CSE 6250 Lecture Notes

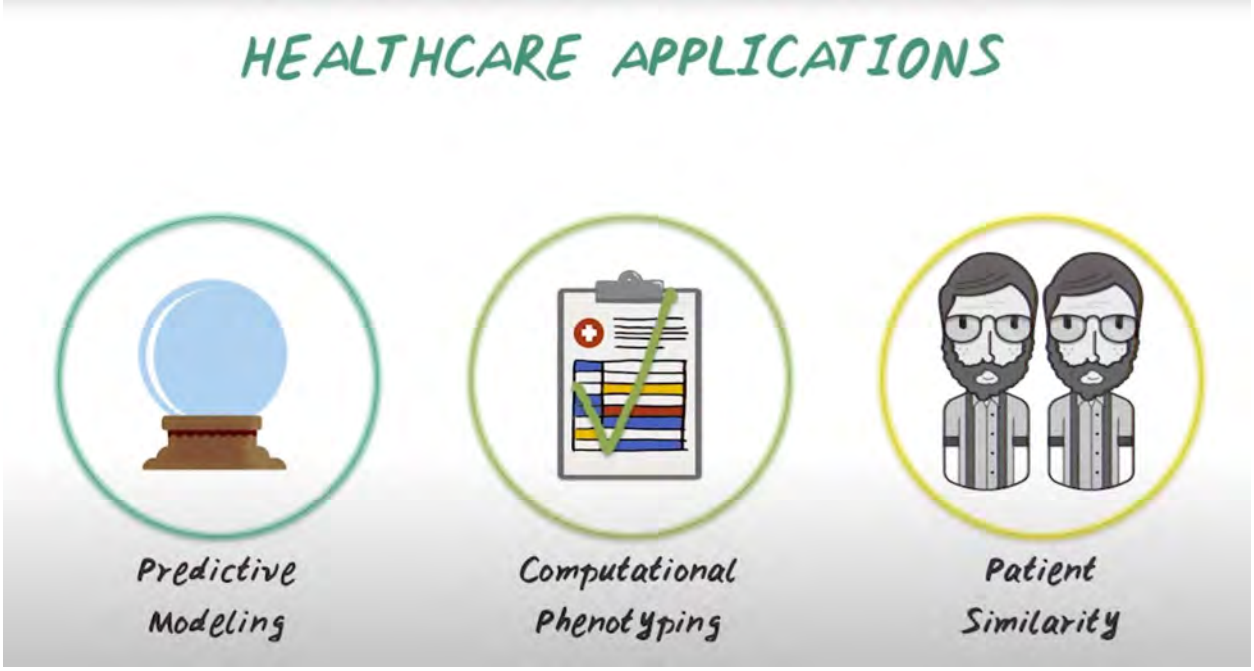
Feel free to make changes/updates to this to fill out gaps in it.

Good luck everyone!

Introduction to Big Data and Course Overview

The Four V's

- Volume - Massive amount of data for algorithms/systems to act on
    - Ex: Each genome requires 200 GB of raw data, a single fMRI is 300 gigabytes
- Variety - Many different types of data (Sensors, video, tweets, fitbit, etc…)
    - Clinical information, patient diagnosis, clinical notes, on body sensors, etc...
- Velocity - Data coming in real time, needs to b e processed/analyzed in real time
    - Blood pressure measures, heart rate, drug dispensing sensors
- Veracity - A lot of missing data, noise, false alarms, errors



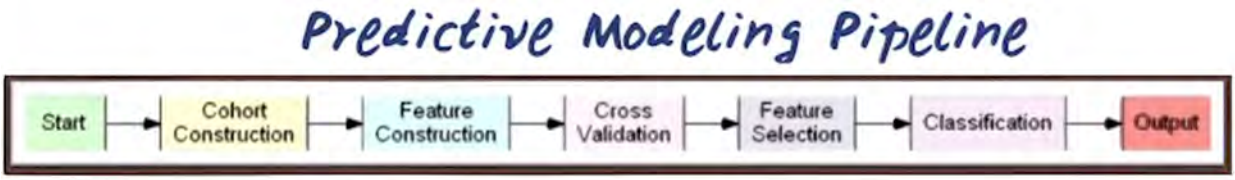
Predictive Modeling: using historical data to view the model for predicting future outcome.

Computational Phenotyping: turning messy electronic health records into meaningful clinical concepts.

Patient Similarity: uses health data to identify groups of patients sharing similar characteristics.

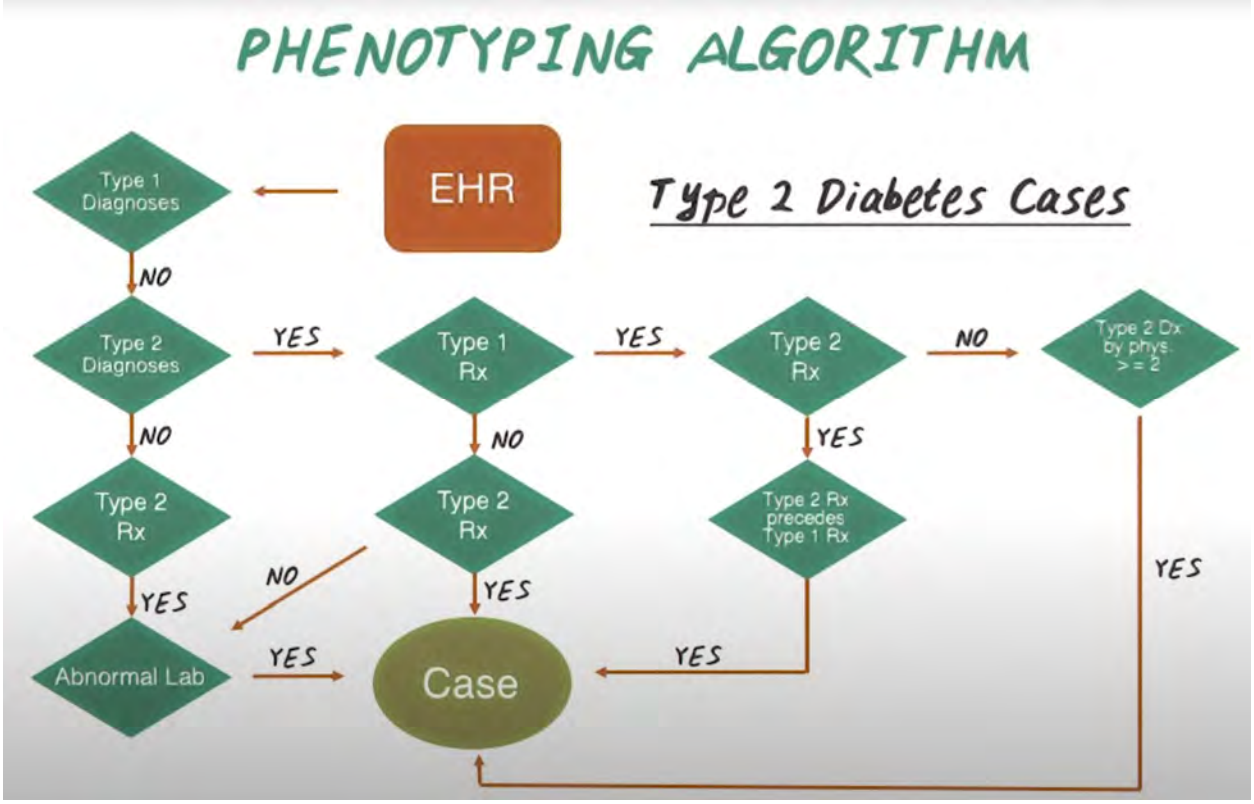
Predictive Modeling Challenges

1. So much data!
2. So many models!
3. So many pipelines! (each step can be different)



Computational Phenotyping: Input is raw patient data such as demographic, procedure, diagnosis, lab tests, medication, clinical notes. Computational phenotyping is turning the raw data into medical concepts (Phenotypes)
Some of the waste products are: Missing data, duplicates, irrelevant data, redundant data

Example: Phenotyping Algorithm for Type 2 Diabetes Cases



Will learn how to create the above in class

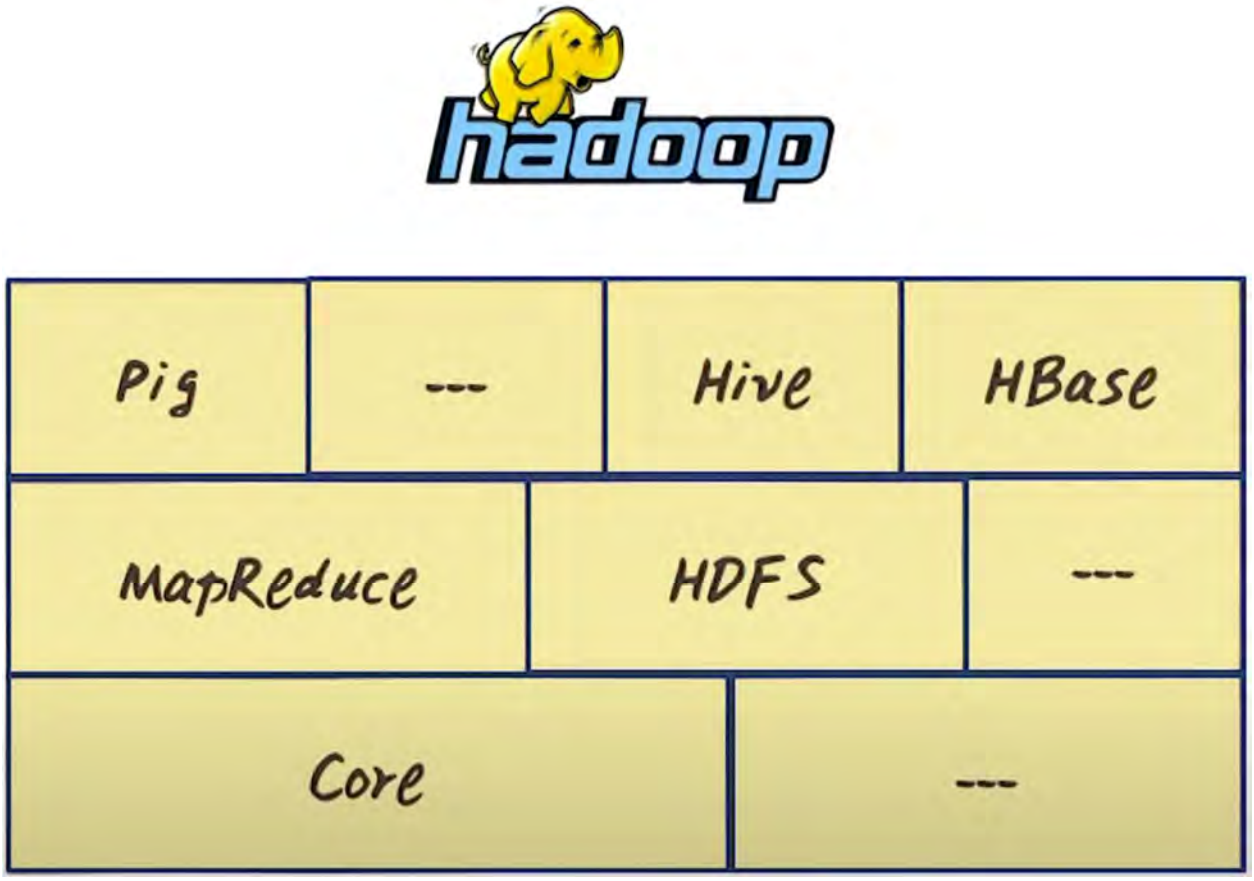
Patient Similarity:
Simulating the doctor's case based knowledge with a computer algorithm. Using more than a single doctor's memory/knowledge but a large dataset of patients. Idea is a patient comes, doctor does an exam, and searches the database, supervising the output to find patients that are truly similar and recommending a treatment.

Big Data Algorithms:

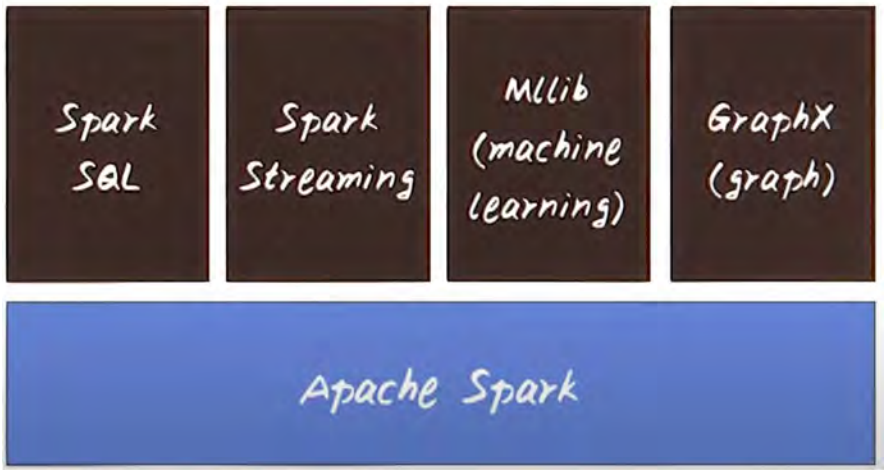
- Classification - Learn a function f that maps subject data x to output y
- Clustering - Matrix x learning function f that clusters data into patient clusters which are similar to each other
- Dimensionality Reduction - Input large matrix x and reduce down to smaller matrix x' with fewer features, just the relevant ones or new features
- Graph Analysis - connect patient diseases related to each other in a network and how related

Systems:

- Hadoop - Distributed disk-based big data system

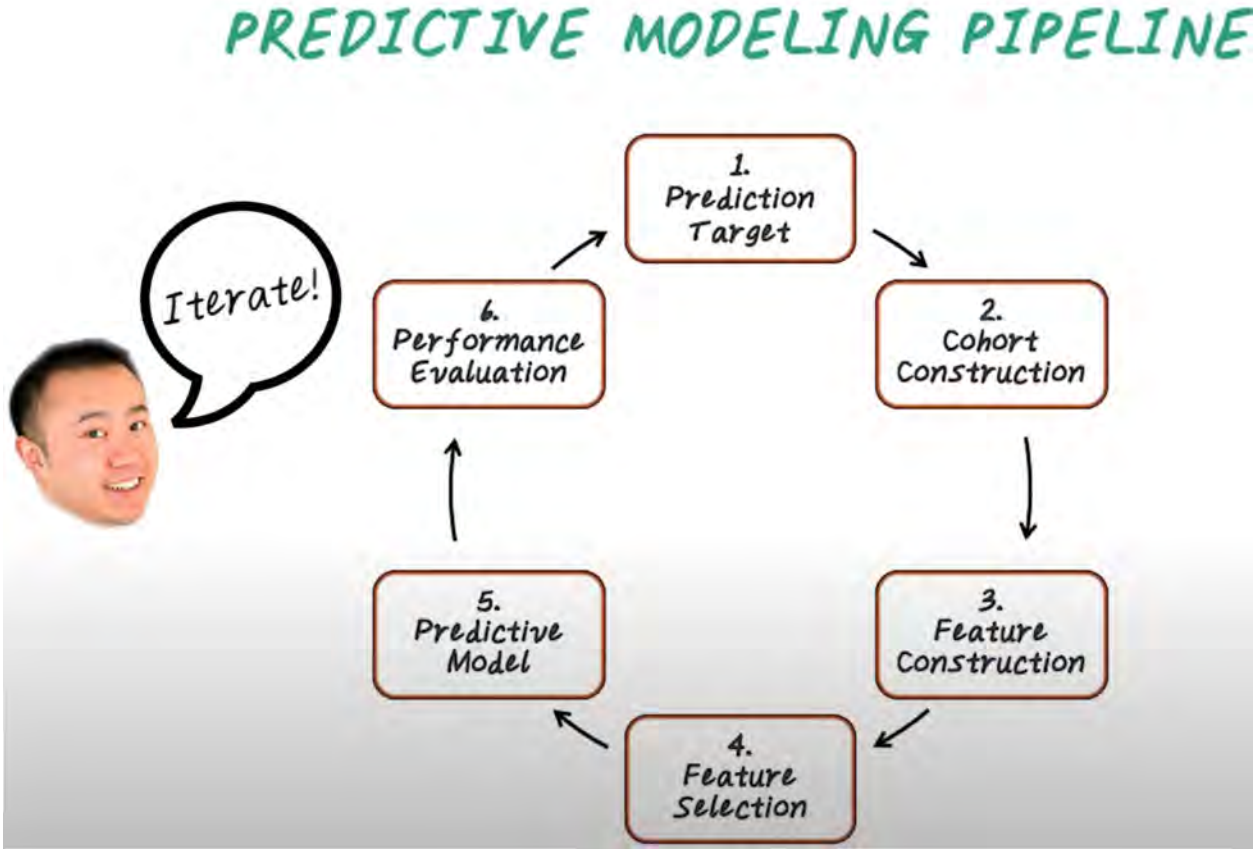


- Spark - Distributed in-memory big data system

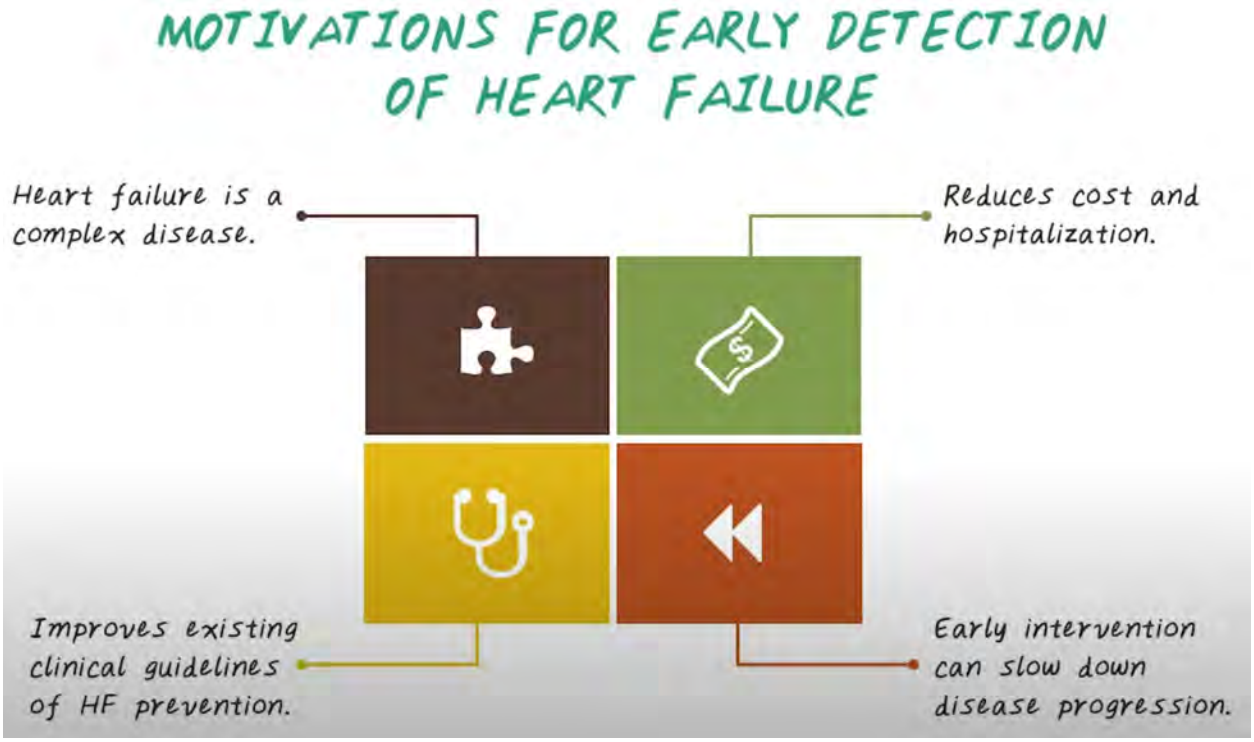


Predictive Modeling

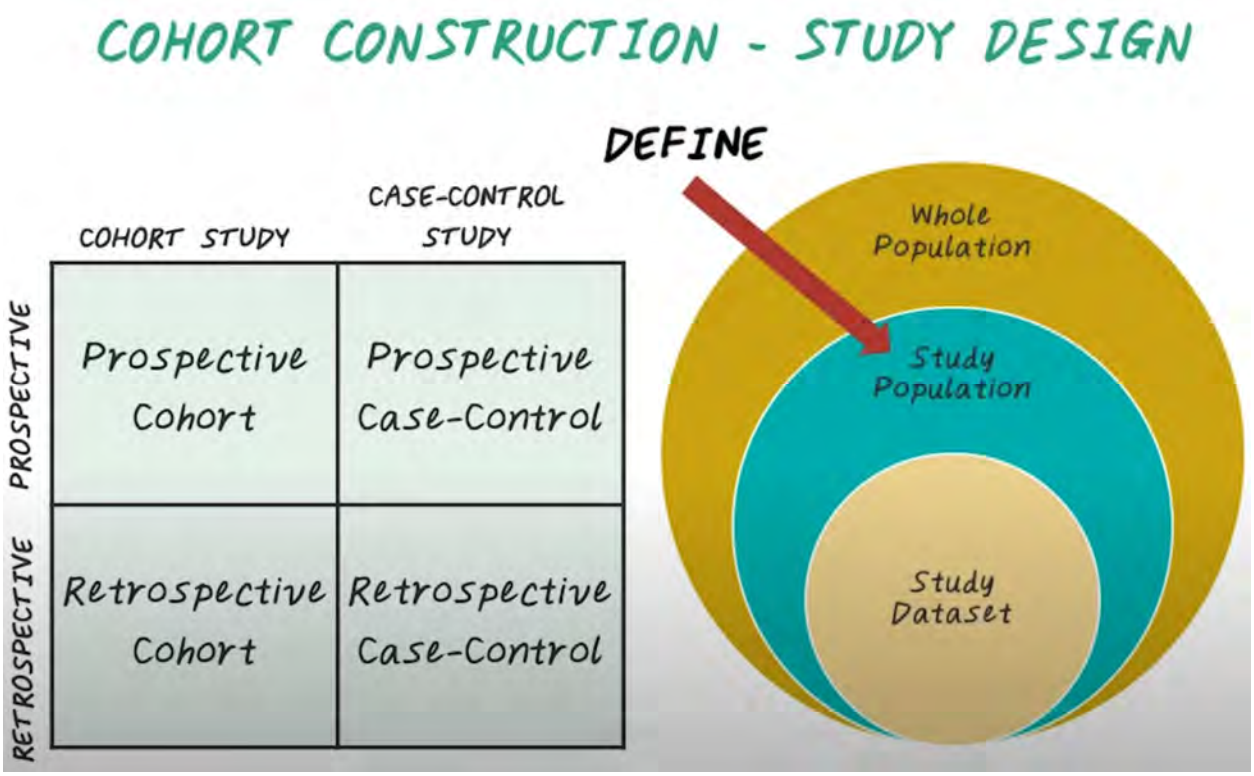
How do we develop a good predictive model quickly?

