents a patient as the tensor. For example, this patient diagnosis medication tensor, then summarize that tensor as a set of rank one tensors. There's several different benefits for using tensor factorization for phenotyping. First, In the unsupervised method, we can discover multiple phenotypes simultaneously, without any supervision from experts. And the resulting phenotypes can have predictive power. As we have shown you, using those phenotypes, we can predict heart failure better than using the raw data.

13 Patient Similarity

251 In this lesson we'll talk about how patient similarity can bring up a new paradigm of medical practice. We'll discuss how to find the most similar patient for a specific clinical context. We'll also talk about how patient similarity can support pragmatic trials and practice-based medicine. Finally, we introduce a supervised distance metric learning algorithm for patient similarity.

252 To motivate the importance of patent similarity search we need to understand the paradigm shift of medicine. Traditional paradigm considered randomized clinical trial as the gold standard for generating new evidence, and this paradigm is often called evidence-based medicine. The current recommendation for clinicians is to follow the evidence generated in the medical literature or clinical guidelines. And this evidence are largely created through RCT. There are several major challenges with this paradigm. For example, patients are heterogeneous, and can be very different from one another. And this one size fits all solution may not work in many clinical scenarios. Sometimes the guidelines are not up to date, sometimes they're not applicable to a specific patient. Thanks to the growth of electronic health records, we have a new paradigm that is emerging. We can conduct pragmatic trials based on EHR data. We can even consider doing practice-based medicine if the data driven evidence is strong. This new paradigm is called precision medicine, where personalized medical decision making is recommended. In this new paradigm, it is extremely important to be able to measure similarity among patients for a given clinical scenario. Next, let's elaborate both paradigms in more details.

253 The traditional paradigm is sometime called evidence-based medicine. The overall theme of evidence-based medicine is to make medical decisions based on the well designed and conducted research. And evidence-based medicine follows this four steps. It starts with perspective randomized clinical trials to test hypothesis. The successful hypothesis become evidence, which can be medical publications or new drugs that has been approved. Then medical experts work together to organize and prioritize all the related evidence into clinical guidelines. Finally, the clinicians apply this guideline in practice for treating patients.

254 Now let's talk about the new paradigm precision medicine. The goal of precision medicine is to create a new year of medicine in which researchers, health care providers, and patients all work together to develop personalized care. And precision medicine follows the following four steps. You start with pragmatic trials, which utilize large amount of historical data in the ehr systems to generate data driven evidence. Then we can apply patient similarity search for a given individual to find the similar patients. Then figure out what worked for those similar patients. Then recommend those treatment for the current patient. And this is often called practice-based evidence. If we follow practice-based evidence, we'll be able to create individualized recommendation or personalized care for a given individual. And this is how we achieve precision medicine.

255 Next, let's talk about randomized clinical trials or RCT. To conduct RCT we start with the study population then we randomly assign everybody in this study population into two groups. In the control group, everybody is taking the current treatment or placebo and in the treatment group, everybody is taking the new treatment we're testing. Then we look at the treatment outcome for both groups and there will be patients have improved outcome in the control group, and some do not have improved outcome. In the control group and similarly in the treatment group there will be people improve their outcome and some people do not improve their outcome. An RCT will compare the outcome from both group trying to figure out whether the treatment group on average have better outcome than the control

group. And if the RCT confirmed the treatment group on average, have better outcome than the control group, would consider this trial as a success. Otherwise, this trial is a failure.

256 Now let's do a quiz on RCT. What are some drawbacks of RCT? Here are the options. It requires a controlled environment. It generally tests only one thing at a time. RCT can test new drugs. RCT is expensive and time-consuming. RCT discovers causal relationships. RCT deals with noisy data.

257 Here's the answers. RCT requires a controlled environment. The studied population in RCT are often times carefully selected with very strict inclusion and exclusion criteria. So, the studied population often do not reflect the general population in the real world. Another drawback of RCT is it generally tests only one thing at a time. Oftentimes a RCT is specifically designed to test one drug. If the hypothesis of this RCT is rejected, the entire effort will be wasted. So, in that sense, RCT can be very risky. And RCT does test new drugs, but this is not a drawback. Another drawback of RCT is, it is very expensive and time-consuming. Because we have to conduct a perspective study by recruiting patients and follow up with those patients, collecting data, then analyze that data to draw a conclusion so that can be very expensive and time consuming. RCT does discover causal relationship thanks for the randomization process, but this is not a drawback. Finally, because RCT is prospective study, and the data are carefully designed and collected. Often time they are clean. So, we don't have to deal with noisy data

258 Next let's talk about pragmatic trials. So, in traditional RCT, we generally measure the efficacy of a treatment that produces under ideal conditions. Often use carefully designed patient population in a research clinic. Pragmatic trials on the other hand, try to measure the effectiveness of a treatment that's produced in a routine clinical practice. For example, when a patient comes to a clinic, we can do a similarity search against a large patient database, try to find a similar patient to the current patient, then group those similar patients by treatment they have taken. Then look at the outcome they are getting, then recommend the treatment with the best outcome to the current patient. This is the overall idea for pragmatic trials. And the design of pragmatic trials reflects a variation between patients that occur in the real clinical practice and aims to inform choices between those treatments that works for a given individual.

259 Now let's do a quiz on pragmatic trials. What are some benefits of pragmatic trial? Can it test new drugs? Can it operate in a real-world setting? Can it automatically gather more data through this process? Is it expensive and time-consuming? Can it discover causal relationships? Does it deal with noisy data?

260 So let's go through this list, one by one. First, pragmatic trials cannot test new drugs. Because pragmatic trials depends on the historical data and all the treatments already happened in the past. In that case, we won't have any information about effectiveness of a new drug. And pragmatic trials do operate in real world setting, which is a benefit. Because it operate in the real-world setting, pragmatic trials can automatically gather more and more data over time. Because every patient encounter will be able to generate new data that can be used for future pragmatic trials. Comparing to RCT, pragmatic trials is not expensive and time-consuming because we're dealing with historical data that already been collected. In general, pragmatic trials aren't able to discover causal relationships because randomization is not involved. Because we're operating in a real-world setting, pragmatic trial has to deal with the noisy data generated in the electronic health record.

261 Now let's look at what's the process to utilize patient similarity today. We start with practice-based medicine. For a given patient, we'll look for similar patients. Then based on what happened to

those similar patients, we can generate hypothesis, what could work best for the current patient. So those hypotheses, are generated based on retrospective evidence. In order to confirm those hypotheses, we oftentimes, have to go back to randomized clinical trials, to confirm those hypotheses through a prospective study. Once we generate those evidence, we can update the clinical guidelines, then apply those guideline in practice. Patient similarity, and practice-based medicine provide an intelligent way to generate hypotheses, in order to guide the randomized clinical trials, and evidence-based medicine. Here's another illustration of how we can use patient similarity in clinical practice. Imagine a patient comes into the clinic. The first thing we can do is, try to see whether appropriate guideline available to apply to the given individual. If there are a proper guideline, then we can directly use the clinical guideline to treat this patient. If we don't have a guideline, that is suitable for this individual, we could go ahead look for similar patient in the history. If we do have a large cohort of similar patient, we can use practice-based medicine, figuring out what treatment, were likely to work based on similar patients. If we don't have enough similar patients, available in the database, then we have to rely on professional judgment. And as we go through this practice-based medicines, many times, we can develop hypotheses, that worth conducting RCT. If we're ready for RCT, then we can conduct the RCT, to improve the guideline. If we're not ready for RCT, we can still use existing guidelines, along with similar patients' information to treat future patients. This region indicate evidence-based medicine, where clinical practice is largely depends on the guideline. And the green region, indicate a way to generate practice-based evidence from data. As you can see, both paradigm are quite complementary to each other. And they can work very nicely together, as indicated here. But the challenge is, how do we find similar patient from historical data? Next, we'll illustrate some of the algorithmic approach for patient similarity.

262 One way for solving patient similarity problem is to approach this as distance metric learning problem. Assume we have a list of patient, and we know who is similar to whom. We also have a patient representation, and every patient is represented by a feature vector. For example, here are two patients, X1 and X2. And here, Y indicate the ground truth. For example, if two patient, X1 and X2, are similar, then they have the same label. If they are different, then they will have a different label. Then it's become a supervised distance metric learning problem. Given the ground truth label and feature vectors, we want to learn a distance metric, d(x1,x2). And this function will tell us about the distance between those two patient. If they are similar, the distance will be small. If they are not similar, the distance will be large. Besides distance metric learning, we can also use a graph-based similarity learning to figure out patient similarity. For example, given a set of patients, we want to figure out who is similar to whom. In medicine, we have a lot of medical knowledge that often represented as ontology, or a graph. Here is the human disease network. Every node indicate a disease, and every edge indicate a connection between two disease. And if you want to know more about medical oncology, we have a separate lecture specific on that. Now given the medical ontology or, in this case, a disease network, we can connect those patients to the diseases. For example, this patient have one disease, this patient have two. And the graph-based similarity learning is trying to figure out, given this heterogeneous graph that connecting patients to diseases. How can we figure out who is similar to whom?

263 Now let's learn a specific distance metric learning algorithm. Called locally supervised metric learning. First, let's illustrate the intuition behind this algorithm. For example, we want to develop a distance metric under a specific clinical context. Such as, heart failure management. We have a query patient comes in and using some base line similarity measure. For example, Euclidean distance or cosine similarity we can retrieve a set of patients. That are potentially similar to this query patient. This