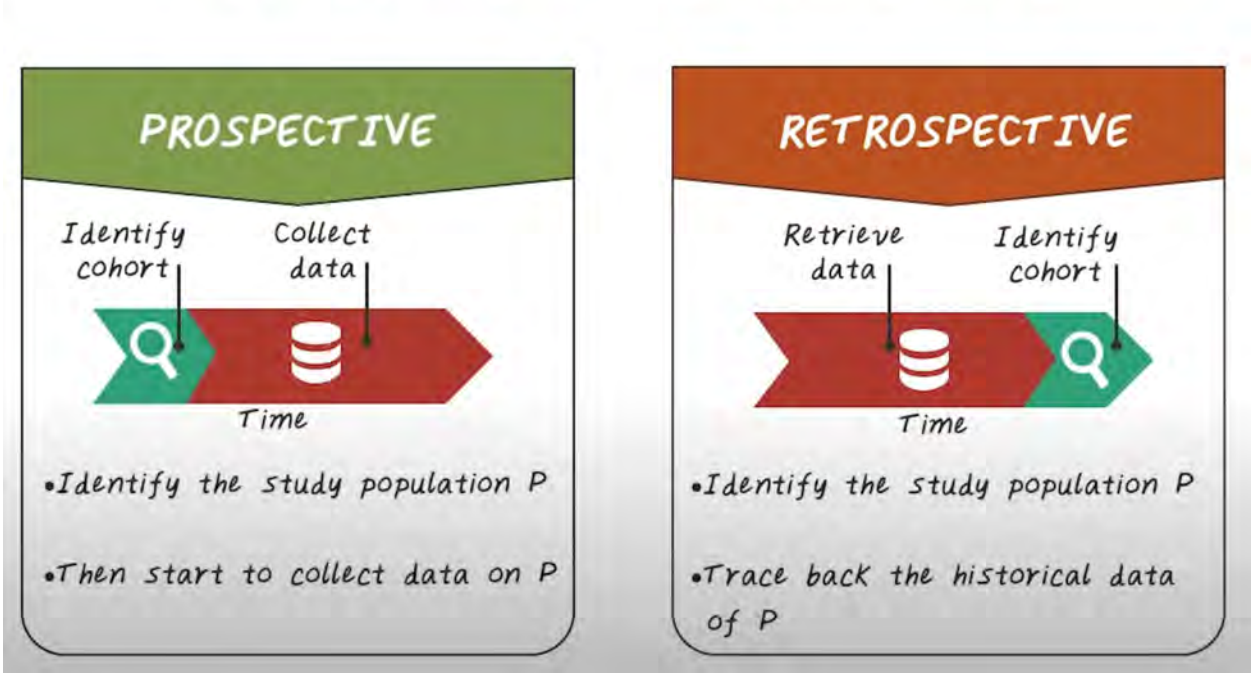


1. **Prediction Targe**t: There's many things which an investigator might find *interesting* to predict but only some are *possible*. An investigator should focus on what is both *interesting* and *possible*



2. **Cohort Construction**: Defining the study population. Given the whole population, only a subset is relevant. Usually only a fraction of the study population is possible to get a study dataset





Prospective: Identify study population, then collect data

- Quality of the data is usually higher as the data is designed specifically for the study
- Typically more expensive and takes more time to conduct as data needs to be collected from scratch
- Data is often smaller and more limited as it is more focused and more expensive/time consuming

Retrospective: Data already exists, identify study population and then find the data

- The data of the retrospective study typically has more noise as it was created for a purpose other than the study it is now being used for
- Typically cheaper as the data already exists and less time as you don't need to collect the data
- Dataset is often larger as it requires more data to get good data and it is

Property	Prospective Study	Retrospective Study
More noise in the data	☐	☑
More expensive	☑	☐
Takes a longer time	☑	☐
Common on large dataset	☐	☑

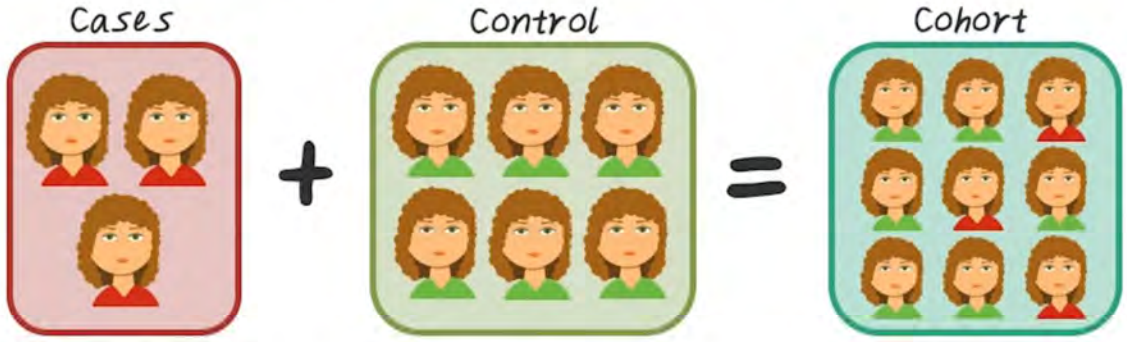
COHORT STUDY

Select a group of patients who are exposed to the risk

TARGET: Heart Failure Readmission

- COHORT: all HF patients discharged from hospital
- KEY: define the right inclusion/exclusion criteria

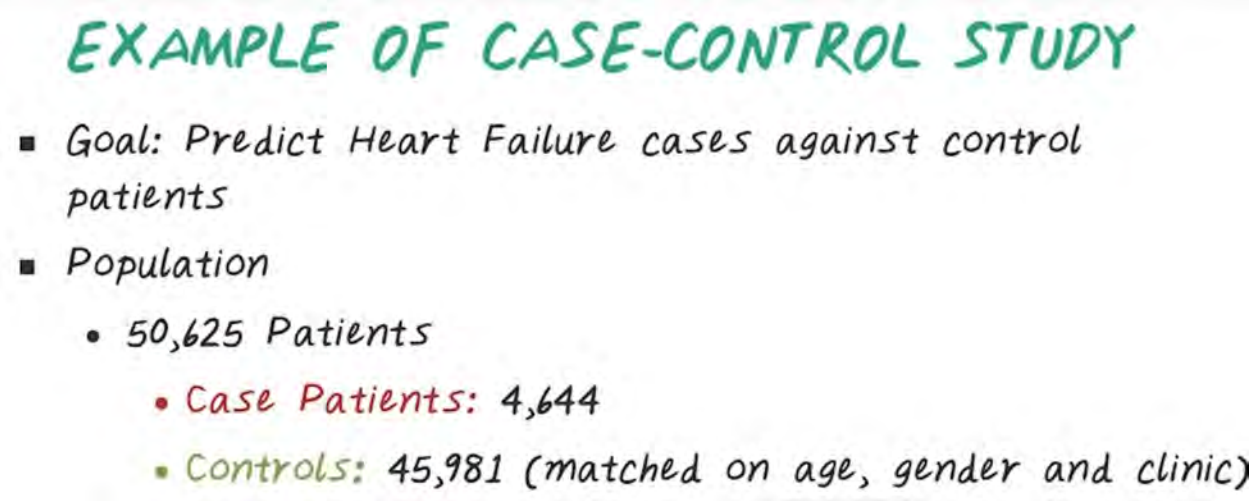
CASE-CONTROL STUDY



CASES: patients with positive outcome (have the disease)

CONTROLS: patients with negative outcome (healthy)
but otherwise similar

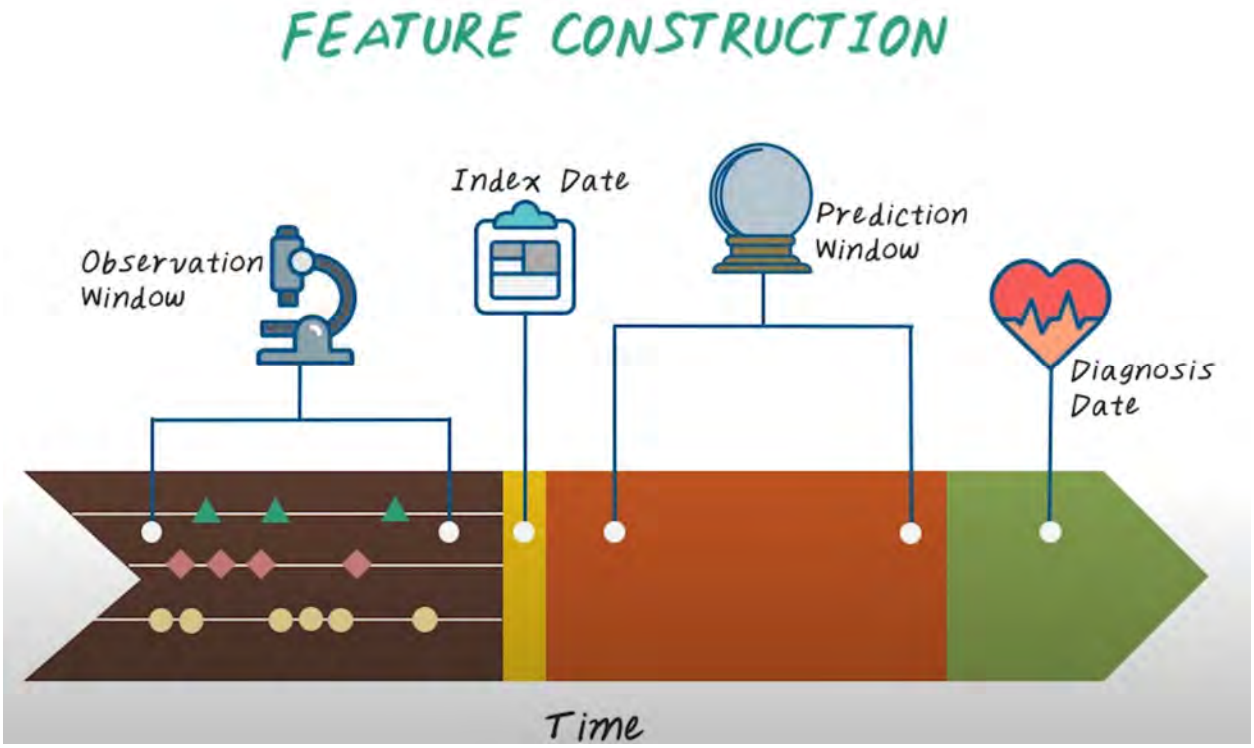
KEY: matching criteria between cases and controls



Very typical the cases population is far smaller than the control population, as usually the disease we are searching for is harder to find in patients than healthy similar patients

To summarize, in a case-control study, we first identify the cases, then try to match them to a set of control patients.

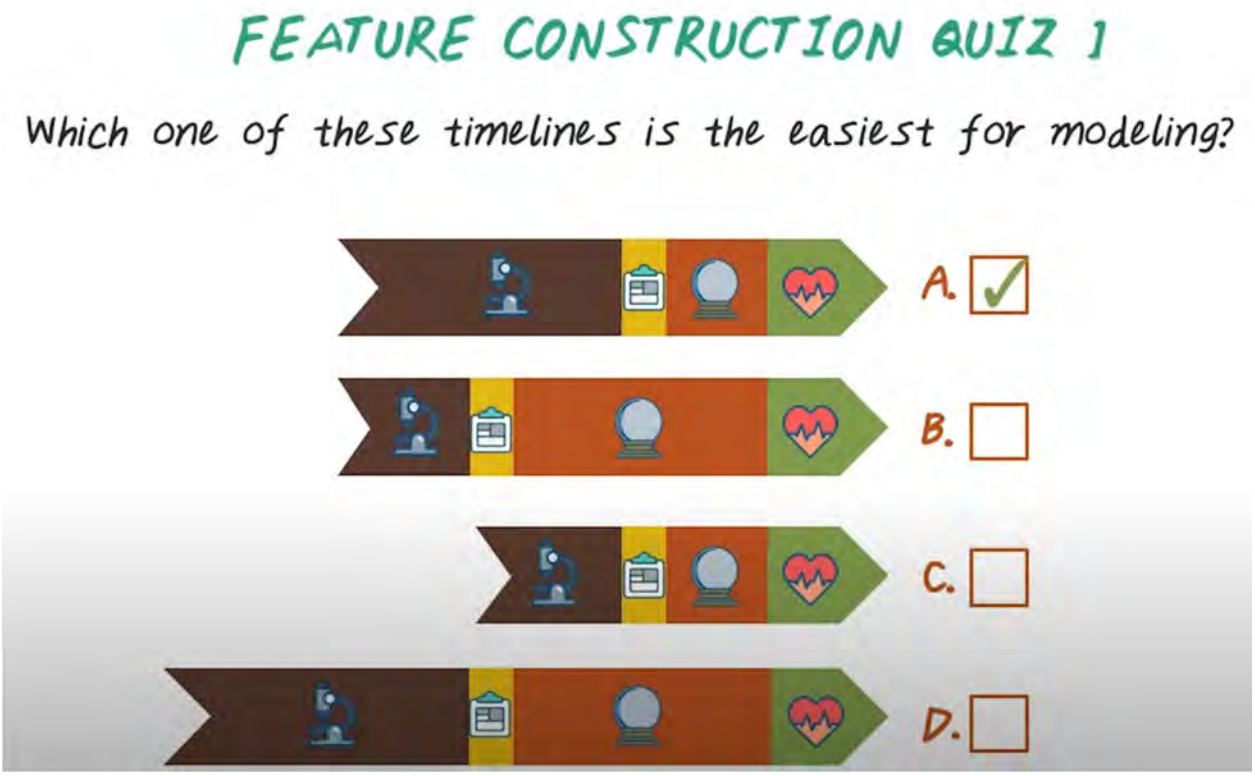
In a cohort study, we'll identify all the patients who are exposed to the risk and the matching criterias are not involved.



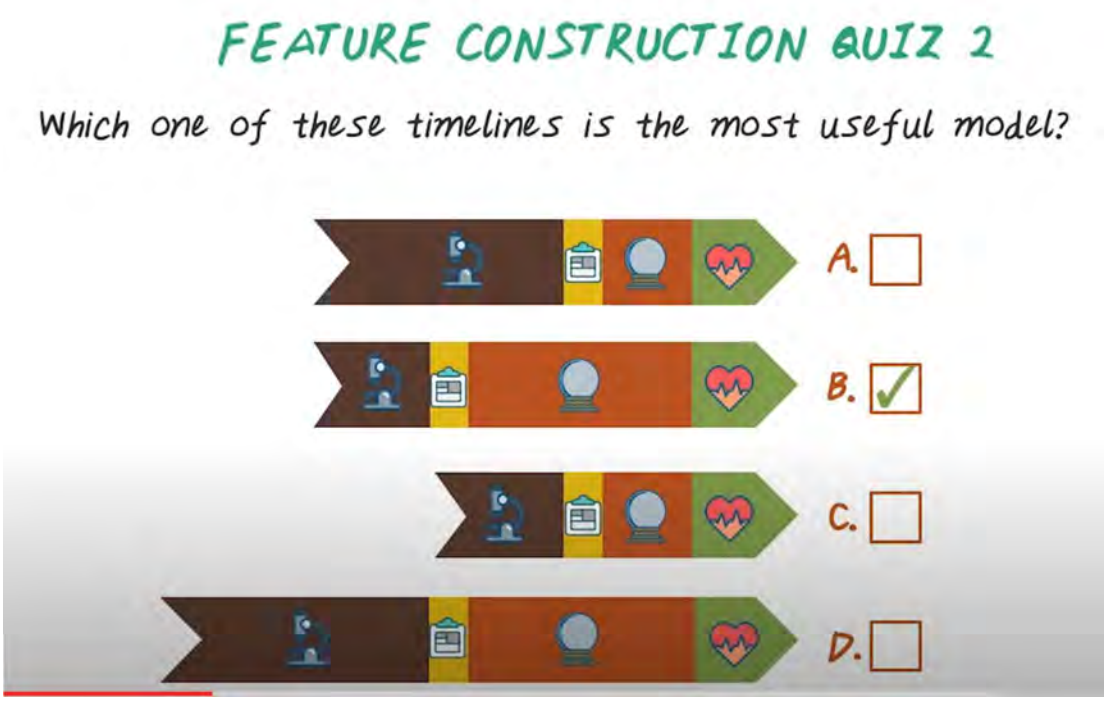
Construct features from data in the observation window
There are many different ways to construct features. For instance, we can count the number of times an event happens. For example, if type two diabetes code happened three times during this observation window, the corresponding feature for type two diabetes equals three.

Or sometimes we can take the average of the even value. For example, if a patient has two HBA1C measures during the observation window, we can take the average of these two measurements as a feature for HBA1C.

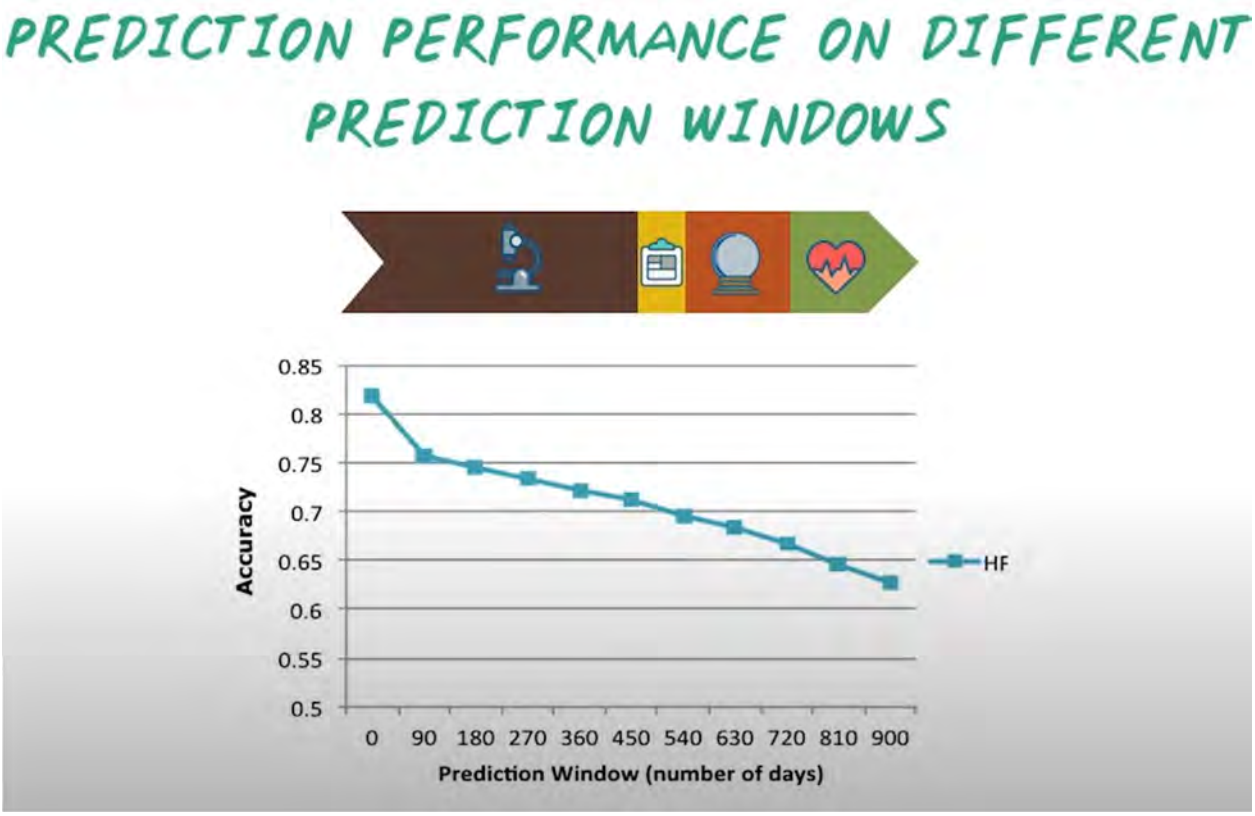
The length of the prediction window and observation window are two important parameters that going to impact the model performance.



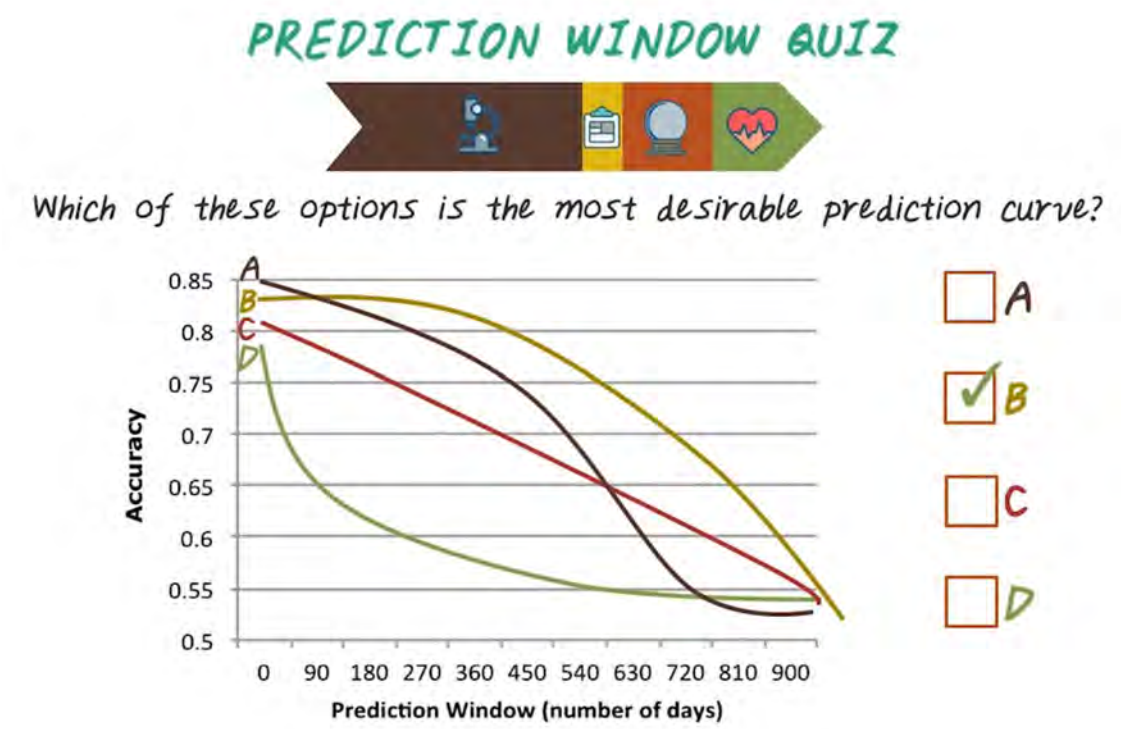
Easier to predict events in the near future with lots of data



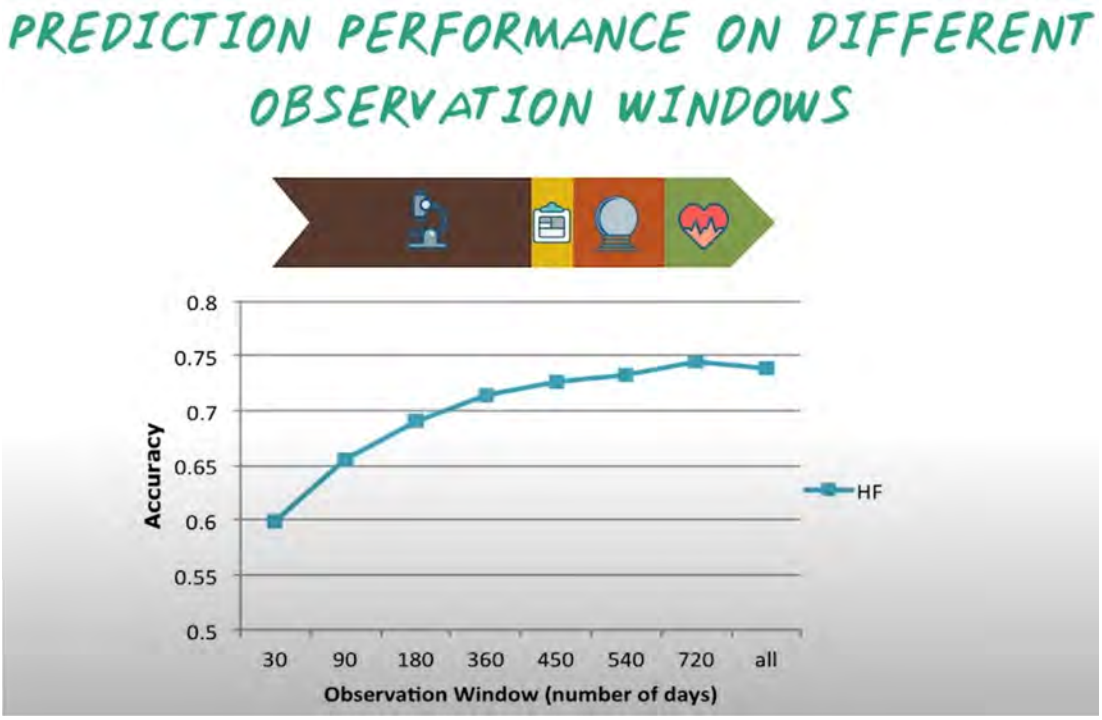
Ideal is to observe a very small timeline, but predict far into the future. But is usually unrealistic and much harder to accomplish