algorithm is a supervised approach. So, we have some ground troops label. For example, we know, these four patients are indeed similar to this query patient. And we call them homogeneous neighbor. And we also know, these four patients are not similar to the query patient. We call them heterogeneous neighbors. And given these two sets of neighbors, we want to change the underlying distance measure. So that, the homogenous neighbor becomes closer and closer to the query patient. And the heterogeneous neighbor, becomes further and further away from the query patient. And we want an algorithm that can do this automatically. Now, understanding the intuition behind the algorithm. Now, let's formulate the problem mathematically. The goal of this problem is to learn a generalized Mahalanobis distance. For a specific clinical context. That is, we want to learn a sigma matrix. Which is d by d, where d is number of dimensions in the feature vectors. We assume the sigma matrix is symmetric and low rank. So that we can factorize sigma, as w times w transpose. Where w is rectangular matrix of d by K, where K is much smaller than d. And in this case, the goal is to learn the w matrix. Mathematically, we want to learn this distance function, d(xi,xj). Which is very similar to equating distance. Except the sigma matrix in the middle. And the sigma matrix, is this symmetric low rank matrix, can be factorized as w times w transpose. And w, is what we need to learn from the data. So, the locally supervised metric learning, follows intuition we explained earlier. We want to define this margin for each patient. The margin is defined as, the total distance to the heterogeneous neighbors. Subtract the total distance to the homogeneous neighbors. Intuitively, it's indicated by this gap over here. And we want this margin to be large, so that the truly similar patient will be closer to the query patient. And we'd want to do this for all the patients. So, we could define the total margin, which is the summation over all the margin for each patient. And the goal is to maximize the total margin by changing the W matrix. Now we understand the objective function, is to maximize the total margin. We can rewrite the objective function in this matrix form, as trace of W transposed times H times W. And H is defined as, the difference between these two matrix. Le coming from all the heterogeneous neighbor, and L0 come from the homogeneous neighbor. Since H in this case, is a symmetric matrix. And the solution, is the eigenvectors of H with all the positive eigen values. And the complexity for solving this problem, is eigen validate composition of this matrix H.

14 Deep Neural Network

264 to describe the deep neural network we start with the simplest one that comprises of a single neuron a neuron is a computational unit that takes an input value x1 x2 to xn and their associate weight w1 w2 2 WN and a bias term B and going through a computational process and output a value Y and this computational process involve a linear combination and a nonlinear activation more specifically the linear combination produces an intermediate output Z which is sum of W I times X I plus the bias term B then we pass the Z through a nonlinear activation function G to produce the final output Y depending on the specific tasks Y can be either binary for classification tasks or numerical for regression task to learn a model of the single neuron we need to specify the nonlinear activation function G and learn the weight w1 w2 to WN and the bias term B from data

265 activation functions describes the nonlinear transformation in the neuron which needs to be specified by the modeler it is not usually learned from the data for activation functions there are a few popular choices including the sigmoid 10 age and rectified linear with sigmoid function the input can be arbitrary real values and output will be in the range of 0 to 1 which can be naturally interpreted as the probability of an event for example the probability of having heart diseases as a result a signal function is a popular choice for classification tasks mathematically sigmoid function is specified as 1 over 1 plus e to the minus X and Sigma function has vanishing gradient problems as we will show later neural network learning depends on gradient based optimization if the greeting is too close to 0 optimization process will not be able to make progress this is called vanishing gradient problem for example the gradient of both end of the signal function is very close to zero for example the gradient when X is very small or the gradient for X is very large are both close to 0 as a result it may cost vanishing gradient problem

266 10h is another popular activation function it has multiple mathematical forms it can be either specified as e to the X minus e to the minus x over e to the X plus e to the minus X or after some calculation you see it's equivalent to 2 over 1 plus e to the minus 2x minus 1 the second expression is probably computationally more efficient because it only involved one exponential calculation well in the previous case in in mauve two different kinds the output of 10H functions is centered around 0 and bounded between minus 1 to 1 in fact 10H is the shifted and rescaled version of signaling because of the risk Geylang 10H has larger greetings especially near zero however 10H still has vanishing gradient problems when input X is far away from zeros in this region or in this region

267 in the rectified linear functions which is called ReLu is a simple and more modern activation function rarely is specified as maximum over zero and input X the output of value is between zero and plus infinity and visually it's linearly increasing curve when X is greater than zero and threshold at zero rarely is very different from 10h and signaling in a sense that the output doesn't have an upper bound unlike Sigma and 10h rally does not suffer from vanishing gradient problem as grading is constant one for all the value of x greater than zero

268 so to summarize activation functions are crucial building blocks for newer networks there are a few popular choices of activation functions including sigmoid 10h and ReLu their relative relationships are plotted in this figure you can see sigmoid and 10h are bonded in small range while ReLu is only lower bounded by zero and does not have an upper bound to learn in your network model we need to specify the choice of activation function as part of the neural network architecture which activation functions to choose is highly dependent on the application

269 now let's consider the simplest neural network a single neuron and trying to see how do we learn the model parameters for such a neural network in particular we want to learn the weight w1 w2 to WN and the bias term B and we break down the computation of the neuron into two steps the linear combination and the nonlinear transformation the linear combination confused the weighted sum of x1 to xn and the bias term B we call this intermediate result z and the nonlinear transformation as the activation function over at z to produce the output Y in a supervised learning setting we want the output Y to close to the target value T for example T can be a binary indicator of whether a patient has the heart disease or not and why it's the prediction of such a target to measure the model quality we need to specify a loss function to measure how much difference is between the output Y and the target T for example we can use a squared loss or squared Euclidean distance like this our goal of training such a neuron is to minimize the loss function on the training data by adjusting the weight of the neural network

270 to use in your network s motto we have to learn the parameter of the neuron which are the weights w1 to WN and bias term B want to systematically move the weights and bias such that the output is getting closer to the target that is small loss one computationally efficient way for doing that is the stochastic gradient descent algorithm or SGD. SGD takes the training data set and the learning rate EDA as input if first initialize all the weight WI and the bias term B to some small random values then they start iterating over all the training examples for each training example here X is the input feature vector and T is the target label you will first compute the gradient vectors with respect to all the weight W and the bias term B note that this nebula operator represent all the derivative of the loss with respect to different ww1 to W n so it's n dimensional vector the derivative of the loss with respect to the bias is a scalar after we have the gradient we simply update the weight vector W and the bias term B in the opposite direction of the gradient here the EDA is the learning rate which need to be set or can be adjusted using a different algorithm and the key for neural network learning is how do you compute the gradient of all this different parameter efficiently

271 so to update an Iran based on the training data we need to perform two passes over the network one form has to compute output Y and the loss and one backward pass to compute the greeting for each parameter in this case we first compute the linear combination Z which is sum over all WI times XI plus bias term B then we perform nonlinear activation G over Z for example in this case the activation function G can be a signal function keep in mind this two terms have derivatives which will be used in the backward pass for example the derivative of Z with respect to WI is just XI and the derivative of Y with respect to G is y times 1 minus y for sigmoid the second derivative is a little bit complicated to derive but you can probably still derive that with rudimentary knowledge of calculus finally modern deep learning software package or have automated greeting computation so in most of cases you don't have to worry about specifying greeting yourself but it's important to note overall algorithms

272 after the four pass we know the output Y and then loss we can perform a backward pass to find all the derivative of those parameters and mathematically we're going to apply chain of the derivatives from output to input for example using chain rule the derivative of the loss with respect to WI can be specified as the product of three derivative the derivative of L with respect to Y and the derivative of Y with respect to Z and the derivative Z with respect to WI then we can do some calculation of the derivative to find out the first derivative e goes to y minus T at the second term equal is y times one minus y and the last one is just XI so the final derivative is y minus T times y times one minus y times XI so that's the derivative for WI similarly using chain rule the derivative of L with respect to bias term B is the product of three derivative as well the derivative of L with respect to Y the derivative of Y with respect

to Z and the derivative of Z with respect to B and similar calculation apply we have Y minus T times y times 1 minus y times derivative of Z with respect to B in this case it's constant so here's the final derivative the only difference is we don't have this X I term anymore and it's just Y minus T times y times 1 minus y now we have the derivative we can use them in any gradient based optimization algorithm such as the SGD algorithm we specified earlier to find the optimal weights and the bias

274 to train such a multi-layer in your network again we use stochastic gradient descent algorithm to do that input to the SGD algorithms are the training data and the learning rate we first initialize all those weight and bias at each layer to small random values then we process the training example one at a time we first compute the derivatives of the loss with respect to the weights and with respect to the bias term then we perform the gradient updates so this is very similar to the algorithm we shown earlier for training a single neuron again the key is how to compute those derivatives for an arbitrary deep neural network

275 the four of house of newer network is important for scoring a new data point to get the output value why it is also an Cuccia buting block for learning the new network as we will explain later in the backpropagation hours so now let's illustrate the for computation step by step here is an example given X1 X2 X3 a 3 dimensional data point we need to compute 3 linear combination for h1 h2 and h3 this 3 hidden unit in particular Z1 superscript in parenthesis 2 indicates the linear combination for unit h1 which is specified as sum over w1i superscription process is 1/10 X I plus B1 superscript in process is 1 so visually all this are highlighted here this Z1 superscript in parenthesis 2 eCos W1 1 superscript in parenthesis 1 times X1 plus W1 2 superscript empresas is 1 times X2 n w1 3 super scribbling process is 1 times X3 plus B 1 superscript in process is one here superscript impress this is 1 indicates the layer number in this case is the inflator after this linear combination H1 will perform a nonlinear activation G 2 superscript impresses 2 over this intermediate result Z1 superscript in process is 2 to obtain the output value of H1 which is represented as a 1 superscript in parentheses 2 so similarly for the second linear combination for unit H2 we can compute Z 2 super Scribbins two as another weighted sum over input X plus the corresponding bias term and likewise the unit H2 performed the same nonlinear activation over a different linear combination Z 2 superscript in processes 2 to obtain the output for H2 again we do the same for unit h3 to compute the linear combination these three superscription parentheses - then we perform the activation over the linear combination to get the output a3 superscript in parentheses - now we have all the output for unit h1 h2 and h3 they become the input for the next layer in this case it is the output layer in this network in particular the linear combination of the output unit Y is Z superscript in parenthesis free which is the weighted sum of all the AJ superscript in France this is 2 plus is biased term B superscription processes to note that unlike the previous layer since there's just a single output unit why there's only one subscripts for the weight and there's no index for bias term be visually is highlight here so these superscript process is 3 is sum of a one super scrubbing process one times W1 superscript in process is 2 plus a 2 super scribbling process is 2 times W2 super scrubbing process is Q plus a 3 superscript in processes 2 times W 3 super scripting process is 2 plus B superscription practices - likewise the upper unit wine performs the nonlinear activation function G superscript in parenthesis 3 over the linear combination G superscript in parentheses 3 to get the final output a superscript in parentheses 3 and it's also represented just simply sy

276 to summarize we put equations for forward computation for this particular neural network here the neural network is fully connected meaning that all the unit between layers are connected for example X1 X2 X3 connect with H1 H2 H3 then H1 H2 H3 connect with Y each unit computes a linear combination Z first then followed by nonlinear activation G as you can see there are a lot of symmetry here multiple

neurons are involved to perform similar operations and collectively they help to learn a complex mapping from input to output