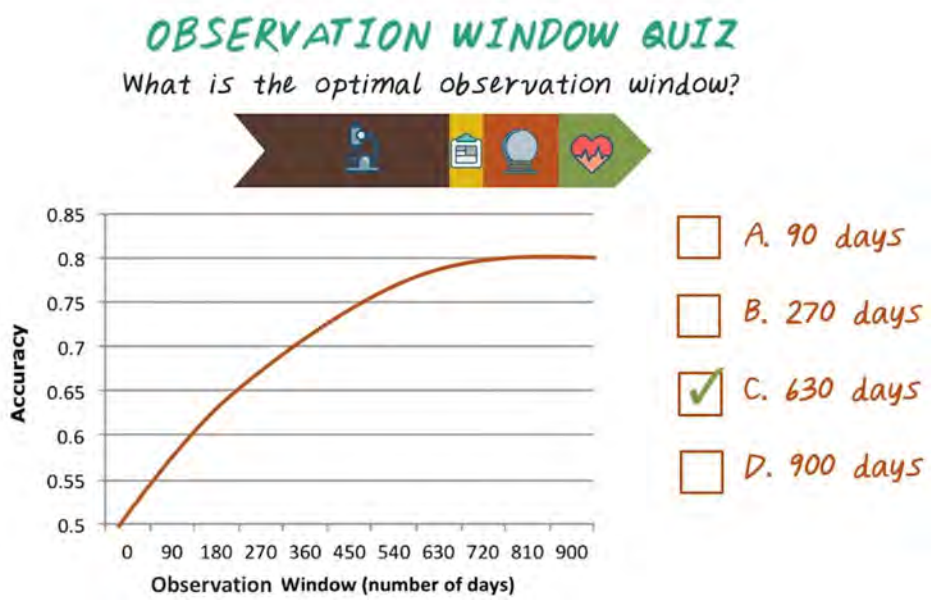
Typically prediction accuracy drops the longer the prediction window is



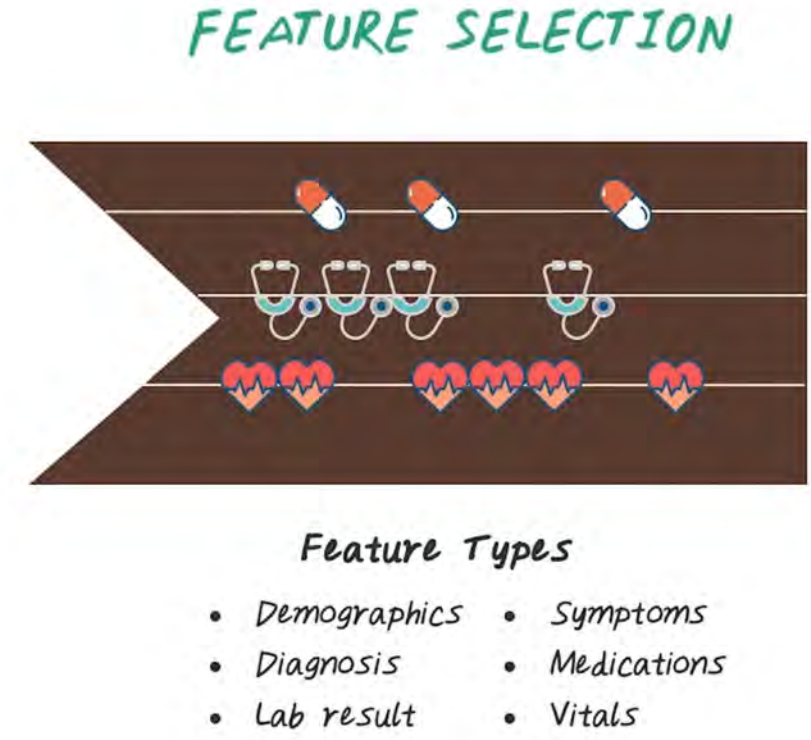
More useful to have longer high accuracy than initial very high accuracy that drops off quickly



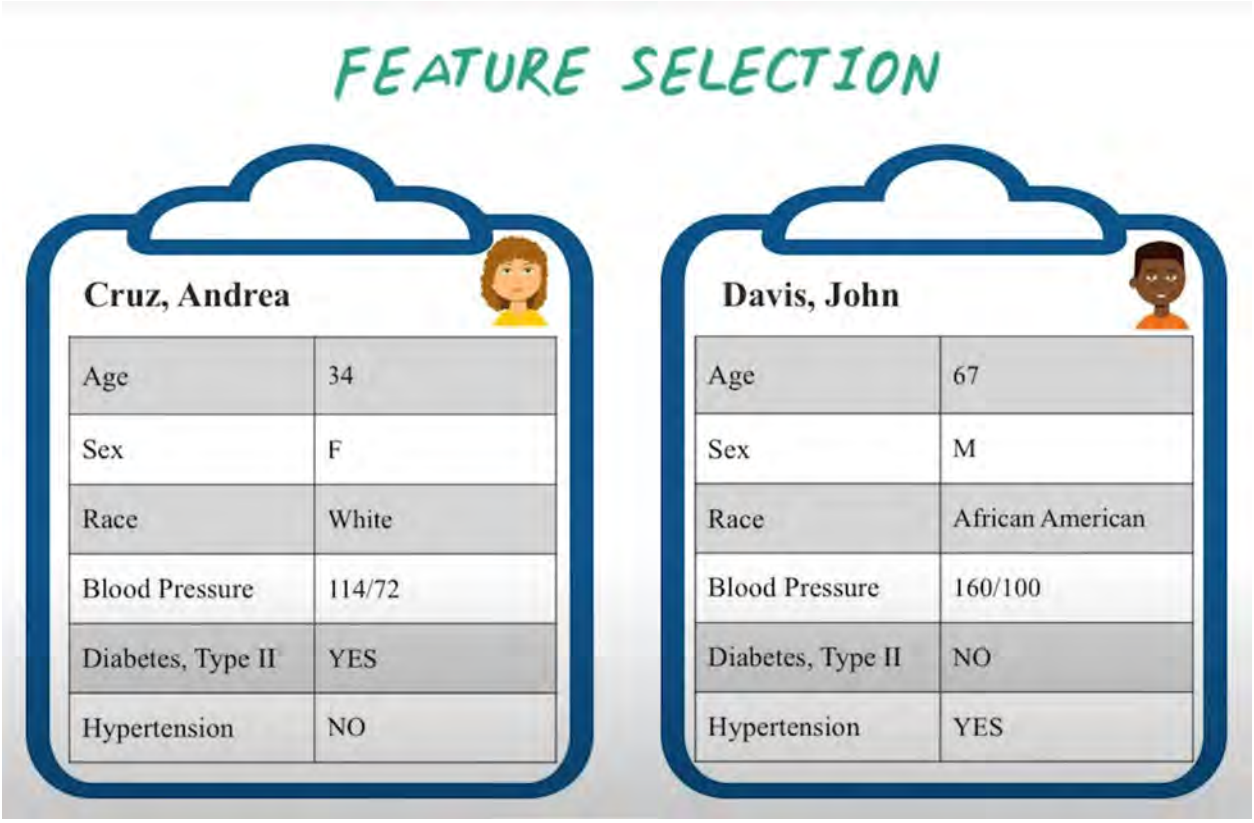
Typically accuracy increases the longer the observation window is as we learn more about the patient



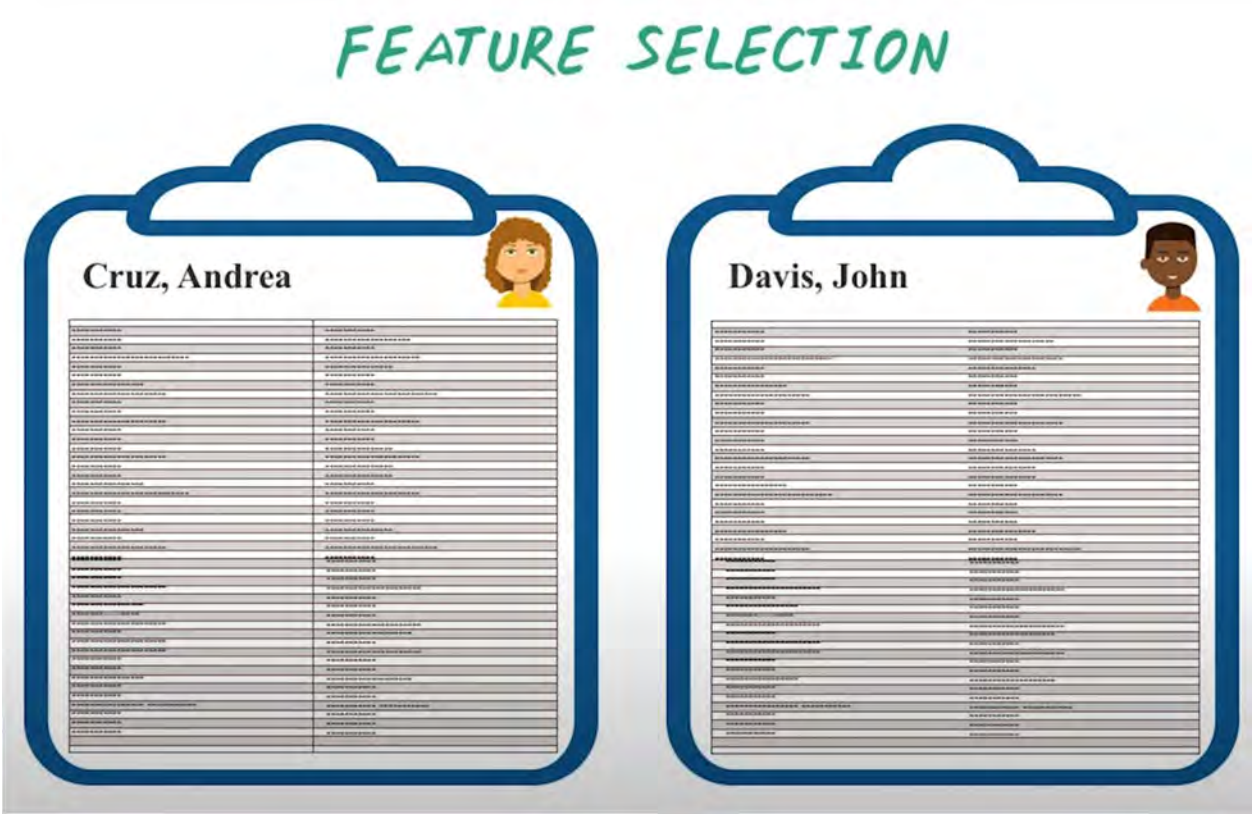
Diminishing returns after 630 days. We want maximum accuracy, but not to wait too long, it might exclude patients that can't be observed that long



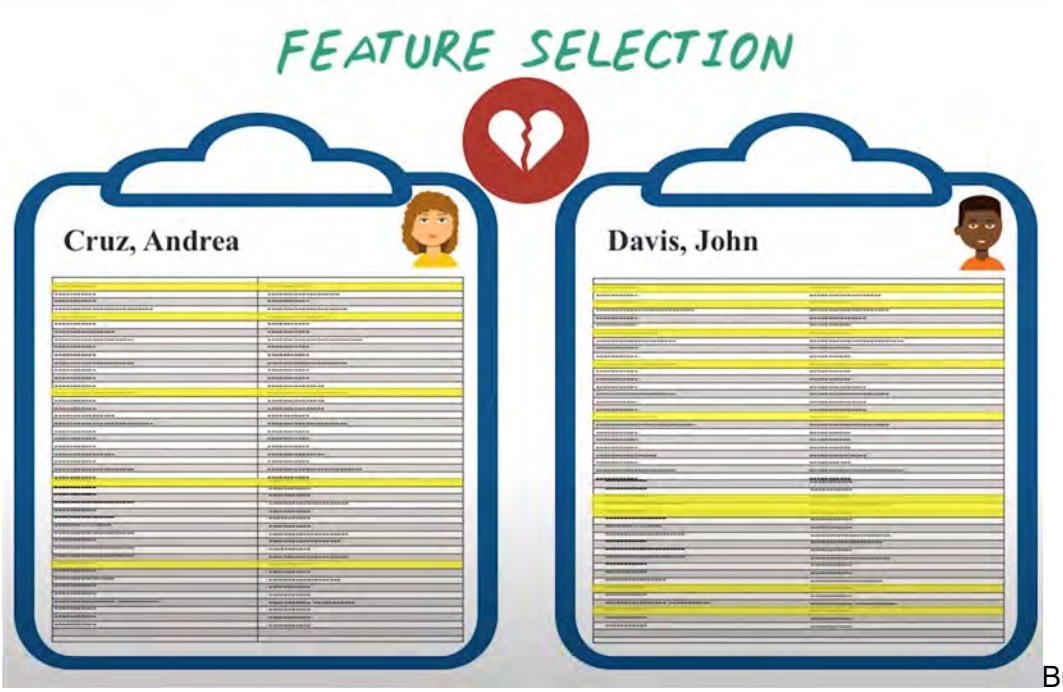
We can construct features from all the events, but not all features are relevant. The goal of feature selection is to find the truly relevant features to use for prediction



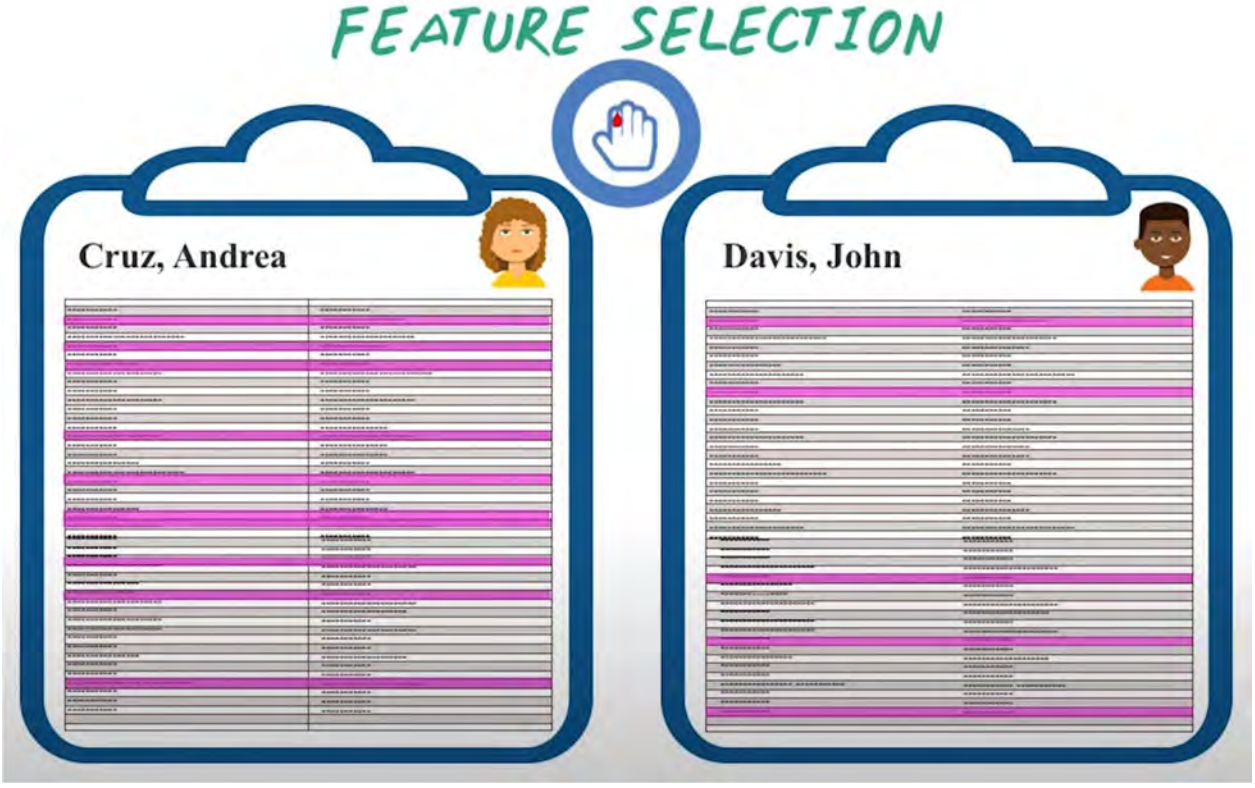
Small list of relevant features comes from a large dataset of a huge number of features



Heart Failure uses some (yellow)



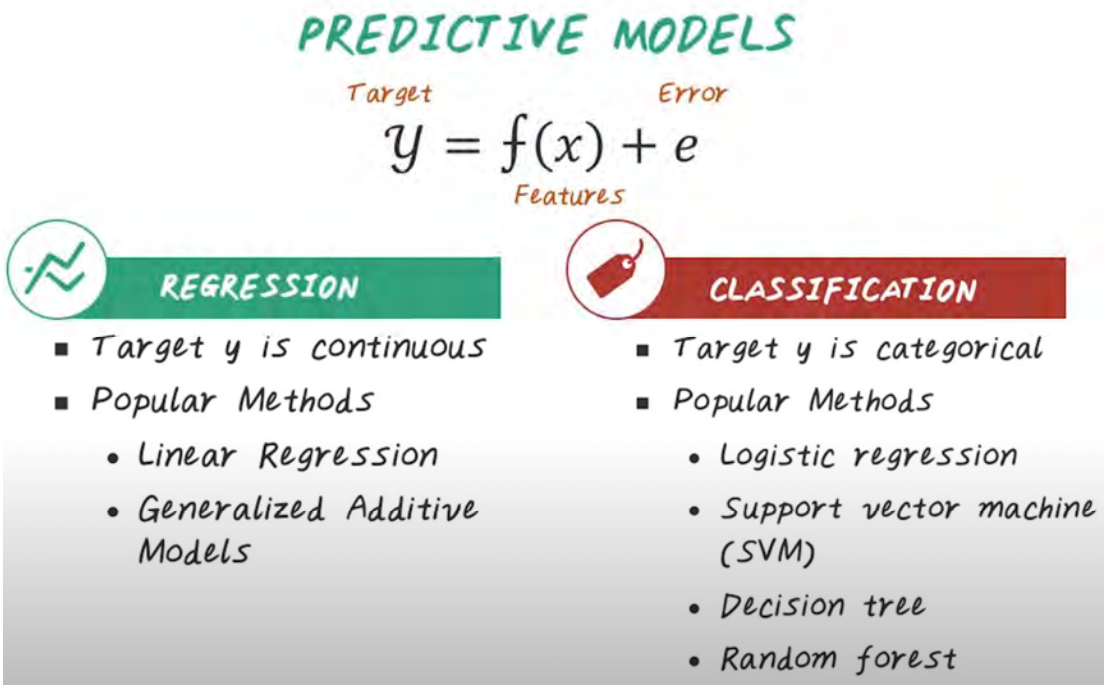
But diabetes has different relevant features (pink)



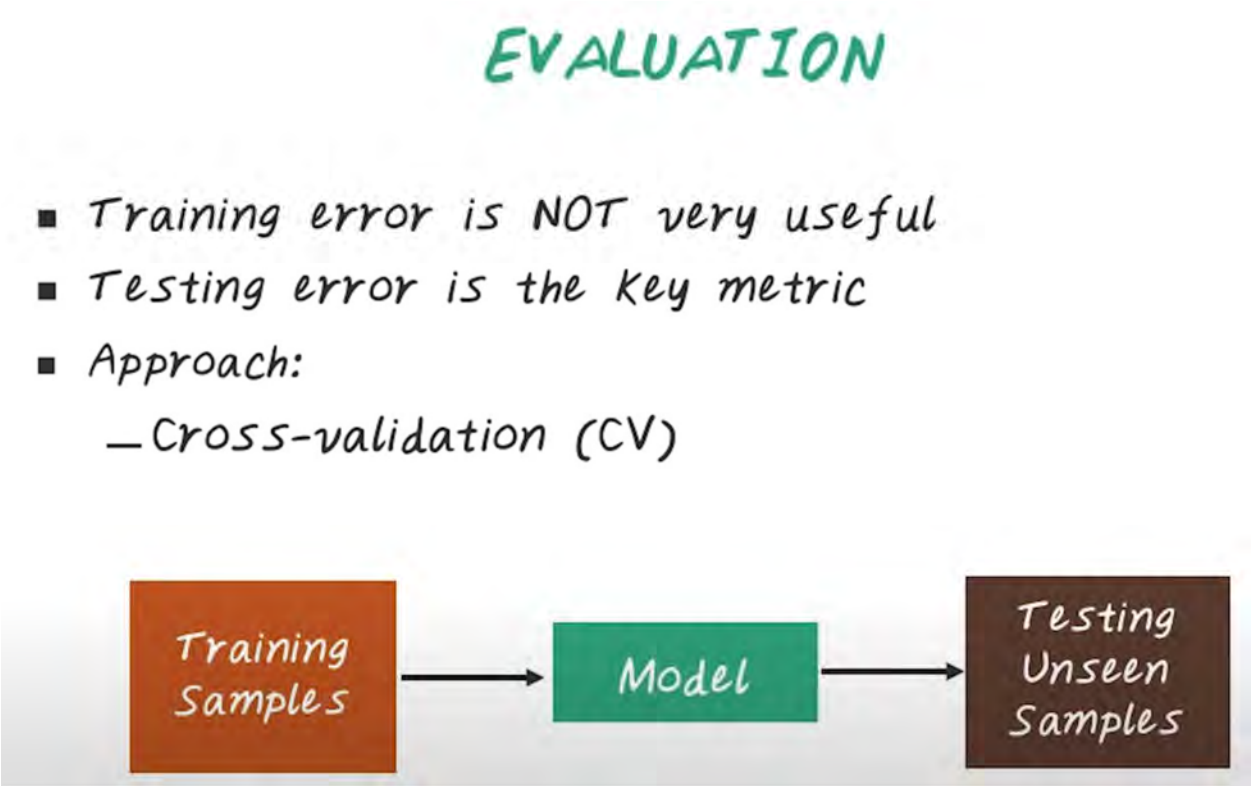
Goal of feature selection is to identify the relevant ones for each prediction target