277 this forward computation can be represented by a mall compact vector notations so the weight W IJ superscript in parentheses one become a matrix W superscript in parentheses 1 and the bias be super scribbling parenthesis becomes a vector and the linear combination for all those weights become a matrix vector multiplication so Z superscript in parenthesis is 2 as a vector EKOS W superscript in parentheses 1 temps input vector X plus this bias term B superscript in processes 1 we also extend activation functions to apply to vectors in an element-wise fashion so for example g superscription parentheses to applied to this vector z superscript in parentheses 2 is just applying the same activation functions to each element of this vector z superscript in processes 2 so likewise we have linear combination for the output layer the superscript in parentheses is 3 and getting the final output a superscript in parentheses 3

278 so far we have shown you a simple architecture with one hidden layers however in general this forward computation is not limited to that simple architecture a more general case of forward computation from layer L to layer l plus 1 can be specified with this simple equation so the superscript L plus 1 in parenthesis equals to this matrix of weights superscript impresses as L times the input from the previous layer a superscript appearances as L plus this bias factor be super scary impress SSL and similarly this activation applies for each element of the superscript L plus 1 and to get the output from this layer and it's also the input to the next layer

279 here's the complete summary of four paths of a newer network we can represent all the computation in a unit-by-unit fashion like this or using a more compact notations represented as vector form or in more general case for arbitrary depths of neural networks in fact is actually code to represent a neural network computation as a vector operation so that they can be run more efficiently in parallel

280 to recap we have talked about discrete InDesign algorithms for learning parameters for your network in this case we initialize the weight and the bias term to some small random values then for each training example we try to compute the corresponding derivative with respect to W and with respect to B then after that we just do a simple update based on the gradient for both W and B, so the key is to efficiently compute this two derivative that's where the back propagation algorithm comes in

281 how do we use back propagation to compute the gradient of the loss with respect to w ji superscript in process L so as we will show you that this particular derivative is a product of two term the first term is actually straight forward is AI superscript in parentheses L minus 1 so this is the input from the previous layer and the second term is more interesting is actually the derivative of the loss with respect to the linear combination the J's superscript in process is L so this term intuitively measures how much this particular node HJ at the L layer was responsible for the final error again let's use the chain rule so the derivative of the loss with respect to W ji superscript in parentheses l equals to this two terms as partial derivative of L with respect to the linear combination ZJ superscript in process as L and the derivative of VJ superscript and prances as L with respect to W ji and the first term by definition is this Delta J superscript impress assist L so we'll worry about that later that actually turned out to be the key part for back propagation but the second part is actually easy to derive so if we write out the formula for Z is really just the sum over WJ s times S Plus this bias term bs so if you look at this second derivative term you notice that all this weighted sum term we have a derivative at zero except one when this w ji

matches this wjs so in that case the corresponding input is the breeze out of the derivative so AI superscript empresas L minus 1 and that's how we get this formula

282 now let's look at the second derivative term the derivative of the laws with respect to the bias term BJ superscript in-process SL it's actually quite similar to the previous derivation for W and again here Delta J superscript L is the derivative of the loss with respect to the linear combination Z J superscript in processes L and using chain rule we can breaking down the derivative with respect to B as this product of two terms again the first term derivative of L with respect to Z is by definition this Delta term Delta J superscript in process SL again the second term was the same logic will lead to actually a constant one as the derivative so the result for this whole thing is just Delta J superscription processes L so now we understand how do we compute this two derivative terms and the key now is how do we efficiently compute all this Delta J term for arbitrary neural network

283 in summary now we have this two derivative one with respect to Wji1 with respect to Vj and it involved in W case the input from the previous layer and this Daode term is actually lead to the derivative for the output and in this BJ case is just this doubt at J superscript princess is L now we have to figure out how do we compute this doubter term efficiently in fact the Delta J superscript empresas L can be computed in a backward fashion from the output layer to the input layer

284 now let's see how do we compute all this doubt a term in a backward fashion at the very end of this network we have doubt a que superscript in parentheses is four in this case is just partial derivative of the loss with respect to Z K superscript n processes for and by definition in this final stage this Z case superscript princess is 4 is just output Y and the loss function can be specified here and in this particular example the loss function is the square loss in this particular case the derivative is just - t - y

285 now we have this Delta 4 layer for now let's see how do we compute the other term for layer 3 so in this case this Delta for layer 3 is partial derivative with respect to the loss for the superscription process 3 and again applying chain rule and we have partial derivative of L with respect to Y and the partial derivative Y with respect to Z superscript in process of 3 and the first term is just Delta 4 which we just computed in the previous slide and the second term is just the derivative of the activation function because these three Z superscription processes 3 is input to this activation function now we're just taking the derivative of this activation function so the final result is just minus t minus 1 times the derivative of this activation function G's superscript in parenthesis 3

286 now let's look at Delta superscript in parentheses - in the second layer in the second layer we have multiple unit h1 h2 and h3 correspondingly we have Delta 1 Delta 2 and doubt a 3 for the second layer so let's just look at this doubt I super scribbling process is 2 so in this case again we applying chain rule the first term is partial derivative of laws with respect to ZJ superscript in process is 3 in and the second term is partial derivative of ZJ's superscript process 3 with respect to VI superscript processes to the first term is actually easy is just Delta superscript in parenthesis 3 is something we just computed previously and the second term again we applying chain rule again and to get this two terms one is this is EJ super square be princes 3 write it out and with respect to a I superscript empresas to this input and then partial derivative of AI superscription process - whispers track - zi super scrubbing process is - so again with similar kind of derivation you will see that the middle term just lead to you one term here WI super su Caribbean process is - when AI this - AI matches and the rest cases they are all zeros and the last term is just the derivative of the activation function again so it's the derivative of this activation function G super scribbling process - and with the input VI superscription processes -

287 so now let's write down all this different Delta functions so we can see that Delta superscript parentheses for the last layer we have minus t minus 1 and then for Delta 3 we have minus t minus y times this derivative of activation function at the third layer and so on so you can see that the pattern already and in general this Delta J superscript L follows this form so it's really a summation of K of wkj super scrubbing processes L times Delta K superscript L plus 1 n times this derivative of the activation function at this else layer so the first term is a little bit complicated in the simpler example we shown here where we only have one output unit and there's only one term here but when you have multiple output unit and you can lead to multiple term and weighted by the weight and sum them up but the important thing is all this derivative at any position of this new network can be computed with a single backward propagation from this network so you don't have to go through this network multiple times just one pass backward to compute all the gradients for all this different Delta and then just multiply the staudte with corresponding input you can compute the derivative for W and or for the derivative with respect to the bias

288 now let's summarize the backward propagation algorithm it involves two passes of the newer networks one forward has is star wist input X and goes through the entire network to compute output value and in this process you will compute all this intermediate result the linear combination Z and the nonlinear activation a for each layer then we do the backward pass we first compute this Delta function for the output unit then going backwards to compute the rest of the Delta function and also use that to compute the derivative of the loss function with respect to W and the derivative of the last function with respect to the bias term B so that's the backward propagation algorithm for neural networks

15 Convolutional Neural Network

289 in previous lessons we introduced the feed-forward neural network all the unit from one layer will connect all the unit in the next layer so between this two consecutive layers all units are fully connected when we have high dimensional input data for example medical images with thousand by thousand pixels EHR data was over ten thousand medical codes the fully connectedness will have too many parameters therefore too expensive to learn to make the computation of more efficient we can force the neuron to have smaller number of connections for example in this figure each hidden unit h1 and h2 only connect with adjacent input unit so we can extend this idea of local connectivity to many layers to obtain a deep but locally connected networks convolutional neural network or CNN is one type of this locally connected Network while using locally connected structure like this has significantly reduced number of parameters we can achieve further parameter reduction with weight sharing so weight sharing means some parameters over here will be shared at different location across the input so for example the model on the left have six different weight parameters W1 W2 u2 w6 and if we shared the weight at different locations then we can have one copy of this weight W1 W2 W3 for H1 then apply another copy of the same weights W1 W2 W3 for H2 so this way we can reduce the number of parameters from 6 to 3 and the weight sharing resembled the convolutional technique in signal processing which can be considered as a future or a kernel Hume any position in the input data this future operation is called convolution in signal processing

290 in addition to convolution or weight sharing we can add a pooling layer in addition to the convolutional layer the boast convolution and pooling layers have this translational invariant property for example we may want the output to be stable even when the input images shifted or distorted slightly since the translation is the main source of distortion for images and time series Network design was pooling layers can be more effective in recognizing the correct patterns regardless of translation for example we have this simple example of five dimensional input X1 X22 X5 then construed the convolutional layers it will generate two hidden dimensions H1 and H2 if we apply another pooling layer on top for example max pooling which takes the maximum value from all the input to this unit then that will give us the final output for example if we apply this to different input vector 0 1 0 0 0 vs 0 0 0 1 0 through this architecture we can see that for the first one after convolution will get a vector of W2 for H1 and 0 for H2 because if we do a weighted sum over this W1 W2 and W3 OS X1 X2 and X3 we can see that it will give us W2 because X1 and X3 at 0 only X2 is 1 and H2 likewise will give us 0 and the second input vectors will give us 0 and W2 because only this the fourth position have value 1 and the corresponding weight SW2 and that will give us that reach you at H2 position if we apply a max pulling on this size two vectors it will give us W2 assuming W2 is positive and so in this case in both example the final output will be the same although the input is actually translated by 2 pixel right you can shift X the first example and the second example very similar the only difference is as this first example need to be translated horizontally but 2 pixels to generate the second example so if we apply this architecture it will lead to the same output or we call this translational invariance because the output for both input or BW2 which is what we want

291 to summarize convolutional neural network it's a very powerful model for processing grid light structures such as images and waveform it utilized a set of special operations such as convolutions and pooling the advantage of a CNN motto is it gives us a very sparse interconnection in the newer networks

it also leveraged parameter sharing number of parameters in the CNN motto is often very small and it also supports translational invariance thanks for pulling and convolution operation

292 next let's look closely what CNN architecture looked like so a CNN architecture often involve a stack of operations such as convolution layer pooling layer fully connected layer usually a CNN architecture takes in inputs such as an image or time series and pass it the data through a sequence of convolution pooling and convolution pooling operations then finally it usually goes through a set of fully connected layers to generate the final classification output so that's a very typical CNN architectures so in this particular case we first have a convolution layer followed by max pooling layer followed by another convolution layer by another max pooling layers then we apply three convolutional layer consecutively apply another pooling layer then followed by three layers of fully connected layers next we'll go into more details so that we understand what those numbers mean and how do we design such architecture