Predictive Model

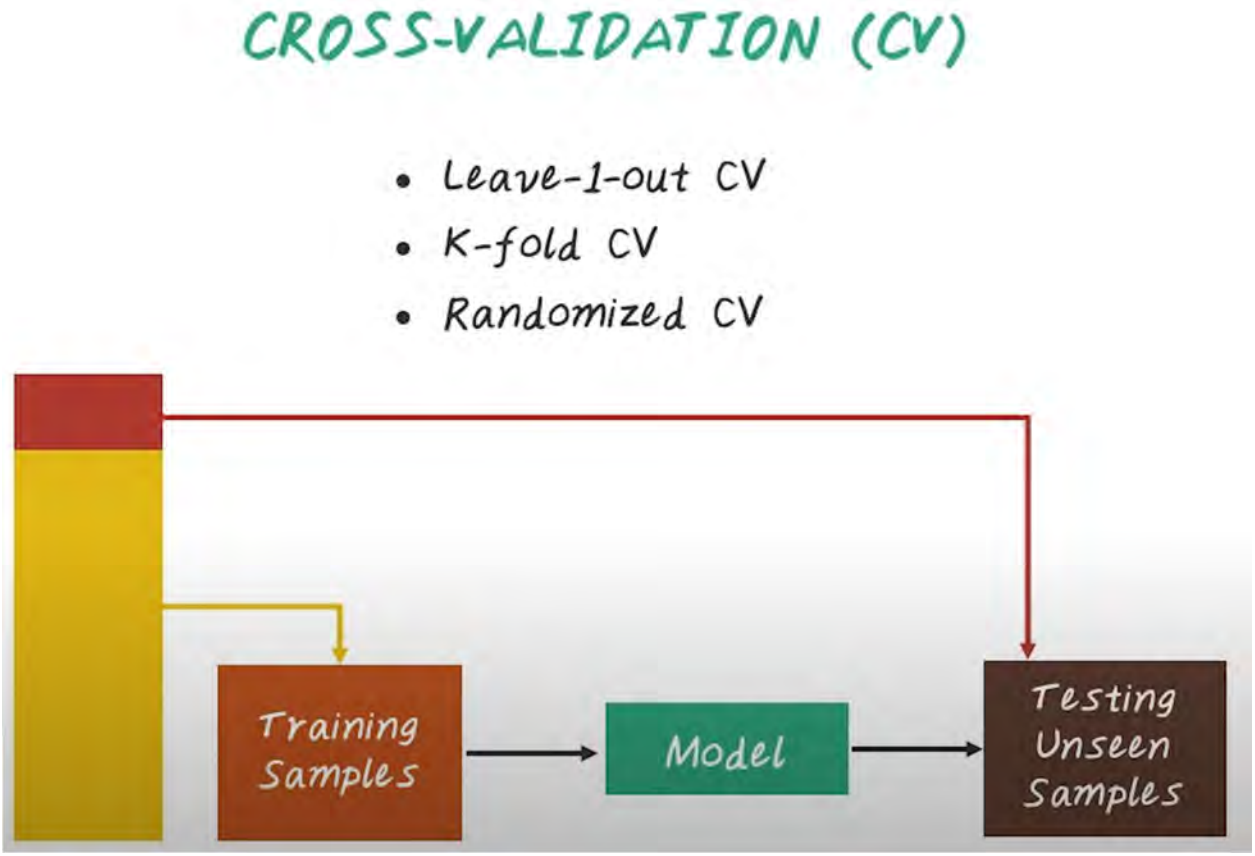
Function that maps input features *x* to target *y*. Depending on the target it can be a regression or classification problem



Performance Evaluation



Develop the model using the training samples, test it on test data, ideally from the future. Just relying on the training error does not generalize well, test error is more realistic. Common approach is cross validation



Leave 1-out CV: Take one example at a time, train the model on the rest of the data, test it and then average. Go through the entire dataset.
K-Fold CV: Very similar to leave 1 -out but break into k-folds
Randomized CV: Randomly split the dataset into test/train sets. Advantage is the proportion of train vs test is not dependent on the k-fold. However it can be imbalanced

MapReduce

What is hadoop MapReduce?

- A programming Model
- An Execution environment
- A software package

It provides Distributed Storage, Distributed Computation and Fault Tolerance

Started as a Google proprietary software to support the search engine. Written in Java. Uses a very limited but powerful programming paradigm in order to support parallel processing on large datasets, allows super scalable computation

Learning Via Aggregation Statistics.
Present a Machine Learning Algorithm as a Mapped function *f* (e.g. list of risk factors) that can be applied on a large data set and is returned as a Reduced function (e.g. performing aggregation statistics)

Suppose a large database of patients, each patient has a record with all their details. Example to get all the disease mentions in a patient record

```
Map(patientRecord)
{
      Disease_list = find_disease_mentions(patientRecord)
      for (disease in disease_list)
      {
            emit(disease, 1)
      }
}
```

This will output a key, value pair for all patients, e.g. (for patient 1 (Hypertension, 1) and (Diabetes, 1) and for patient 2 (Heart Disease, 1) and (Hypertension, 1))

A Reduce function will combine all the data for all patients

```
Reduce(disease, counts)
{
      emit(disease, sum(counts))
}
```

MapReduce System
Why is it this way? Usually data is too big to be stored in one system, it's spread over multiple databases. Each Mapper function will prepare data for each reducer function on their respective databases, to be combined in a final reduce function.

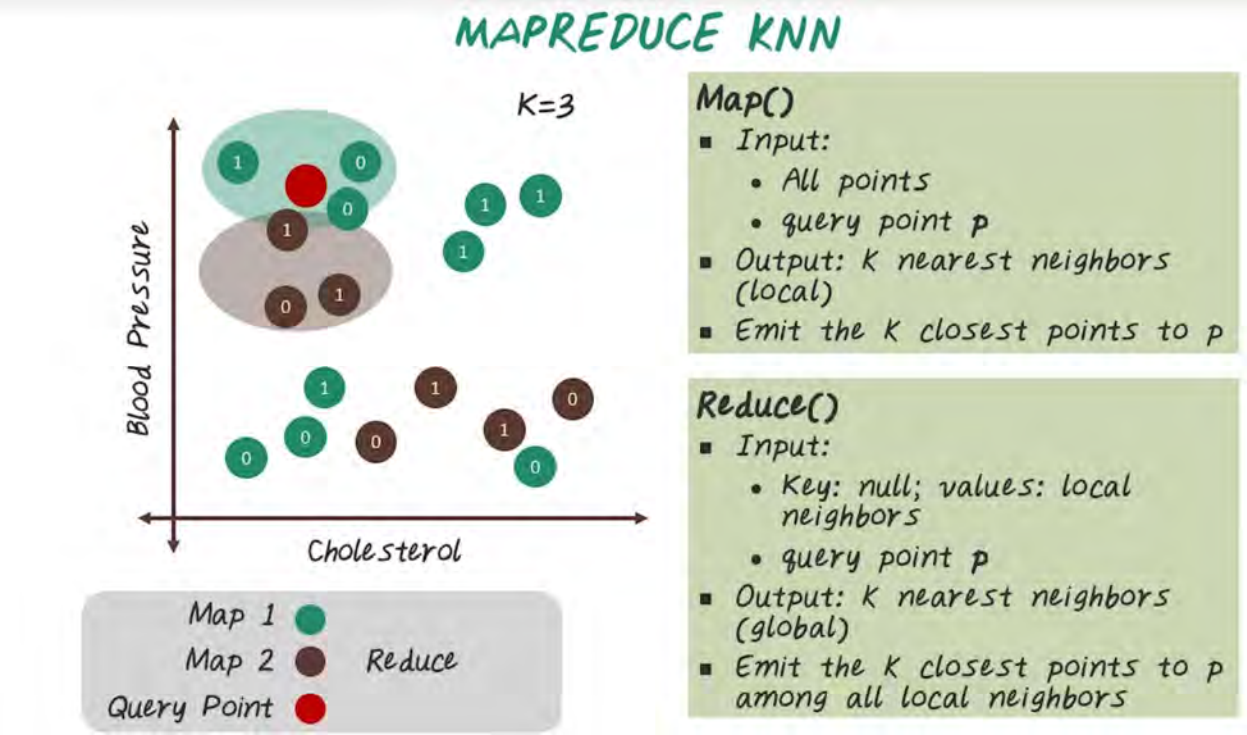
Mapper reduces for the system it's on, shuffled to pass data to the final reducers which then generate the output.

MapReduce Fault-Recovery
If a mapper or reducer fails, MapReduce will automatically restart the Mapper or Reducer. It's designed for only the component that fails gets recomputed.

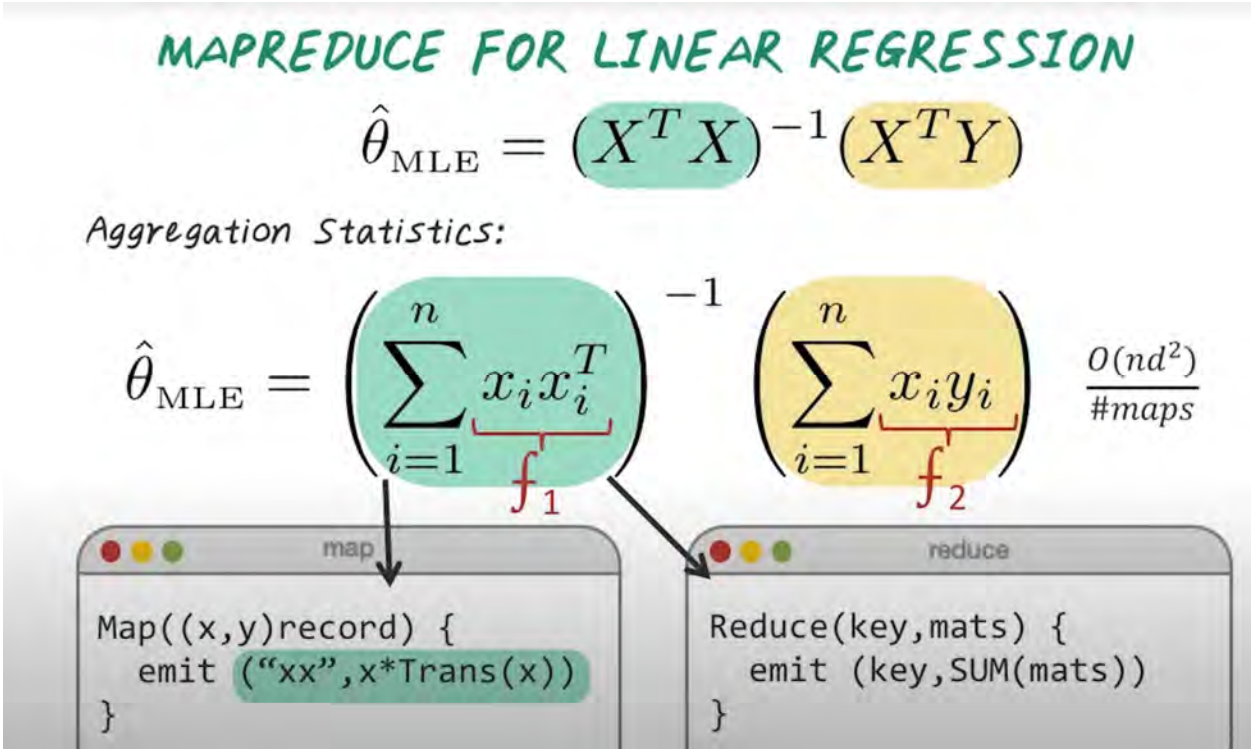
Distributed File Systems
Hadoop Distributed File System (HDFS) will store large files on different machines, breaking up the file so that parts A, B, C, D are on different machines, but also repeated so Machine 1 may have A, B, C, Machine 2 has B, C,D, MAchine 3 has A, D, C, etc… This allows for fault tolerance if one system fails, but also for faster computation as each worker can work on a section of the data.

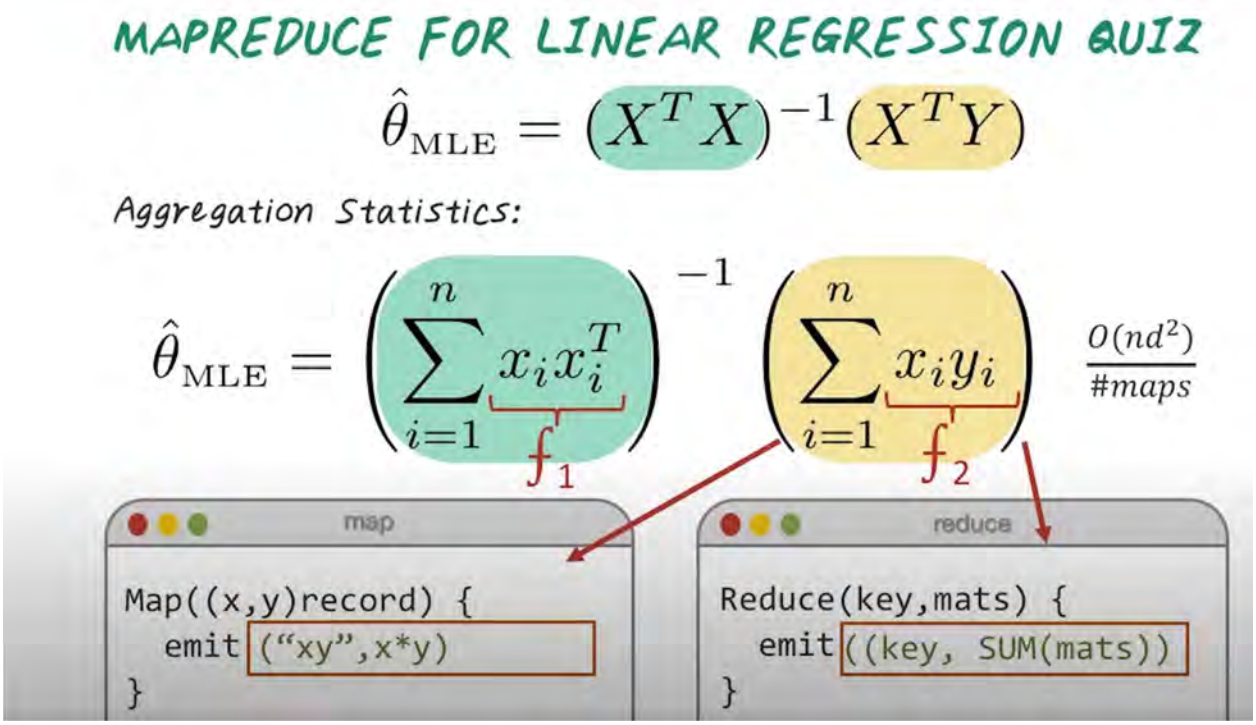
Map Reduce Design Choice
Not focused on what functionality can be added to make it more useful. But what can be removed while still being useful? This makes it more tolerant of failure. Machine Learning Algorithms cannot direction access data, instead all they can access is E[f(x) = 1/n * SUM(f(xi). While very limited, still very powerful.

MapReduce KNN example



MapReduce Linear Regression





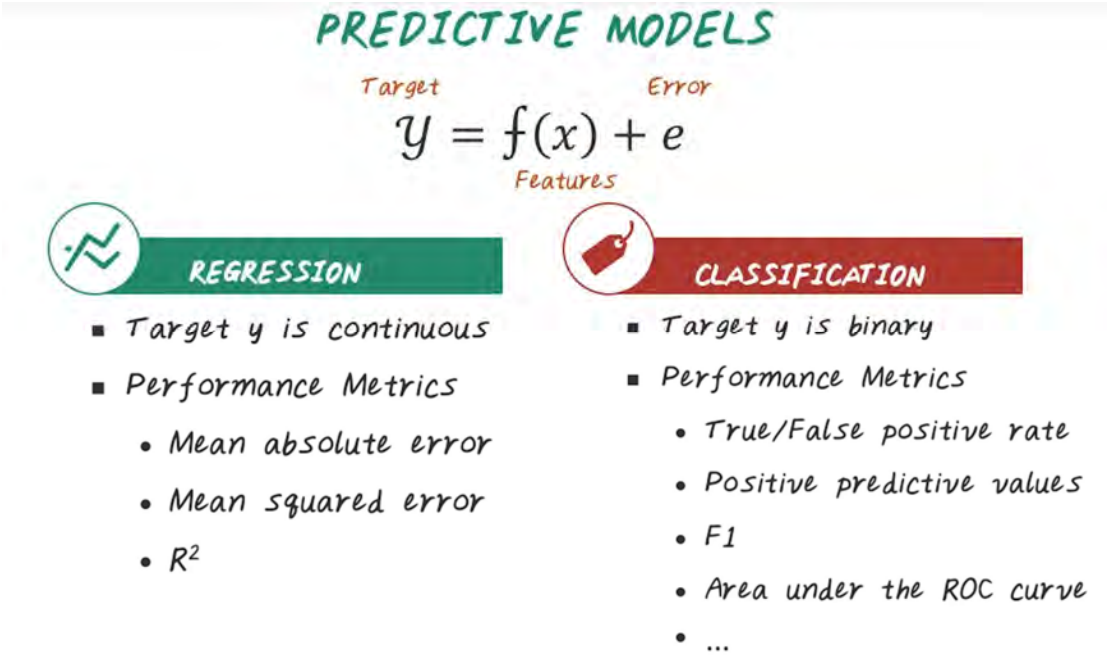
Limitations of MapReduce

Logistic Regression and Gradient Descent don't work well. Requires iterative functionality. For example apply a gradient on the data, return it back, and back and forth. MapReduce is not optimized for iteration and multi-stage computation

MapReduce is best for

- Single-pass functions
- Uniformly-distributed keys
- No synchronization needed

Classification Metrics



Confusion Matrix

		Ground Truth	
	TOTAL POPULATION	Condition Positive	Condition Negative
Prediction	Prediction Outcome Positive	True Positive	False Positive (Type I error)
	Prediction Outcome Negative	False Negative (Type II error)	True Negative

Example

		Ground Truth	
	TOTAL POPULATION 1000	Condition Positive 65	Condition Negative 935
Prediction	Prediction Outcome Positive 155	True Positive 55	False Positive 100
	Prediction Outcome Negative 845	False Negative 10	True Negative 835

Accuracy

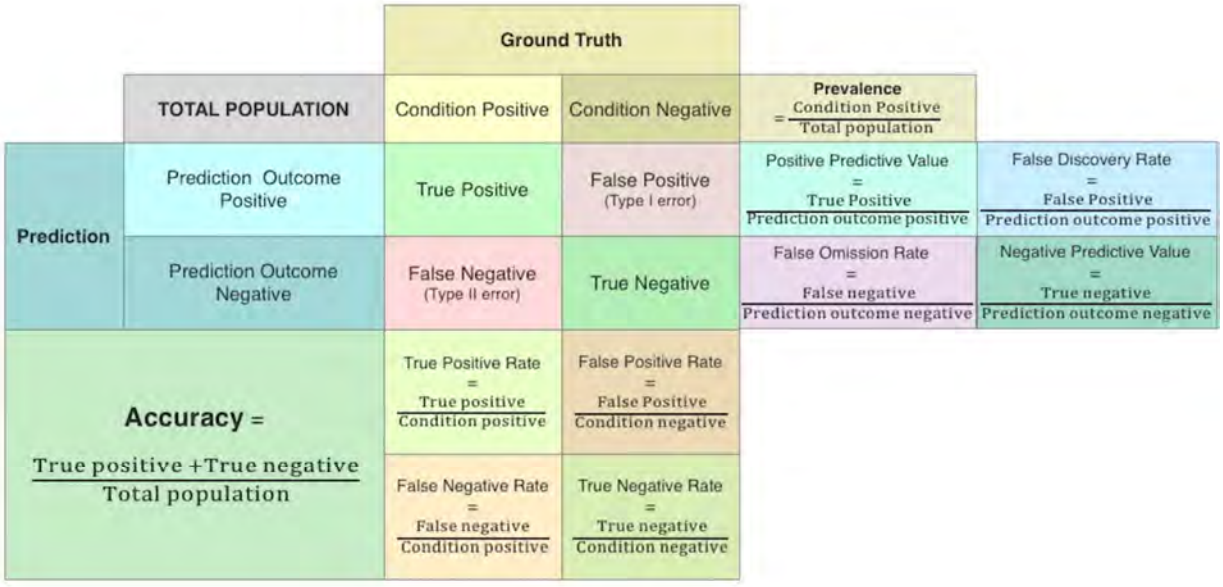
		Ground Truth	
	TOTAL POPULATION	Condition Positive	Condition Negative
Prediction	Prediction Outcome Positive	True Positive	False Positive (Type I error)
	Prediction Outcome Negative	False Negative (Type II error)	True Negative
$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Total population}}$		$\text{True Positive Rate} = \frac{\text{True positive}}{\text{Condition positive}}$	$\text{False Positive Rate} = \frac{\text{False Positive}}{\text{Condition negative}}$
		$\text{False Negative Rate} = \frac{\text{False negative}}{\text{Condition positive}}$	$\text{True Negative Rate} = \frac{\text{True negative}}{\text{Condition negative}}$

True Positive Rate (Sensitivity, Recall): Among all with Heart Failure, what percentage is correctly identified by the model

False Negative Rate: Those who have heart failure, who the model missed
False Positive Rate: Among all without Heart Failure, which are predicted to have heart failure
True Negative Rate Rate(Specificity): For all those without heart failure, how many do not have heart failure

Often it is easy to get high accuracy, but not do well on other metrics

Predictive



Prevalence: How likely disease occurs in total population
Positive Predictive Value (Precision): Among all predicted to have HF, how many have it
False Discovery Rate: Among all predicted to have HF, how many predictions are incorrect
Negative Predictive Value: Among all predicted not to have HF, the percent that actually do not have it
False Omission Rate: Among all predicted not to have HF, which will actually have HF
Example:

		Ground Truth			
	TOTAL POPULATION 1000	Condition Positive 65	Condition Negative 935	Prevalence 7%	
Prediction	Prediction Outcome Positive 155	True Positive 55	False Positive 100	Positive Predictive Value 35%	False Discovery Rate 65%
	Prediction Outcome Negative 845	False Negative 10	True Negative 835	False Omission Rate 1%	Negative Predictive Value 99%
Accuracy 89%		True Positive Rate 85%	False Positive Rate 11%	Please fill in the missing numbers.	
		False Negative Rate 15%	True Negative Rate 89%		

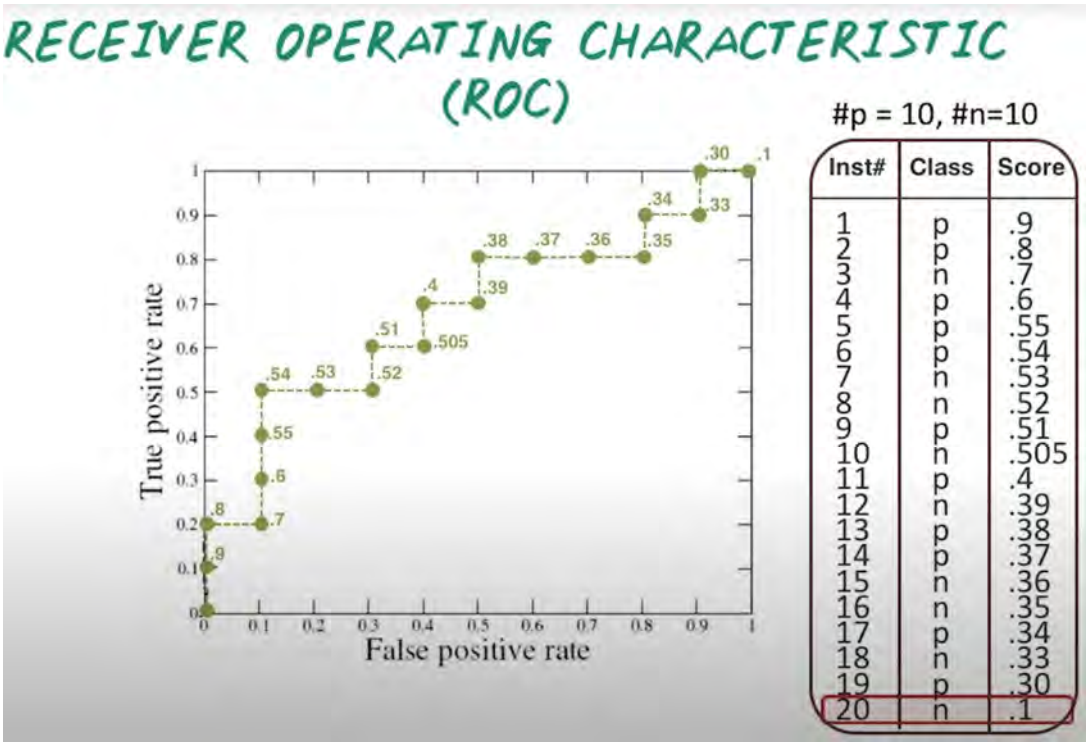
F1 Score:

		Ground Truth			
	TOTAL POPULATION	Condition Positive	Condition Negative	Prevalence = Condition Positive / Total population	
Prediction	Prediction Outcome Positive	True Positive	False Positive (Type I error)	Positive Predictive Value = True Positive / Prediction outcome positive	False Discovery Rate = False Positive / Prediction outcome positive
	Prediction Outcome Negative	False Negative (Type II error)	True Negative	False Omission Rate = False negative / Prediction outcome negative	Negative Predictive Value = True negative / Prediction outcome negative
Accuracy = (True positive + True negative) / Total population		True Positive Rate = True positive / Condition positive	False Positive Rate = False Positive / Condition negative	$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR}$	
		False Negative Rate = False negative / Condition positive	True Negative Rate = True negative / Condition negative		

The harmonic mean of the two numbers

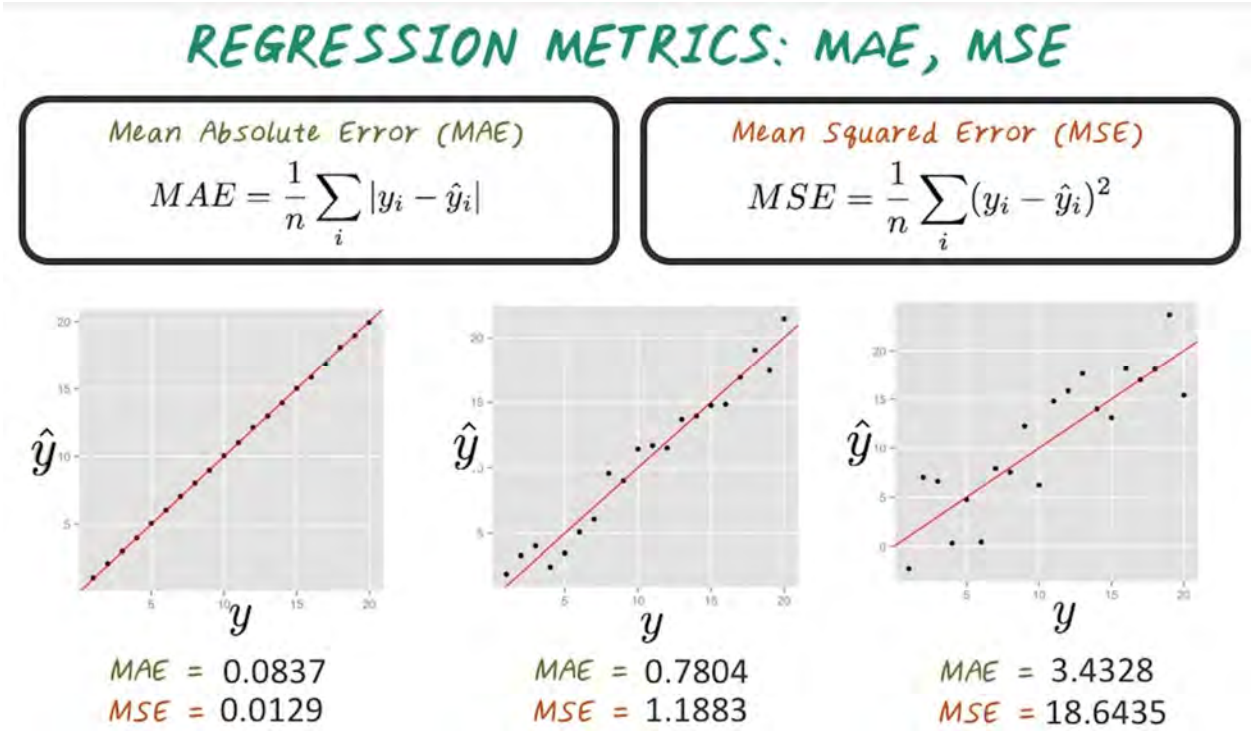
Receiver Operating Characteristic (ROC) Curve

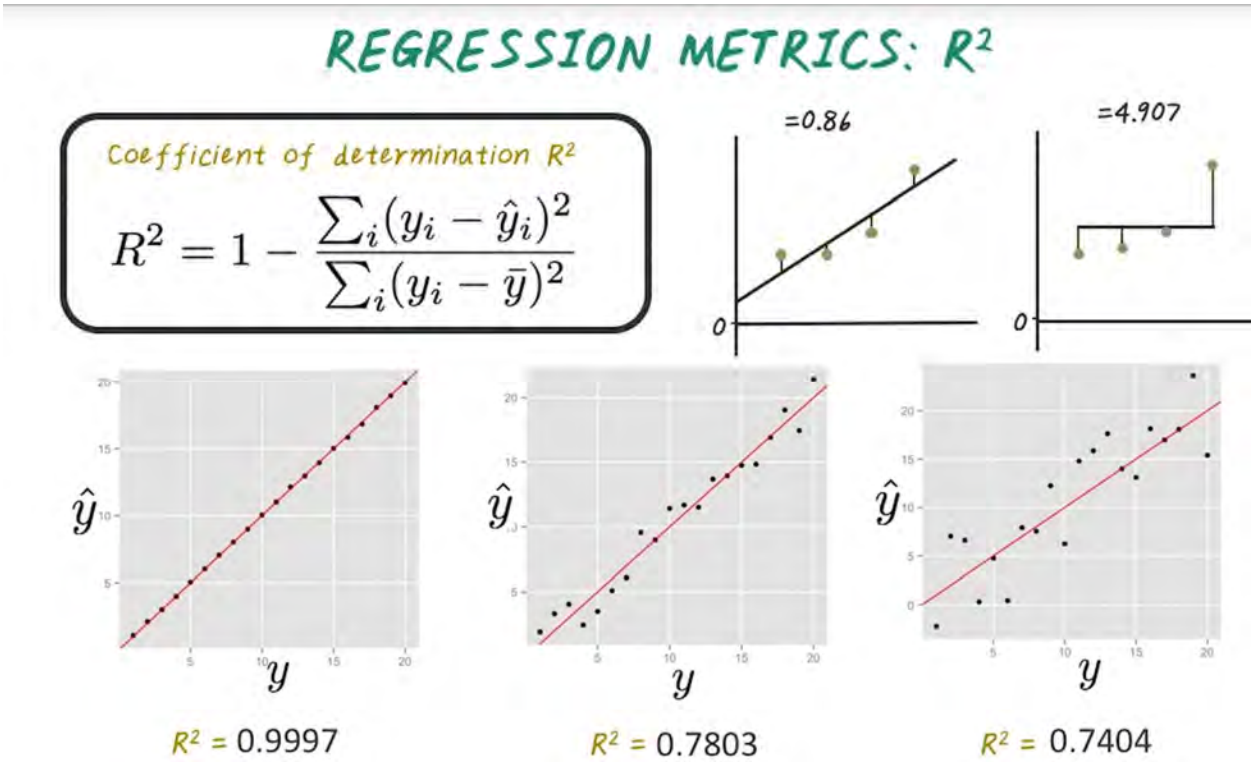
On a grid of True Positive Rate vs False Positive Rate, plot the rates at various threshold values. Sort points based on prediction score, then plot going up for true positive, right for false positive. Once graphed, get the area under the curve (AUC) for a number that describes the quality of the classifier.



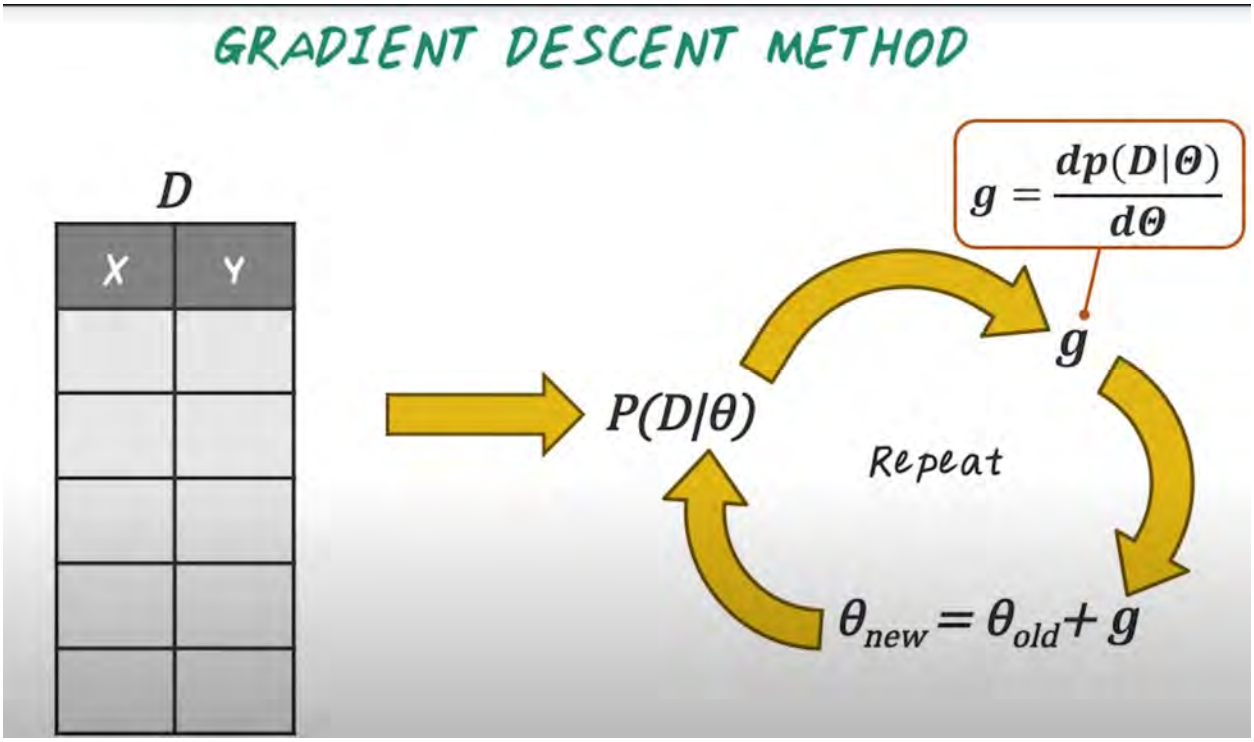
Regression Metrics

MSE is typically easier to work with as the derivative is linear, and as error grows MSE grows very quickly





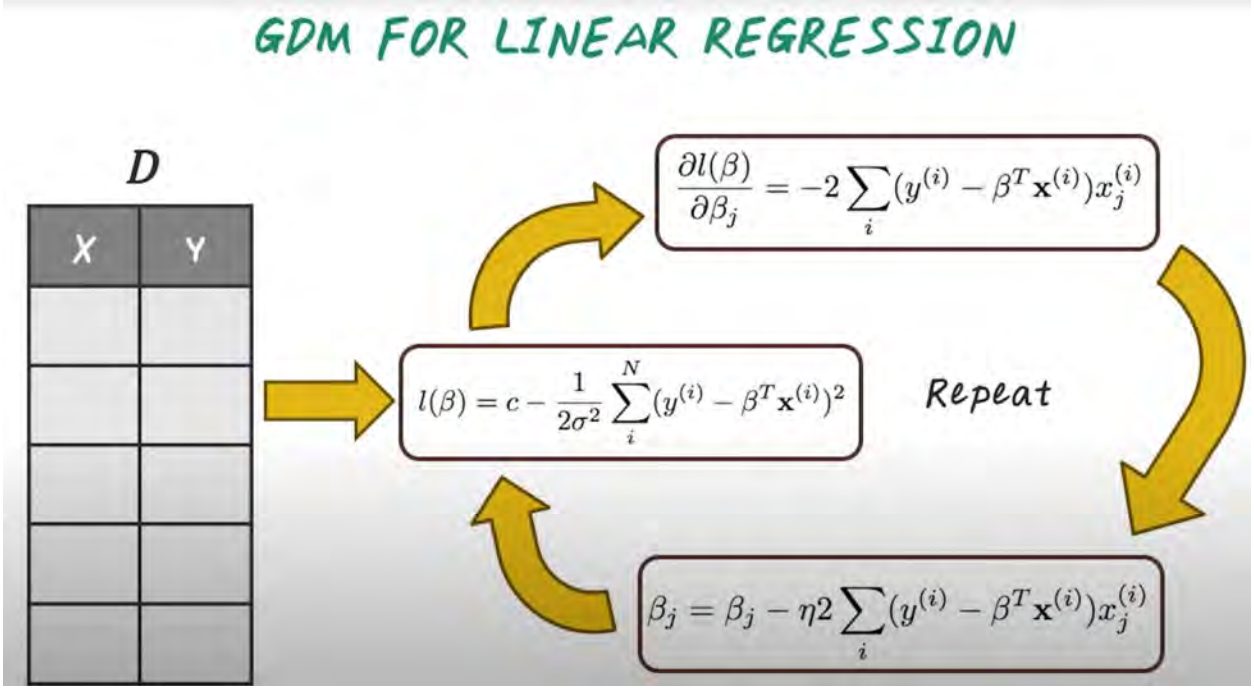
Classification: Ensemble Methods



Gradient Descent Method

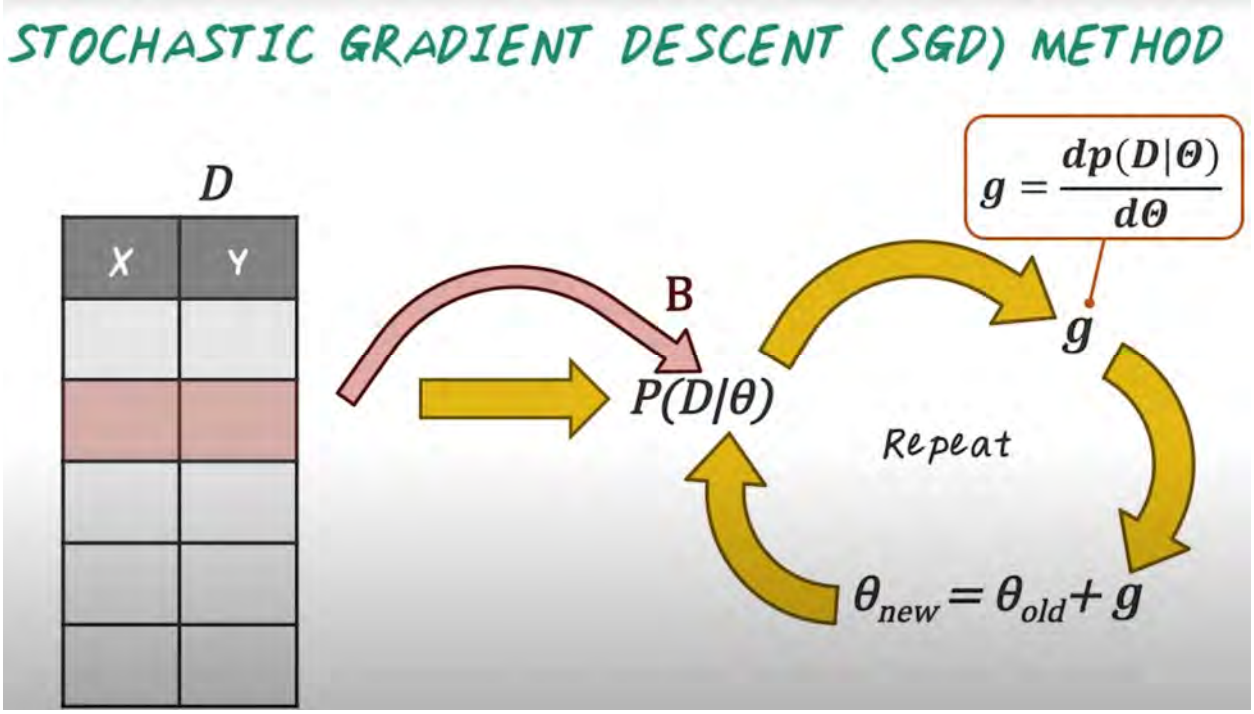
- Can be used for Classification or Regression

- Requires a training set of *n* pairs of data points x and a target y where the final output is a model with a parameter set.
- Three Step process:
    - Specify a likelihood function P of input Data given parameter Theta
        - Likelihood function is the joint distribution of the data points, given model parameter theta
        - Log Likelihood is often easier to compute, since it is monotonically increasing it has the same optimal as the original
    - Find the derivative, called the gradient, of the likelihood function
        - The crucial step of the process
    - Update theta by moving the old theta in the direction of the gradient multiplied by some learning rate
    - Repeat these steps until the difference theta changes is below some threshold



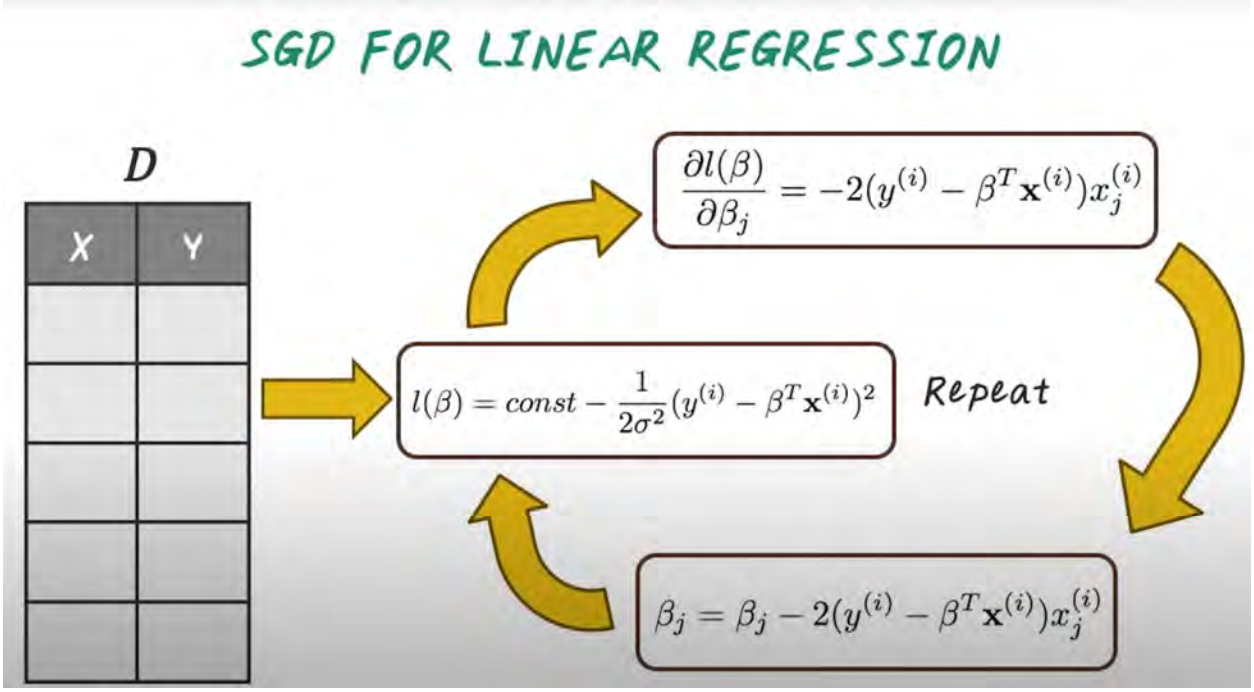
GDM For Linear Regression

- Similar to Classification
- Three step process again
    - Specify Log Likelihood Function
        - Linear Regression assumes a Gaussian Distribution
        - A constant minus the sum of a set of squared terms, the data points
        - Find the true y and subtract what the model predicts
    - Take the derivative to find the gradient. We have a set of parameters, one for feature, and need a gradient for each parameter
    - Update each Beta coefficient by moving towards the gradient direction by learning rate eta.
    - Repeat until convergence
- Can be very expensive on a large dataset



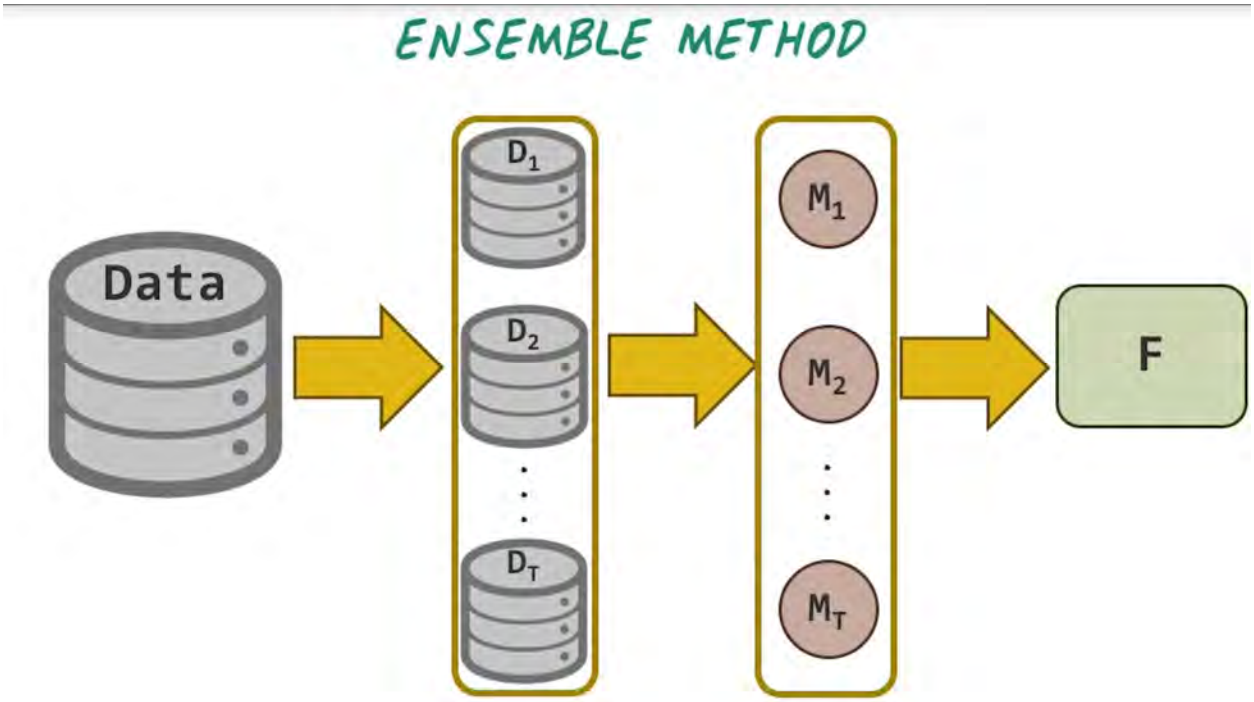
So, Stochastic Gradient Descent, or SGD Method, is a variant of gradient descent for handling big data sets.

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 is 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.



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 involves 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.

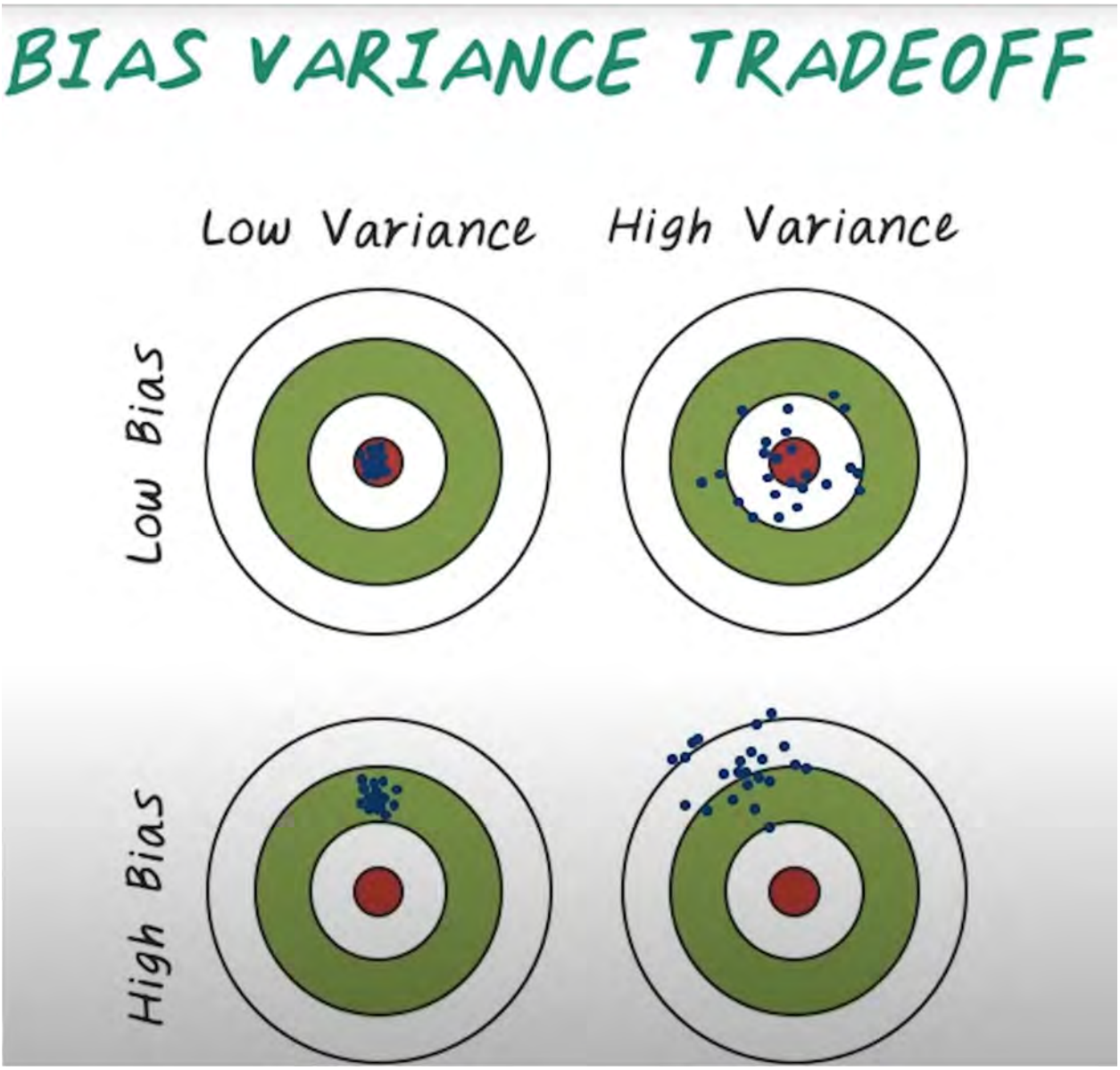
So this process is very similar to gradient descent for linear regression.
So the main difference is, when we compute the log likelihood and the gradient, only a single patient is involved.
We don't have to do this for the entire dataset.
And this is much more efficient.



So far, we've talked about stochastic gradient descent as an efficient method for dealing with large dataset.
Next, we introduce an 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 forests.
However, more generally it can also refer to combining models using different algorithms.
Ensemble models often outperform other classification methods.
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 users 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 on October 2006, and ended on 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 submitted 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.

7
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.



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

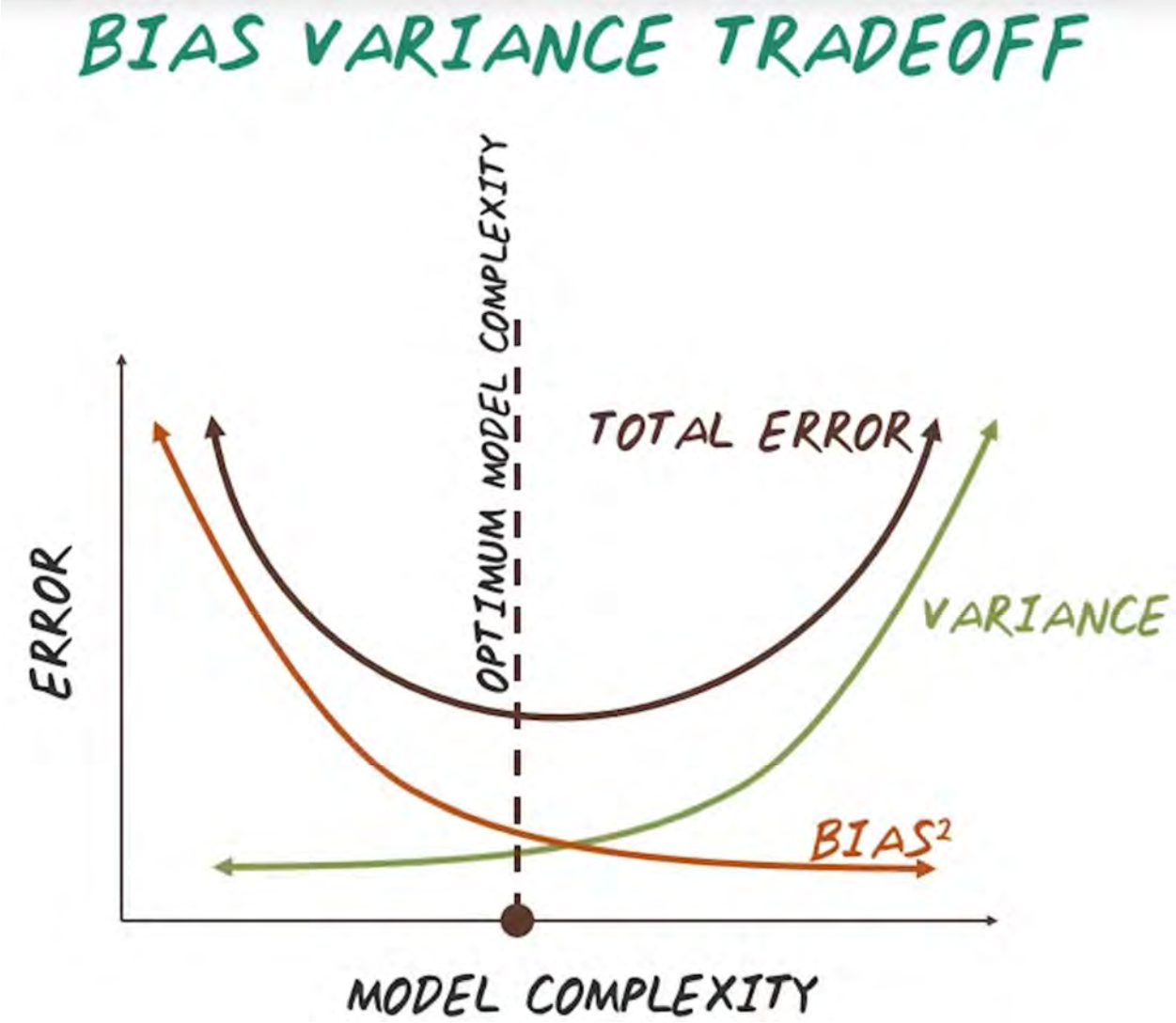
due to the wrong modeling assumption.

For example, if the model assumes linear relationship between the target.

However, in reality the data do not follow the linear trend, then the difference will be due to the bias of the model.

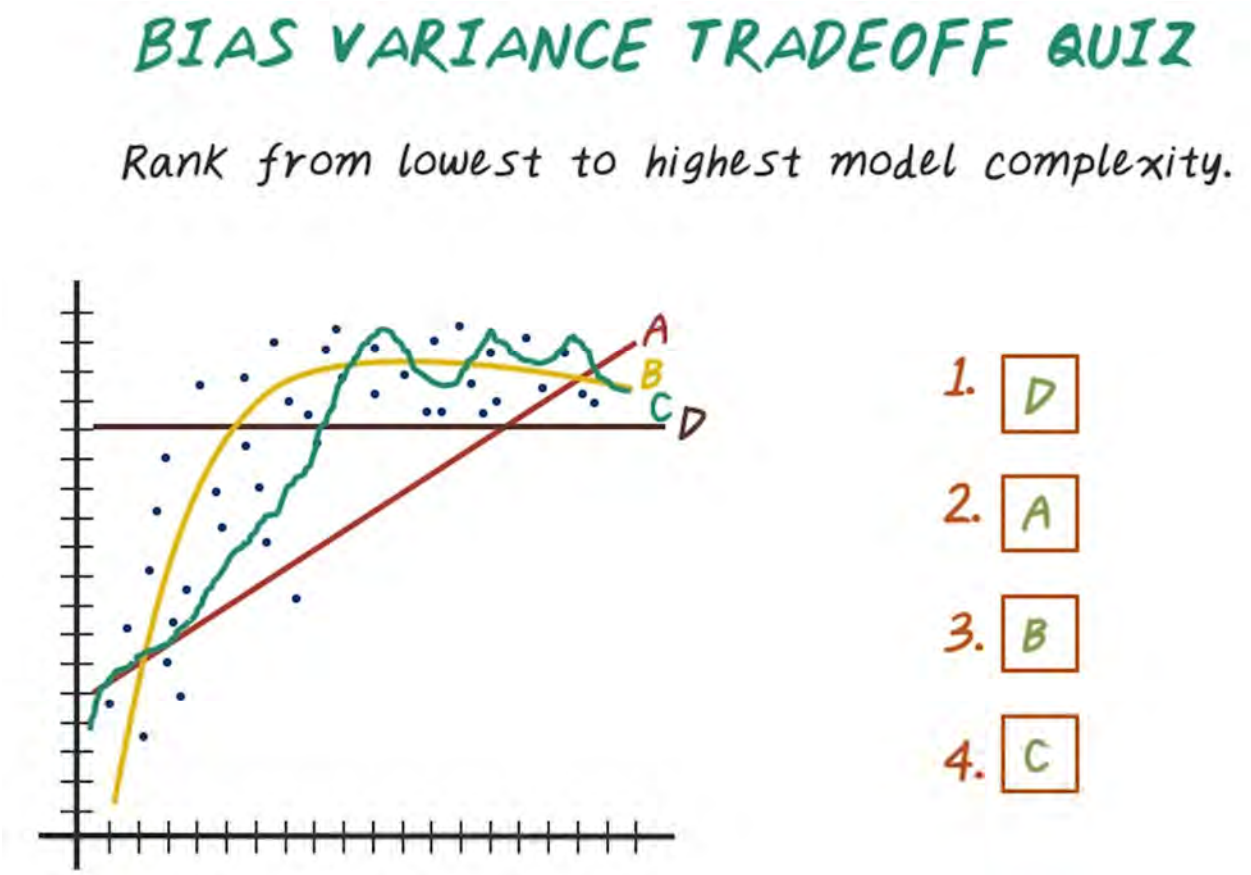
The variance on the other hand, refers to the error from the sensitivity to small fluctuation in the training data set.

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.

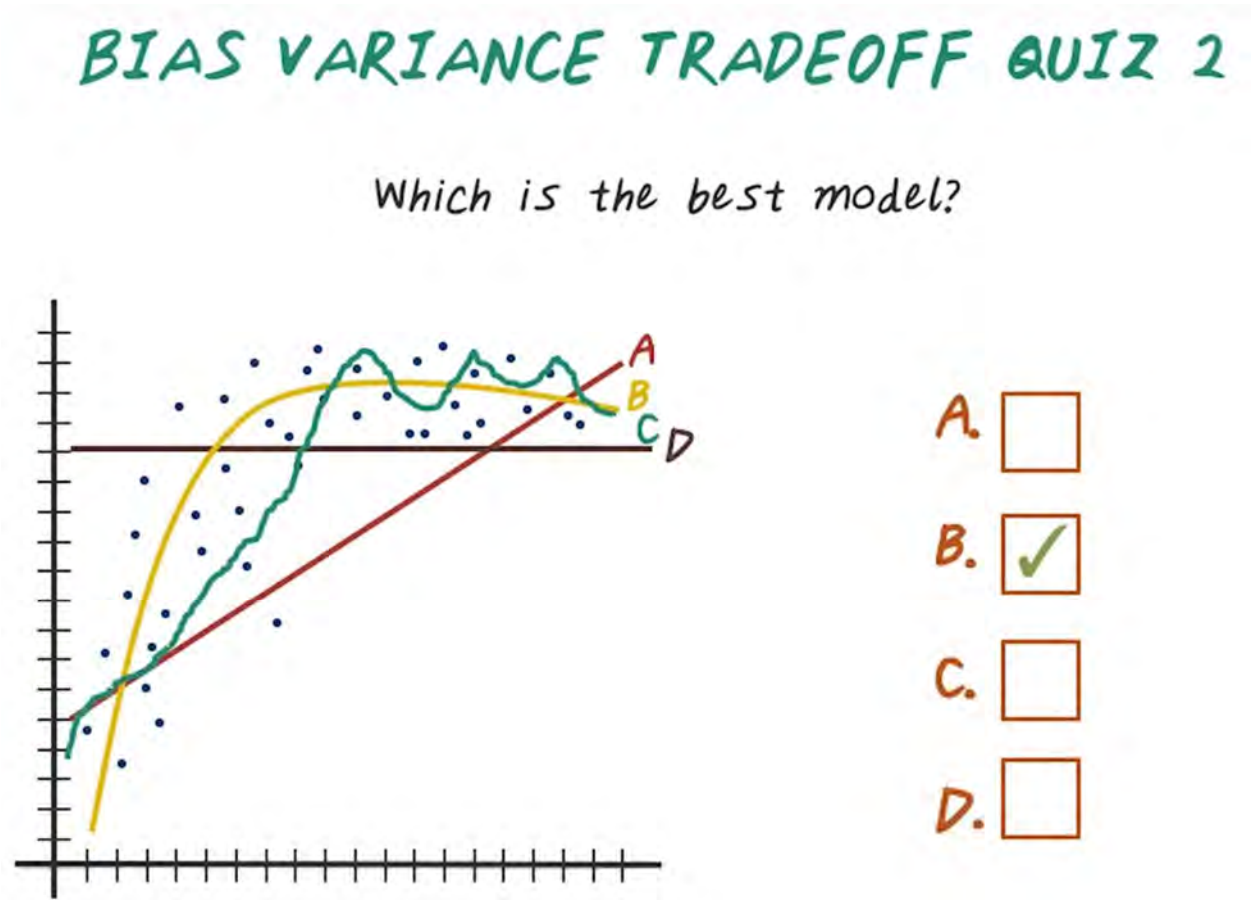
Now we understand the intuition behind the bias and variance.
Now let's look at the relationship between bias variance.
I want to explore the relationship using this plot.
The x-axis is model complexity, and the y-axis is the error from the model.
In general, there's a tradeoff between bias and variance.
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.



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.

10
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.

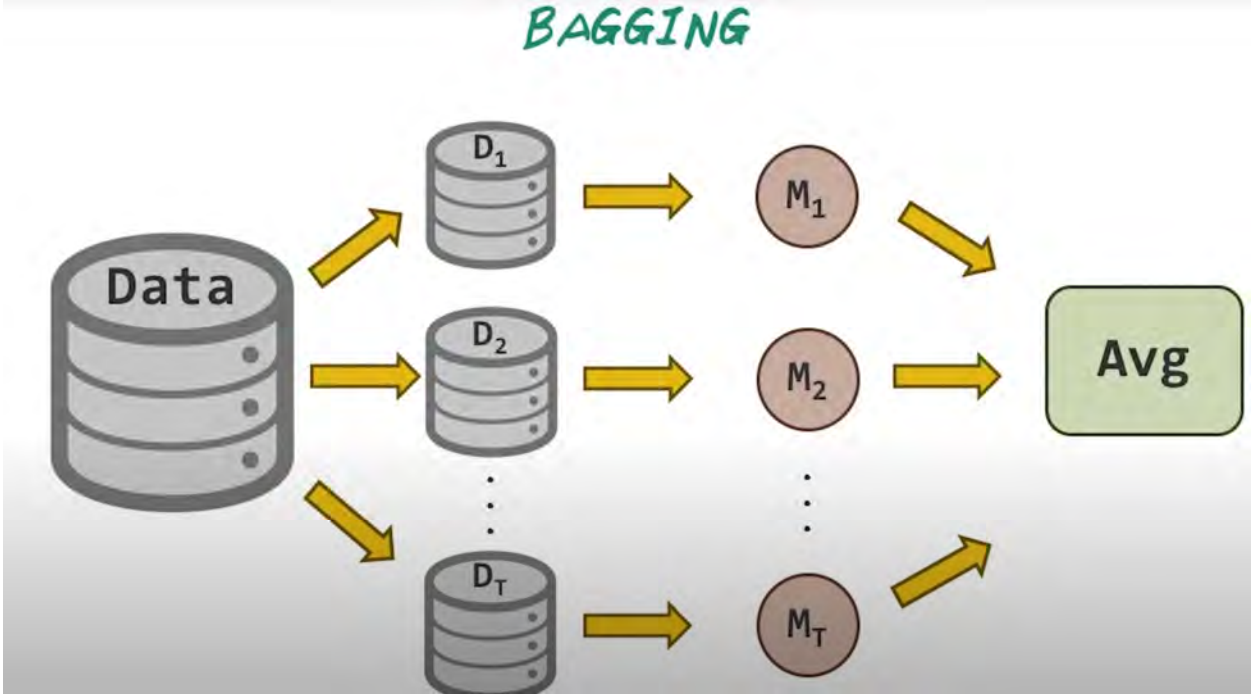


Lets do another quiz, using the same examples.
Note that, all those four models are trying to approximate under line data, which are those loop dots.
So, which of the models is the best for approximating the underlying data?

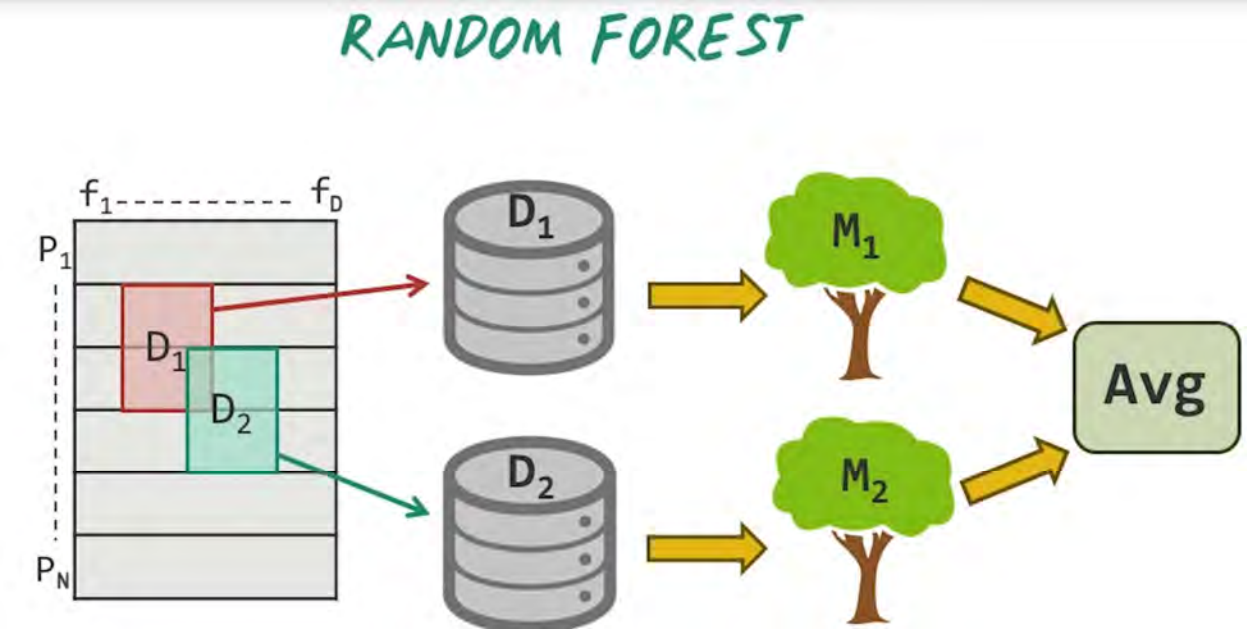
12

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.