293 so let's talk about convolution and pooling operations now let's take a look at the details of the convolution operation in this example we take one day input sequence of 1 2 minus 1 1 and minus 3 and pass it through a 1 the convolution operation and generate another one-dimensional output sequence minus 2 2 1 2 and 1 so how do we generate such output sequence we slide a future of size 3 sometimes we call this kernel over the sequence and every step it involve inner product between the future and a subset of the input so in this case we'll take element wise multiplication then sum the result up let's look at this example more closely so the parameter of the future is 1 0 minus 1 in fact there could be also be a bias term which I'm not showing here for simplicity then we perform an inner product of this parameters with a sliding window through the input sequence we start with the first input dimension with value 1 to construct a window centered around this value since this is the first dimension we usually pad the left hand side with zeros in this case we've had 1 zeros and we call this 0 padding and the output for the first dimension is minus 2 because 0 times 1 plus 1 times 0 plus 2 times minus 1 equal to minus 2 then we just perform such an inner product between the future parameters and the sliding window of the input sequence and we start to generate all this output sequences we repeat this process and here the last dimension is processed in this particular case we've had 0 on the right hand side and apply the same operation to generate the final output value 1 so we call this sliding window operation a convolution operation so mathematically we denote the input sequence as function ft and the future operation GT and the final output of the convolution F star G T another concept of convolution is straight which is the space between two consecutive future calculation so in this case we apply this future at every dimension in this case straight a go to 2 we do this future operation every 2 element at a time so you can observe the number of parameters for convolution operation are fairly small it's just this size of future here we have to apply this future at every location of the input so this can be computationally very expensive luckily this computation can be efficiently done in parallel especially on modern hardware such as GPU

294 so here is the example of neural network representation of 1d convolution with strike two without zero padding the computation in this case can be represented as multiplication of this weight vector W1 W2 W3 with a subset of input dimensions in this particular case X1 X2 and X3 in this particular case the output of the convolution operation has two dimension the first one h1 has W1 times X1 plus W2 times X2 plus W3 times X3 and the second dimension H2 as applying the same future on X3 X4 and X5

296 so now let's visualize convolutional operations for 1d input which shown on the left you can see that we apply one deconvolution first then oftentimes the output of convolutional layer goes through

some nonlinear transformation such as a rail Euler layer sometimes we have multiple futures not just one for example for this three-dimensional input we have two futures of size four by four then we'll have two output feature Maps in this particular case three by three then we'll put them together concatenate them or think of this as a 3d tensors of size three by three by Q and then we can pass this feature map sure another ReLu function to get the final output feature Maps which is three by three by two so don't worry about those numbers for now we'll explain how do we derive and those numbers and we'll the relationship across different layers next but the key here is to keep in mind oftentimes we do apply multiple futures given the same input so intuitionist each future is trying to extract one type of patterns for example this future one is trying to extract average value in the image and the future two is trying to extract some line in this image and so on so it futures corresponding to a single patterns and when you have many different patterns you want to extract you probably want to have many different futures

297 so pulling operation is another common operation in convolution your network and it can be considered as smart non sample strategy for example in this case one the future with stride 2 so we will get a two-dimensional output from this four dimensional input similarly we can apply pooling on two-dimensional input as well so in this case we'll specify a pooling layer with a future Q by 2 with stride 2 then we can apply this 2 by 2 window futures a different location of this input feature map then to generate different type of output feature mapping for example this max pooling operation would take the maximum value in this each filter and just keep that maximum value for example in this particular case we'll generate 4 5 6 & 5 from the original feature Maps of course we can use different pooling operations such as taking the sum or taking mean or average but in general max pooling seems to work really well and it's probably the most popular choices so far

298 now let's understand the relationship between input output size of a convolutional neural network so this is the general case where we'll take input in this case a three dimensional volume of size W1 times H1 times D1 and in this example we have a kind of a color image of size 227 times two hundred twenty seven times three right there's three different channels then for the convolution layers we have a bunch of hyper parameters number futures K spatial extents this is the size of the future things and the stride s and the padding P so in here for example we have 96 futures of k equal to 96 and the spatial extent is 11 so the future size is 11 by 11 and the strike s4 and the output feature map will be a volume of size W2 times H2 times D2 and they can be calculated with this formula so W2 you go to W1 - spatial extent F plus 2 times of the padding divided by the stride plus 1 and similarly H2 you go to H1 - spatial extent F and plus 2 times F 2 padding divided by its strike plus 1 and D2 equals a number of futures so in this particular case you can see giving this convolutional layer and the output feature map is actually of size 55 times 55 times 96 in this particular example actually I didn't specify the number of padding's here but given the output size you can probably figure out what this padding size

299 next let's look at the dimension of the pooling layers so again the general case showing on the right if we take an input volume of size W1 times H times D1 then the hyper parameter here or cooling layer is just the spatial extent F and the stripe s and the output is another feature map of the size W2 H2 times D2 so here it's actually very similar because we don't have the padding anymore and the W2 is just W1 minus F divided by s plus 1 H2 equal to H1 minus F divided by s plus 1 and D2 equal to D1 so after pooling the depths actually remains the same so if we look at this particular example you can see that given this input feature map of 55 by 55 by 96 4 if we apply this 3 by 3 futures doing max pooling with try to we can get this final answer which 27 times 27 times 96