13



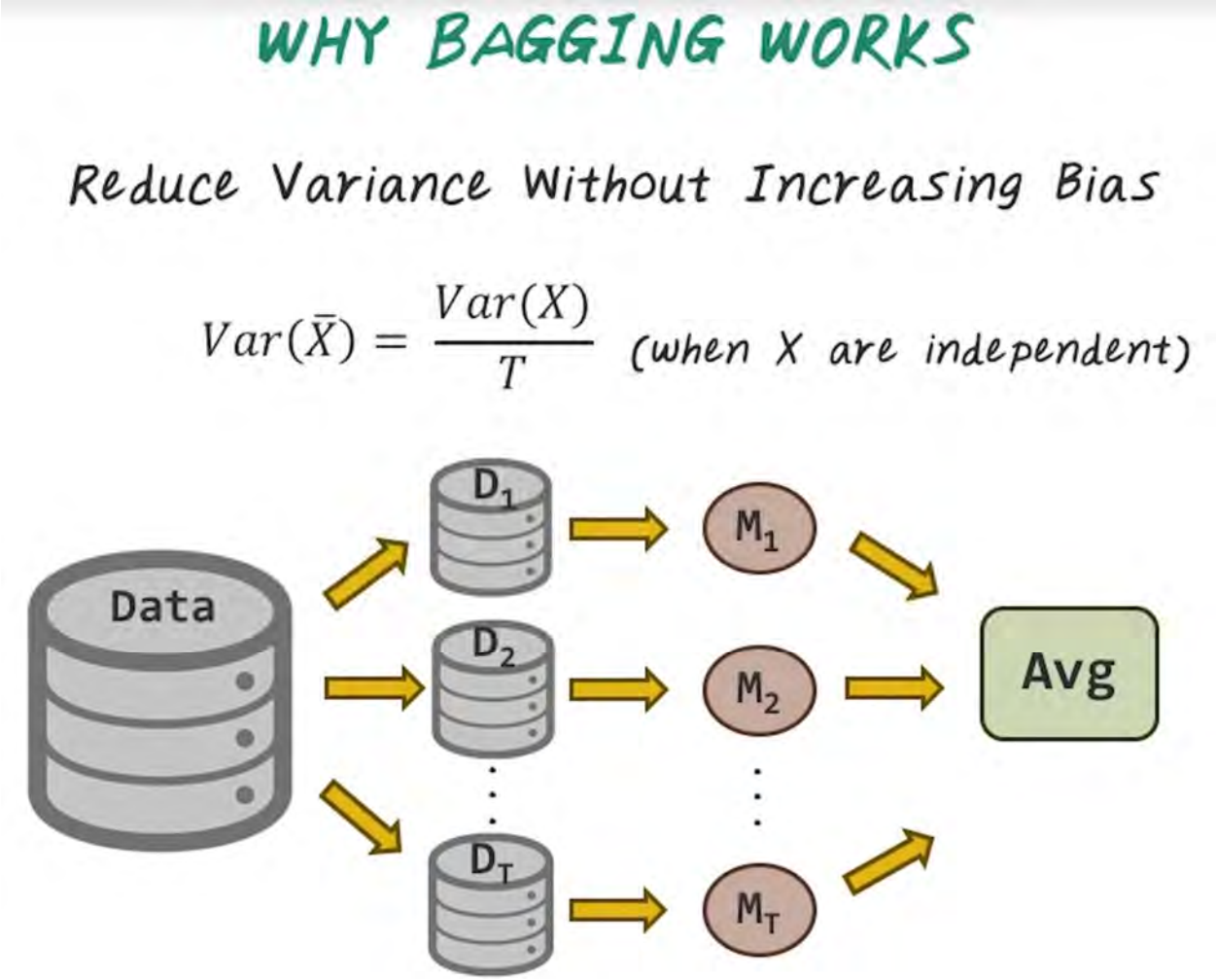
Now let's talk about some popular ensemble method.
Let's start with bagging.
Bagging is a simple, yet powerful ensemble strategy, which is named by Leo Brayman.
Given an input data set, the idea behind bagging is to take repeated bootstrap samples from input data set.
To generate all those sample data 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.



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.

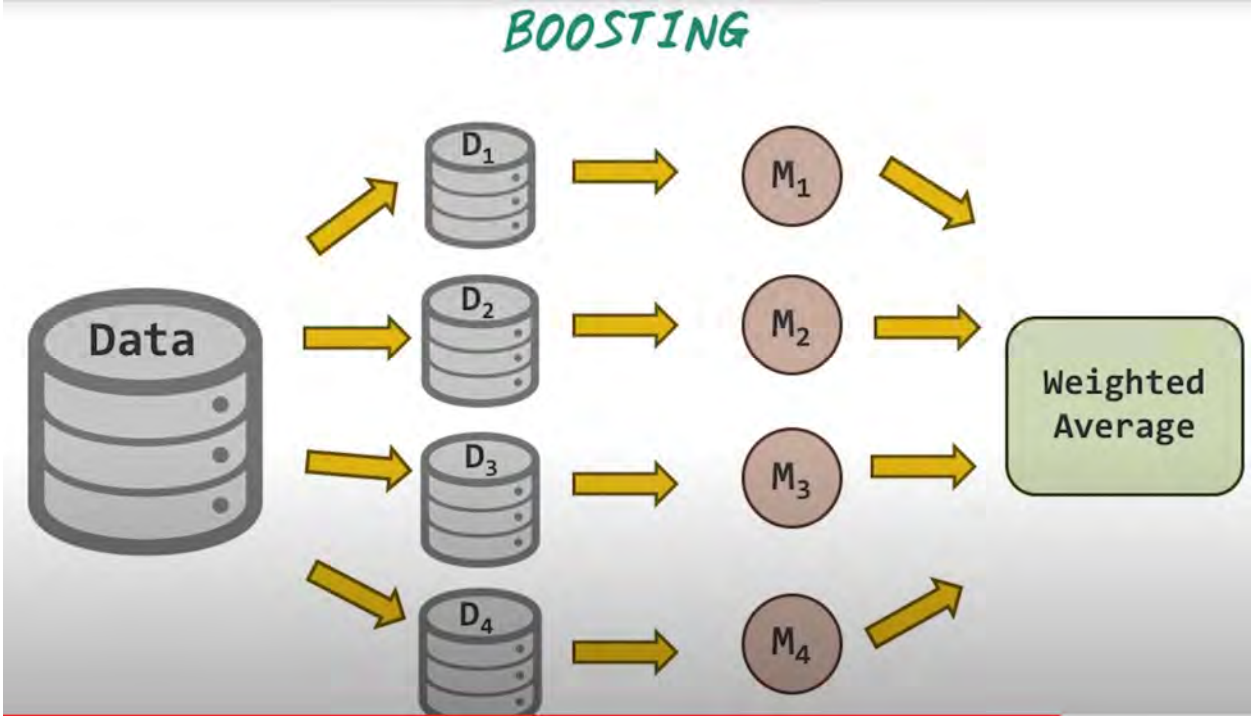
The other benefit for using those simple algorithm is computational cost will be dramatically reduced.

15



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.



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.

17
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?

18

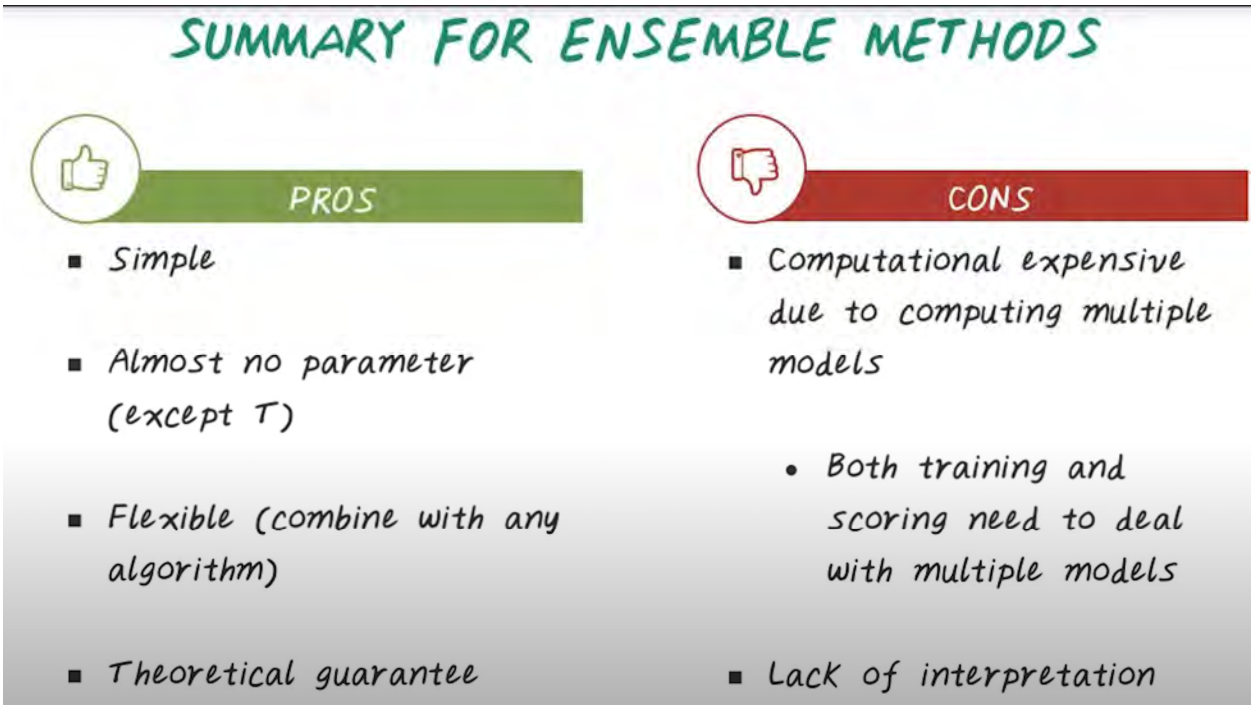
So in this lecture, we talked about ensemble method.
So now, let's summarize the overall pros and cons of ensemble method.
In term of advantages, ensemble method are often simple.
Almost have no parameters, except how many models to be combined.
And it's quite flexible, you can combine many algorithms 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 the training time, but also at the scoring time.
Every time we want to score a patient instead of score 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.

18

BAGGING VS. BOOSTING QUIZ

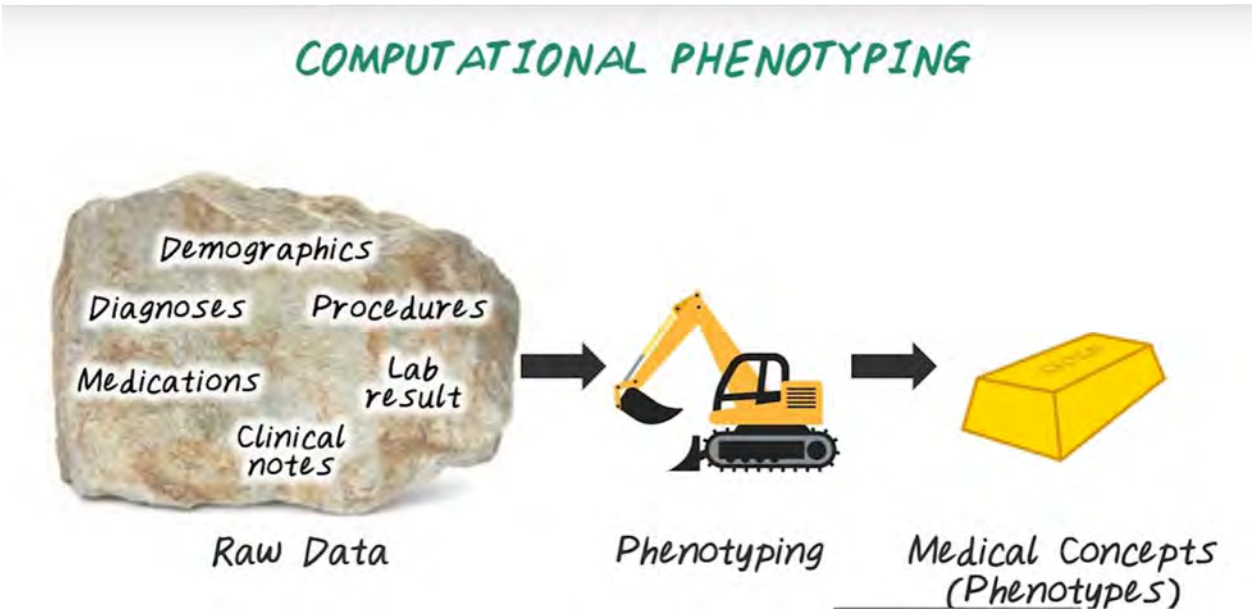
	BAGGING	BOOSTING
COMBINING METHOD	☑ Simple average ☐ Weighted average	☐ Simple average ☑ Weighted average
PARALLEL COMPUTING	☐ Hard ☑ Easy	☑ Hard ☐ Easy
SENSITIVE TO NOISE	☑ Less ☐ More	☐ Less ☑ More
ACCURACY	☑ Good in all cases ☐ Better in most cases	☐ Good in all cases ☑ Better in most cases

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,
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.

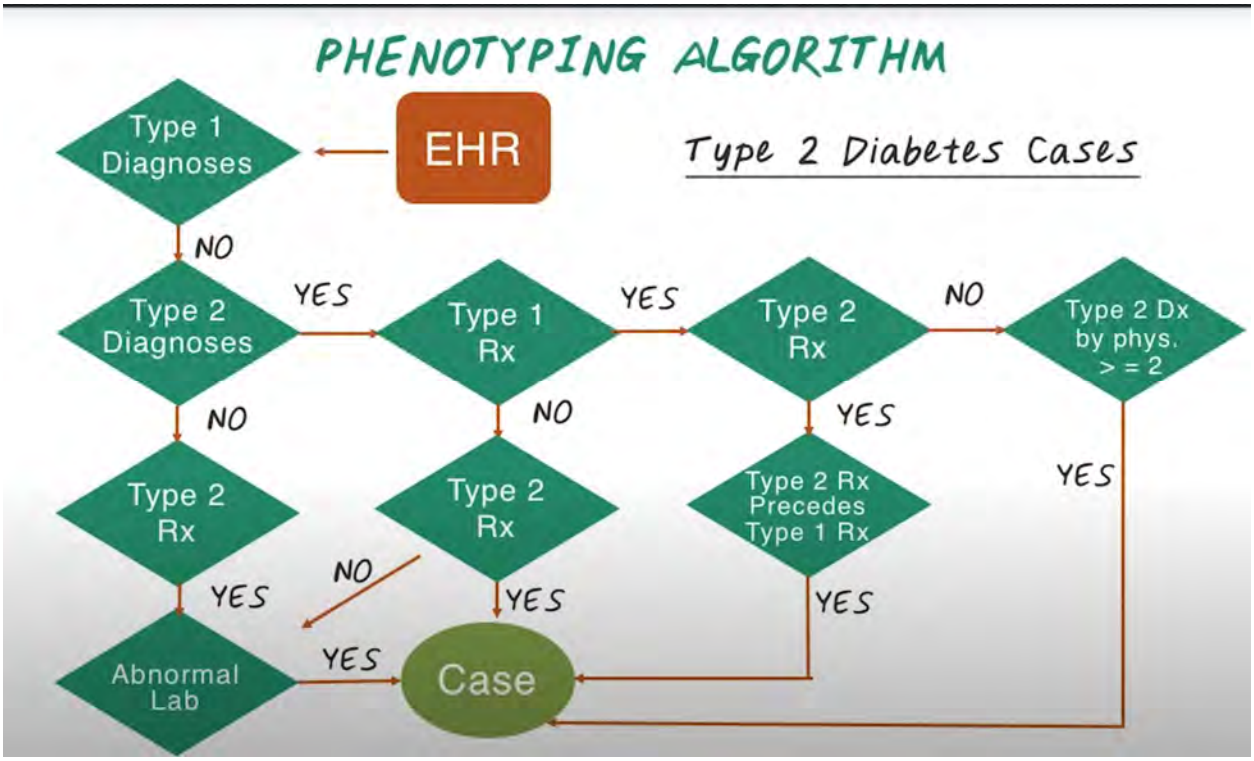


Computational Phenotyping

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.

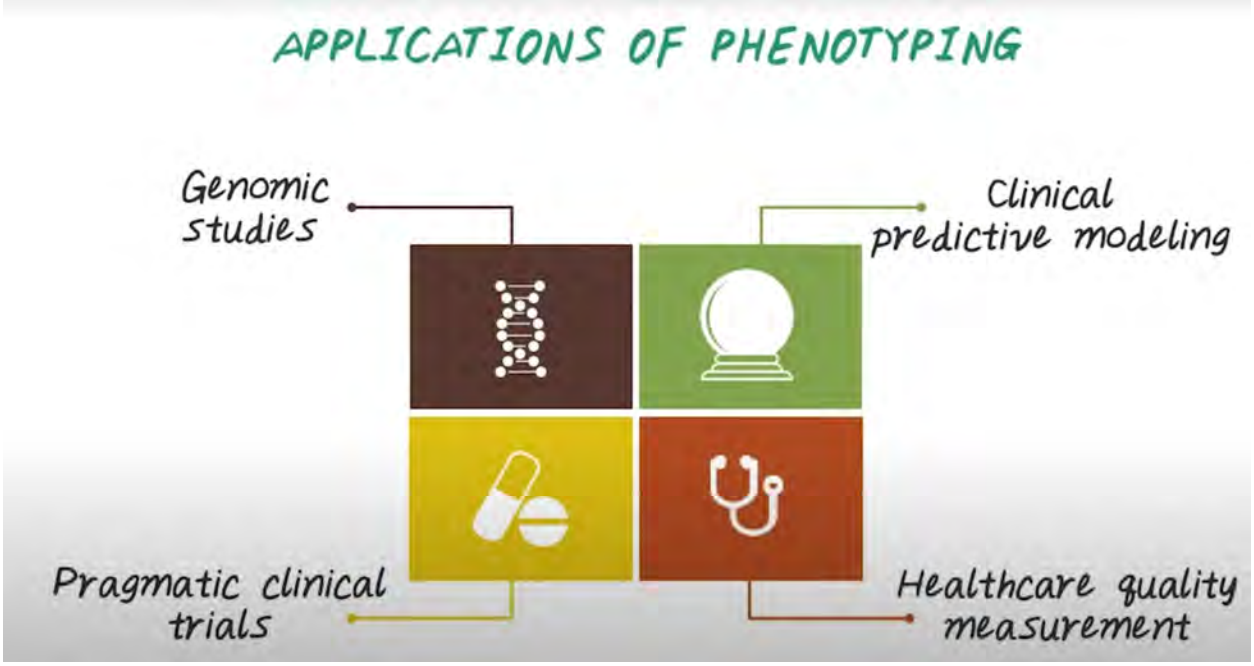


Now let's talk about computational phenotyping.
So recall, computational phenotyping is about converting raw electronic health record through phenotyping algorithms into a set of meaningful medical concepts, or phenotypes.
For example, a specific disease can be a phenotype.
Such as type 2 diabetes. And 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 datas 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.

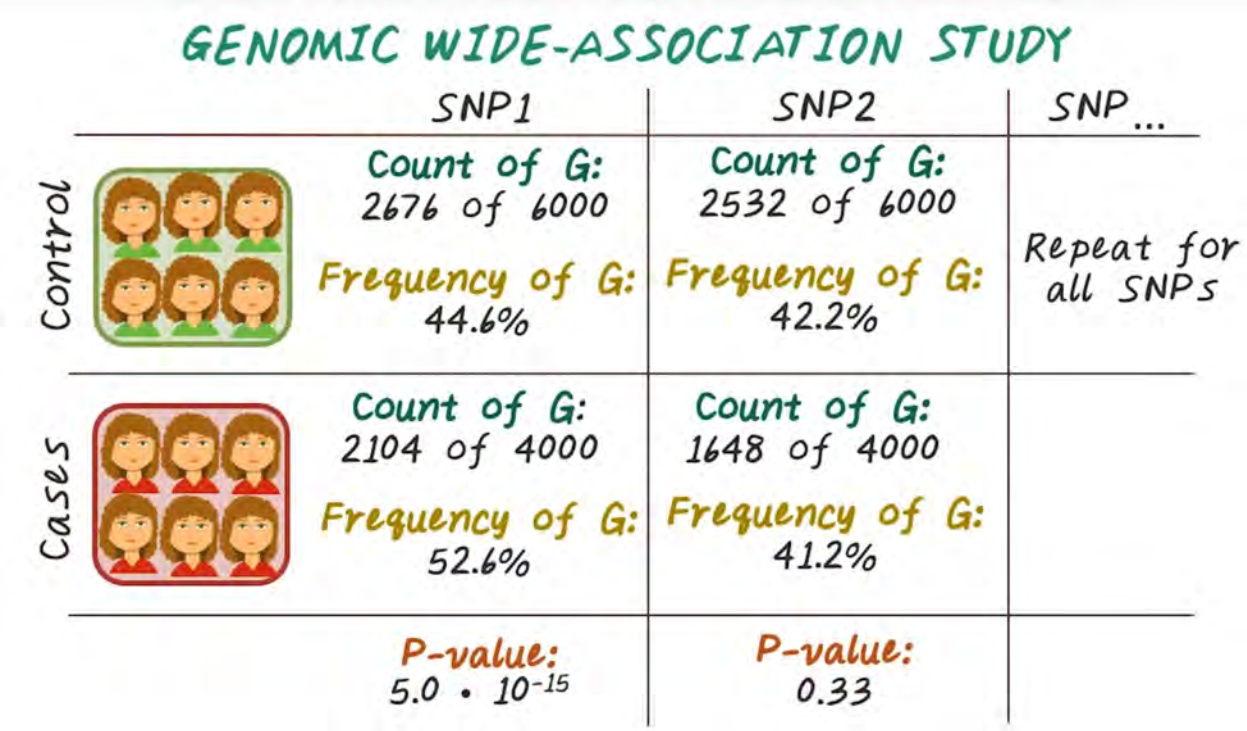


Here's a phenotyping algorithm for Type 2 diabetes.
And the goal here is, we want to determine whether patient has Type 2 diabetes based on her electronic health records.
For example, we can first check whether the patient has Type 1 diabetes code in her record.
If no, then we check whether Type 2 diabetes 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.

4. Applications of Phenotyping

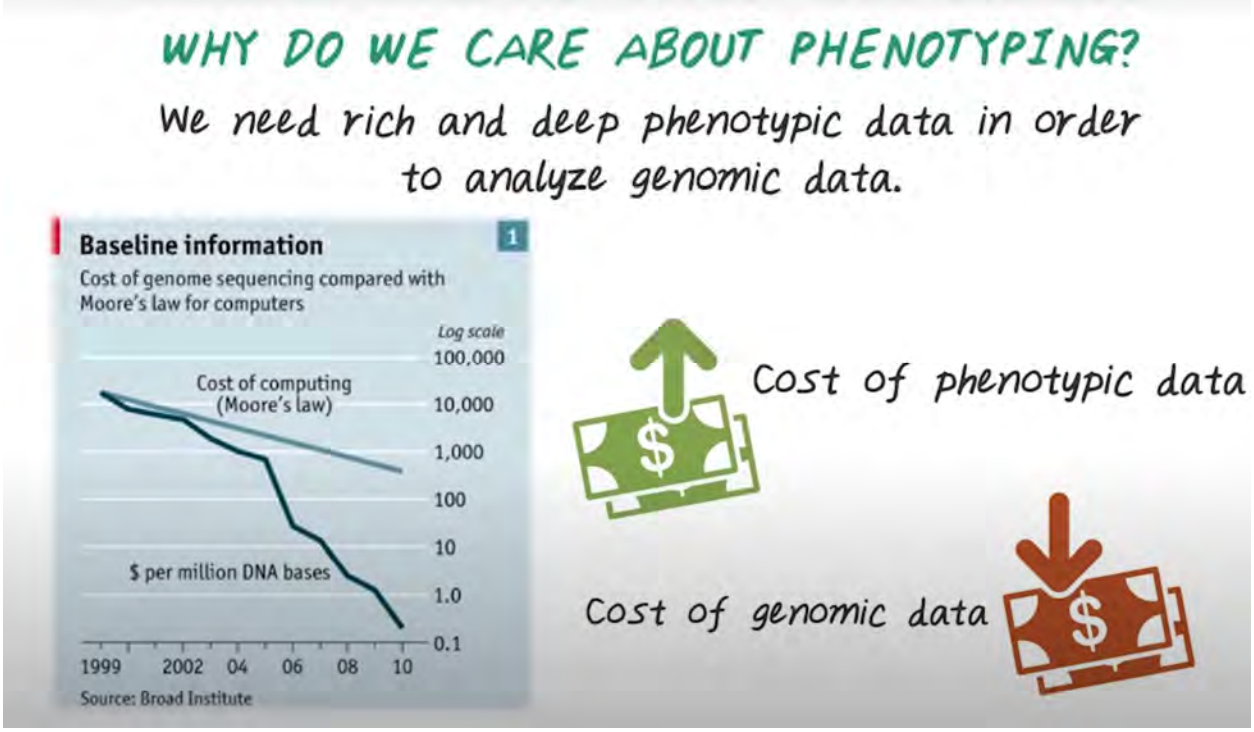


There are many different applications that require phenotyping. For example, genomic study, which is about finding relationship between genomic data and phenotypic data. Clinical predictive modeling, which is about building an accurate, robust, and interpretable prediction model about disease onset and 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.