300 so with this you can now read a convolutional neural network architecture like this and now you can understand that what are those numbers means right there we have first layers which is convolution layer with 96 futures with size 11 by 11 with strike four then followed by a pooling layer of three by three futures with strike two followed by another convolution layer with 256 futures of size 11 by 11 and so on and the important part you should understand is the depths of the network often communists become deeper and deeper and majority of the layers about convolution or pooling x' and a few layers at the end are those fully connected layers if you do a simple calculation of the parameters you actually will realize this convolution layers despite the fact there are many more of those layers then the fully connected layers the number of parameters actually relatively much smaller than the parameter in the final few fully connected layers so in this particular case we have 3.7 million parameters in this all disk emotional layers and for this last three fully connected layers we have 58.6 meaning parameter a lot more so that's kind of the intuition you should have that for a convolutional neural networks the parameter is actually not that many and most parameter actually are still coming from this fully connected layers

301 now let's look at the calculation using convolutional neural network so if we look at the forward calculation of convolutional layer if you look at the general case with the input of size w1 times H1 times d1 and here is a parameter for this convolution layers and the number of calculation you need to do are actually a lot which is really the output size this w2 times H2 times T2 and for every output elements you have to apply this future so this future is F times F times the depth D1 so that's kind of the roughly the number of calculation you need when you apply convolution later so you can use this example to figure out kind of the number of calculations and the size of the output by the way we already discussed previously and I'm just listing the formula here for your convenience

302 and similarly we can't figure out the number of forward calculation for the pooling layer which is pretty much of the same spirit Thanksgiving this input size and the parameter for this pooling layers spatially extend, and stride and we can see okay here's the number of calculation which is again the size of the output plus the future size

303 finally if we look at the fully connected layers the number of calculation is actually quite straightforward it's just the size of the input to this fully connected layer when you and I've vectorized this 3d tensors into a vectors times the output dimensions in this case 4096-dimension times whatever this number is so that will give you the number of calculations you need to do for this fully connected layers in this particular case as the output size T times the input size

305 next let's look at some CNN architectures and then look at some of the healthcare applications so Alex Ned is the first color chenille network that shown great performance gain in an image classification context they won the competition around the year 2010 and by improving the performance by a huge margin in fact this is more or less darker texture we explained in our previous example but they just draw this in a more kind of intuitive structures show the size of the feature map and the corresponding future says Inception that is another very powerful convolutional neural network architectures you can see that every little box here is a layer of easier convolution layer or pooling layer or fully connected layer and so on so you can see that the key differences is first of all it's much deeper many more layers than before second they introduced some kind of this parallel path right so when you go from here to here you actually go either one convolution layer or three of war or average pooling followed by another convolution layer then you concatenate them right you can you can see there as many of those in a parallel pass happening over again later on there is another very powerful convolution

neural networks called residual networks and the key idea is in addition to you all this convolution layers now they also introduced this so-called skip connection right so allowed the input go to the output directly so the integration is when you go through many of those layers of calculation maybe one of the layer in the middle it's already very good features right you want to actually keep the feature towards the end how do we do that if we introduce the skip connection actually allowed the model to great out that okay if this layer is already giving up very good features then all escape connection can still maintain this intermediate and this powerful feature and use it at the end so essentially it actually alleviate some of this vanishing greeting problem because the skip connection essentially reduced the depths of the network and also it combines both the shallow and deep networks thanks to this skip connections if we look at the performance of all the CNN architectures for some real work tasks like ImageNet classification performance here we're showing error rate so the lower the better ways in the past with warner network the best winning performance it's about 25 percent errors then when Alex net introduced in 2012 and error rate reduced to 16 percent then vgg network further reduced the errors to 7 percent and then lately the residual network rest net only produced less than 4 percent error which is actually a human level performance already and in this particular case the number of layers 152 is truly deep neural networks

306 so now let's talk about house care application of CNN algorithms so in this particular paper it's about detecting diabetes representing obviously using CNN algorithms it's published at JAMA Journal of American Medical Association one of the top medical journal so the idea is given the image of the eyes like this can the algorithm figure out which one is a health individual which one is the one with diabetes Rep democracy so this is currently done manually by doctors and thanks to deep learning in this particular case CNN models the algorithm actually can perform to the level of domain experts so in this particular result curve you can see that so this ROC curve is really good and all those dots are actually human experts you can see the algorithm which is this line it's actually performed as good as human doctors

307 so there is another paper called dermatology level classification of skin cancer with deep neural networks and in this particular case it's published at Nature so the goal of this work is to look at actual photograph of skin like this trying to decide whether it's benign or is malignant again the use CNN models achieve human-level performance in this particular case they're showing for three different type of images for different cancers you can see this blue curve is the algorithms and the red dots are the actual domain experts you can see that in many cases this blue curve is above the red dots in particular case this Green Cross indicates average performance of those doctors you can see they are actually entered this blue curve meaning that the algorithm is actually better than the average domain experts in this particular case