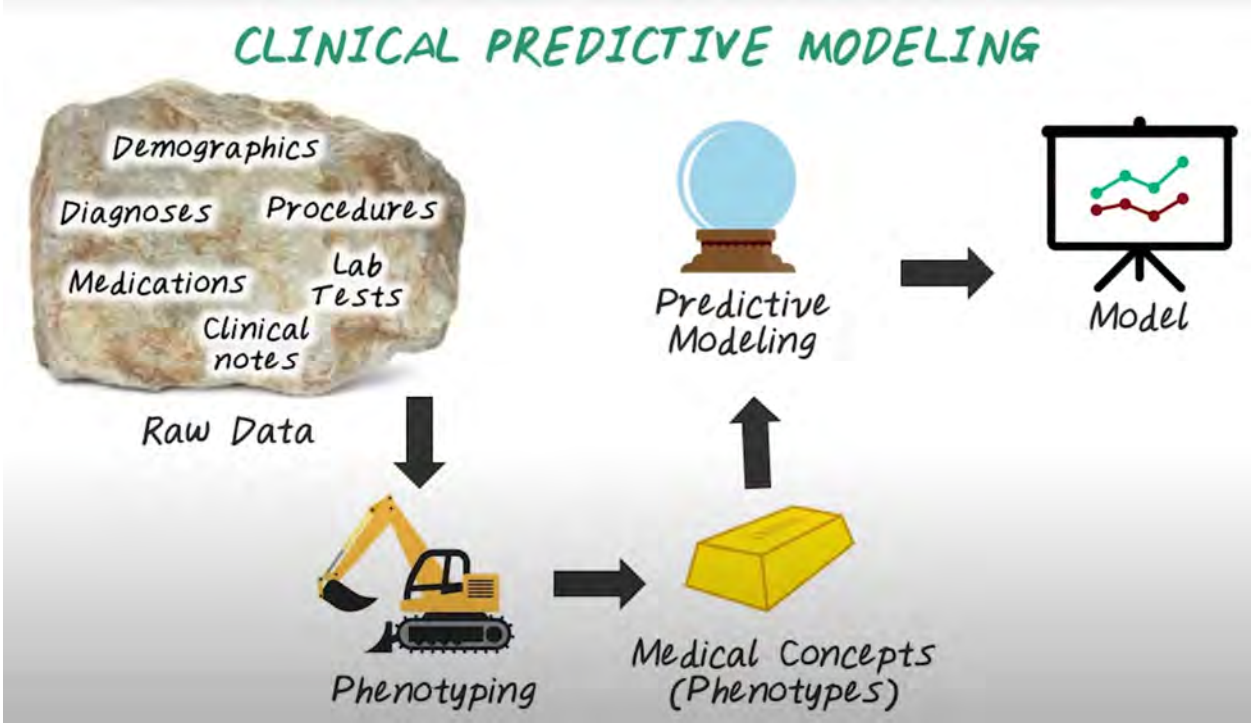
Phenotyping algorithm is very important in supporting genome-wide association study. What is a genome-wide association study? It's an approach that involves scanning biomarkers such as single nucleotide polymorphism. Or SNP's from DNA of many people, in order to find genetic variation, associated with 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, we'll find now 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.



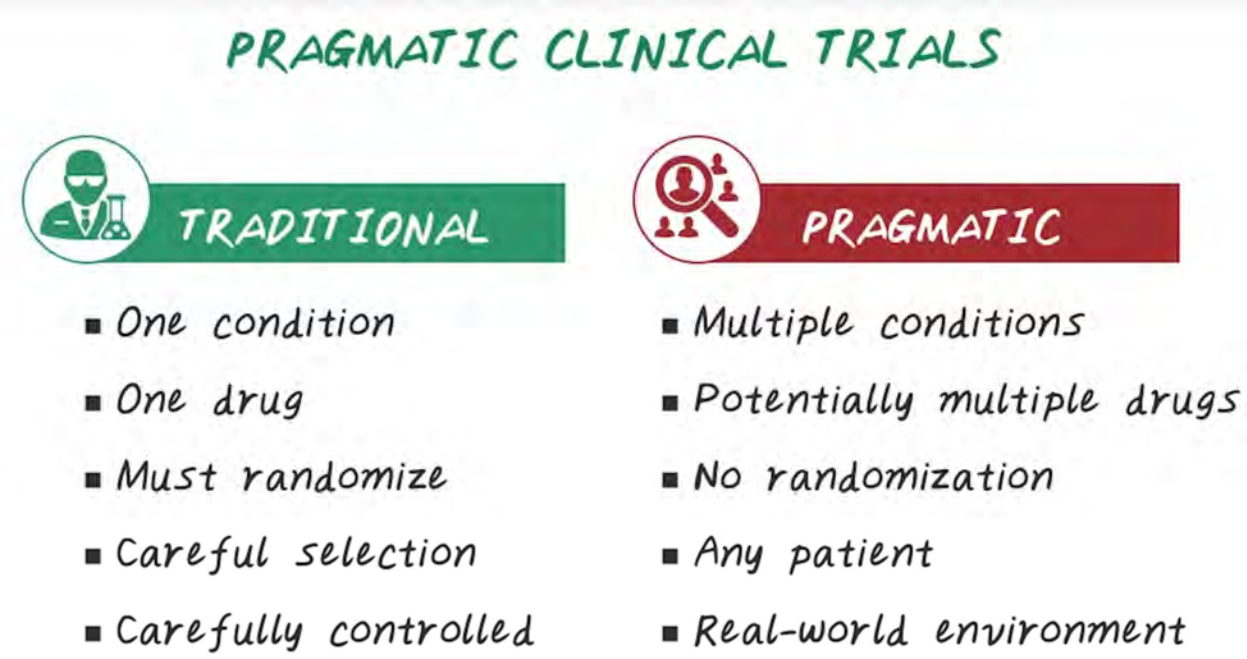
As we have shown in the genome-wide association studies, we need phenotypic data in order to analyze genomic data. But in general, why do we care about phenotyping algorithms in genomic study? In fact, many people argue that we need rich and deep phenotypic data in order to analyze genomic data. Especially 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.

7. Clinical Predictive Modeling



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 interpretival 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.

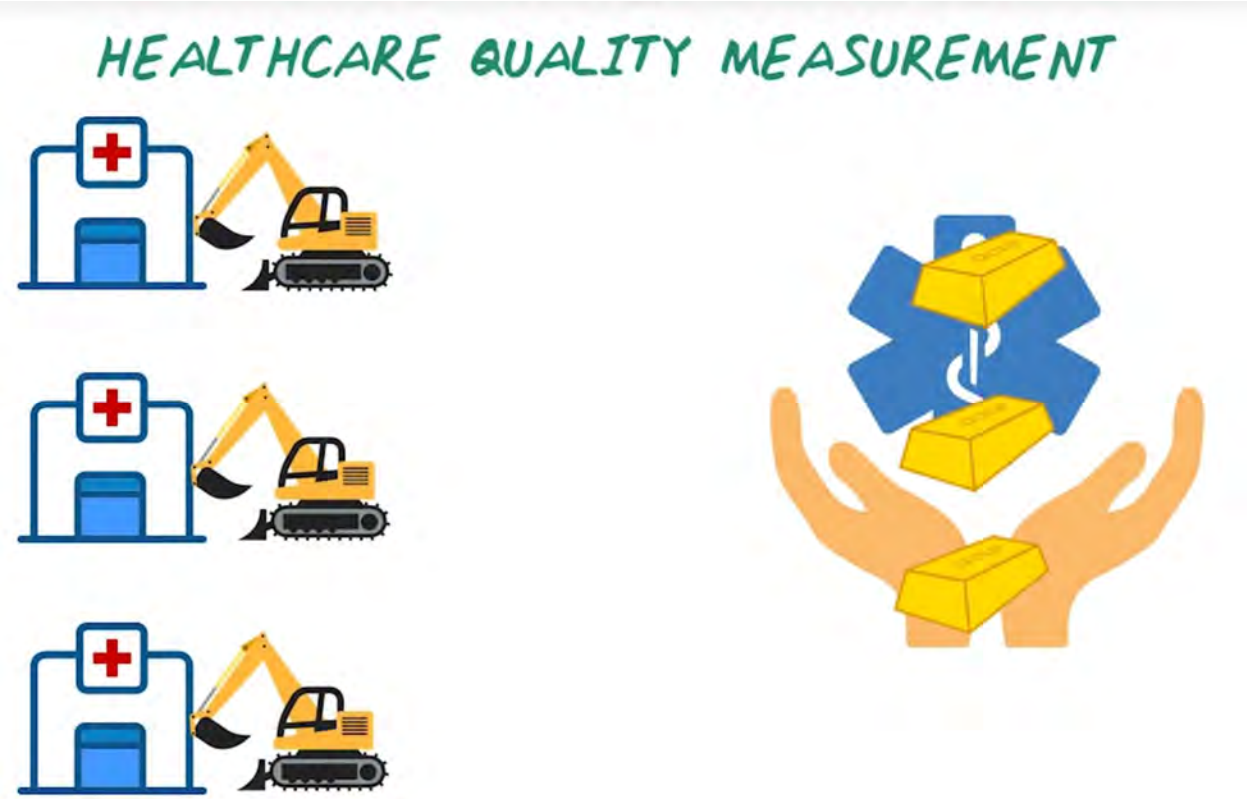
8. Pragmatic Clinical Trials



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. ~~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.~~

9. Healthcare Quality Measurement



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 centro 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.

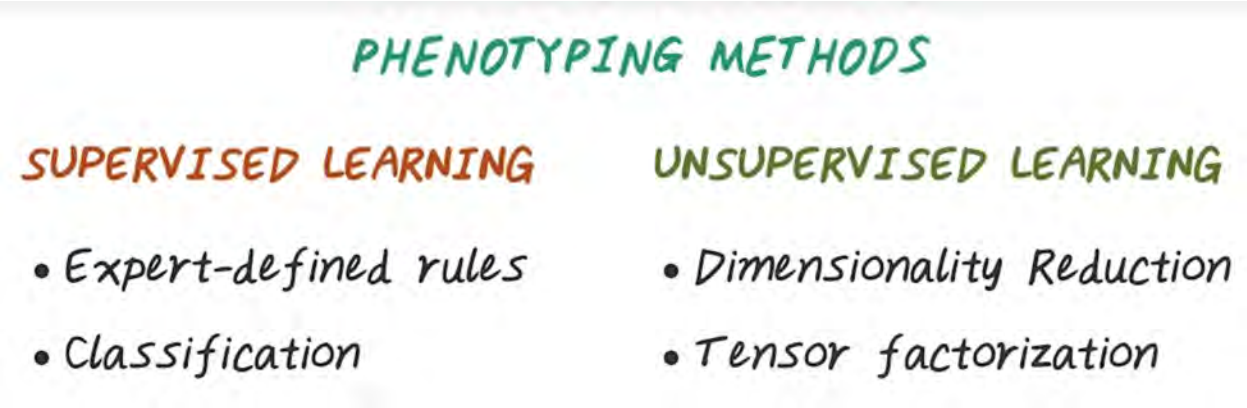
10. Phenotyping Methods Part 1

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 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. >> [LAUGH] 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. [LAUGH] >> [LAUGH] 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. >> [LAUGH] Wait, wait, wait, wait, I'm pixels now? >> As far as we can tell. >> That's fine. >> I however am not pixels. I know I'm not pixels. I'm pretty sure the rest of you are pixels. >> That's right. >> So I have a bunch of pixels and I might say male, or I might say female, or I might say dog, or I might say tree, but the point is I don't have a bunch of labels that say, dog, tree, male, or female, I just decide that pixels like this belong with pixels like this as opposed to pixels like something else that I'm pointing to behind me. >> Yeah, we're living in a world right now that is devoid of any other object. Chairs. >> Chairs, right. >> We got chairs. >> So these pixels are very different from those pixels, because of where they are relative to the other pixels. Exactly, right? So if you were- >> I'm not sure that's helping me understand unsupervised learning. >> Go outside and look at a crowd of people and try to decide how you might divide them up. Maybe you'll divide them up by ethnicity. Maybe you'll divide them up whether they have purposely shaven their hair in order to mock the bald, or whether they have curly hair. Maybe you'll divide them up by whether they have goatees- >> Facial hair. >> Or whether they have gray hair. There are lots of things that you might do in order- >> Did you just point at me and say gray hair? >> I was pointing, and your head happened to be there. >> Come on. Where's the gray hair? >> Right there. It's right where your split curl is. >> All right. >> Okay, so imagine you're dividing a world up that way. You can divide it up male and female. You can divide it up short and tall, wears hats, doesn't wear hats, all kinds of ways you can divide it up, and no one's telling you the right way to divide it up, at least not directly. That's unsupervised learning. That's description, because now, rather than having to send pixels of everyone or having to do a complete description of this crowd, you can say there were 57 males and 23 females exactly. Or there are mostly people with beards or

whatever. >> I like summarization for that. >> I like summarization for that, it's a nice concise description. >> Good. >> That's unsupervised learning. >> Very good. And that's different from supervised learning. >> It's different from supervised learning, and it's different in a couple of ways. One way that it's different is all of those ways that we could have divided up the world? In some sense, they're all equally good. So I can divide it by sex, or I can divide it by height, or I can divide it by clothing or whatever, and they're all equally good absence some other signal later telling you how you should be dividing up the world. But supervised learning directly tells you, here's a signal, this is what it ought to be, and that's how you train. They're very different. >> But I can see ways that unsupervised learning could be helpful in the supervised setting. Right, so if I do get a nice description, and it's the right kind of description, it might help me do the function approximation better. >> Right, so instead of taking pixels as input and labels like male or female, I could just simply take summarization of you, like, how much hair, relative, the height to weight, and various things like that might help me do it, that's right. And by the way, in practice, this turns out to be things like tensity estimation. We do end up turning it into statics at the end of the day. Often. >> It was statics from the beginning, but when you say density estimation- >> Yes. >> Are you saying I'm stupid? >> No. >> All right, so what is density estimation? >> Well, they'll have to take the class to find out. >> I see. >> Okay.

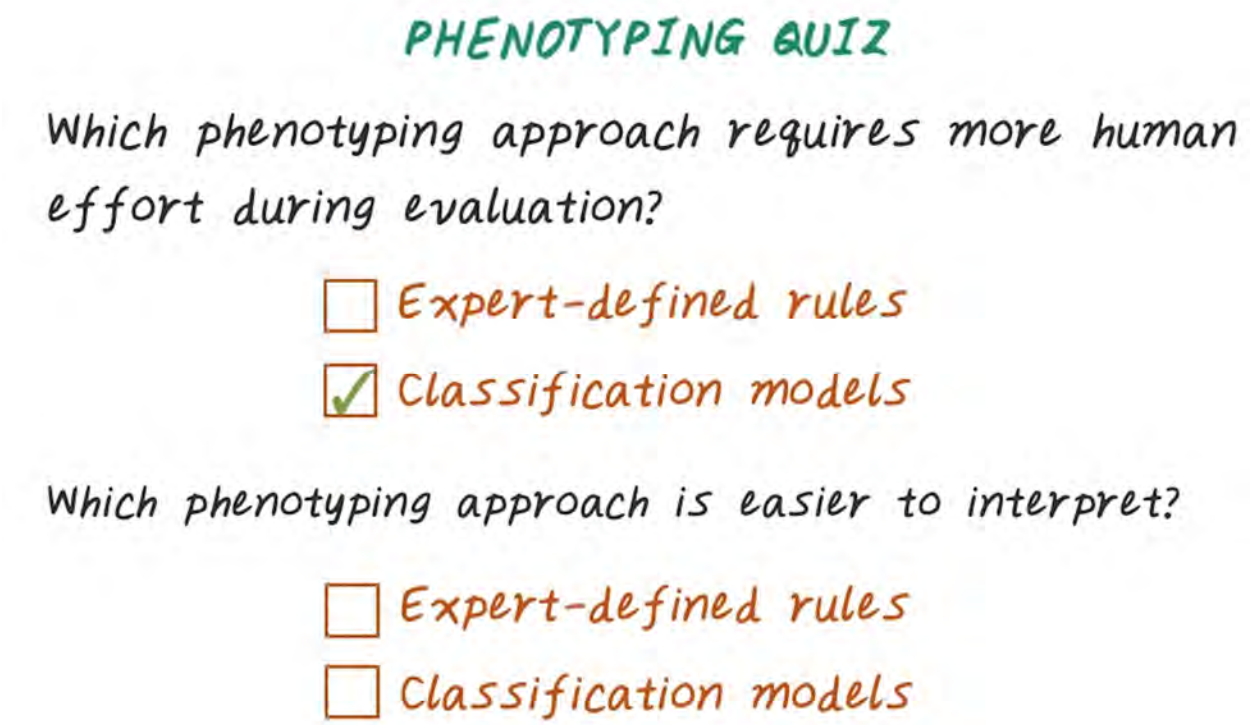
11. Phenotyping Methods Part 2



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.

12. Phenotyping Quiz



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?

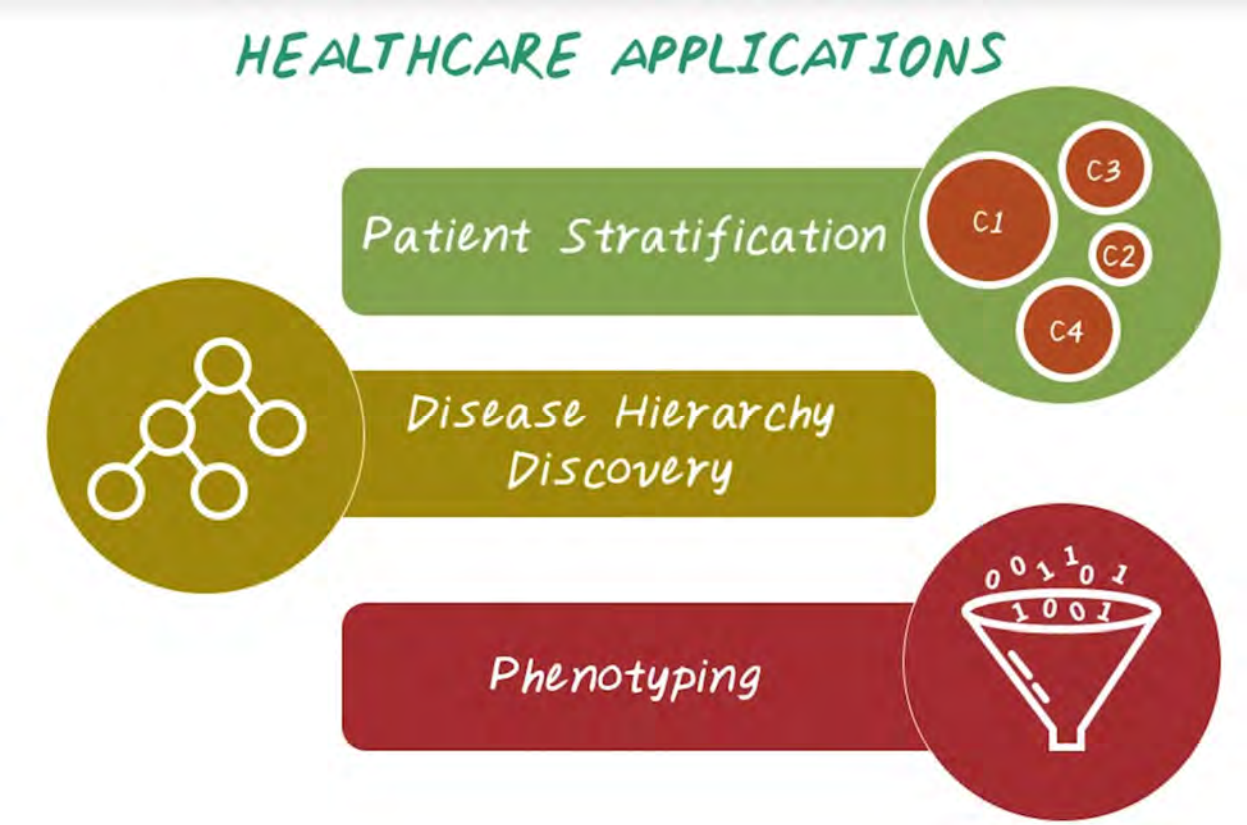
And the answer is classification models often require more human effort during evaluation, because they require a large amount of label training data in order to be a good model. However, the expert-defined rules, because the quality of those rules are often very good, so, during evaluation phase, it 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.

Clustering

1. Introduction to Clustering

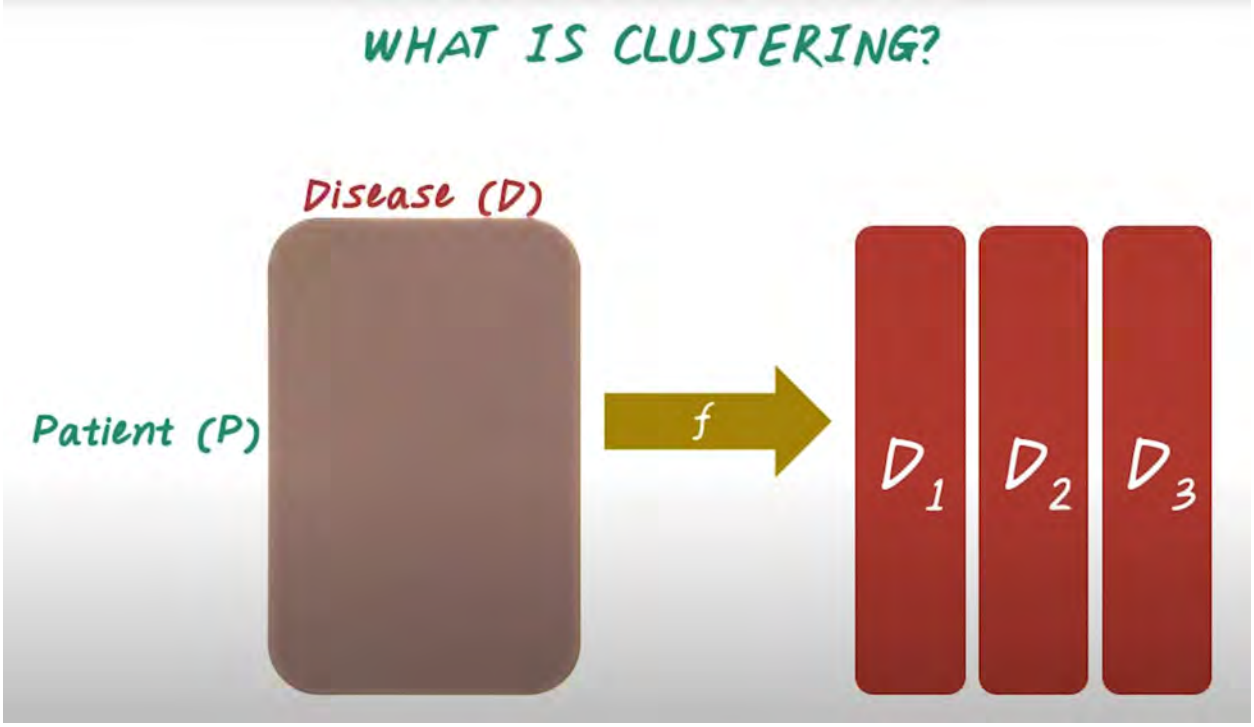
Now we're going to move on to a different method called clustering. In classification, we group data by label or categories that we already knew. But in clustering, we want to discover those labels or categories form raw data. We'll start by defining clustering. Then we'll describe some algorithms for 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.

2. Healthcare Applications



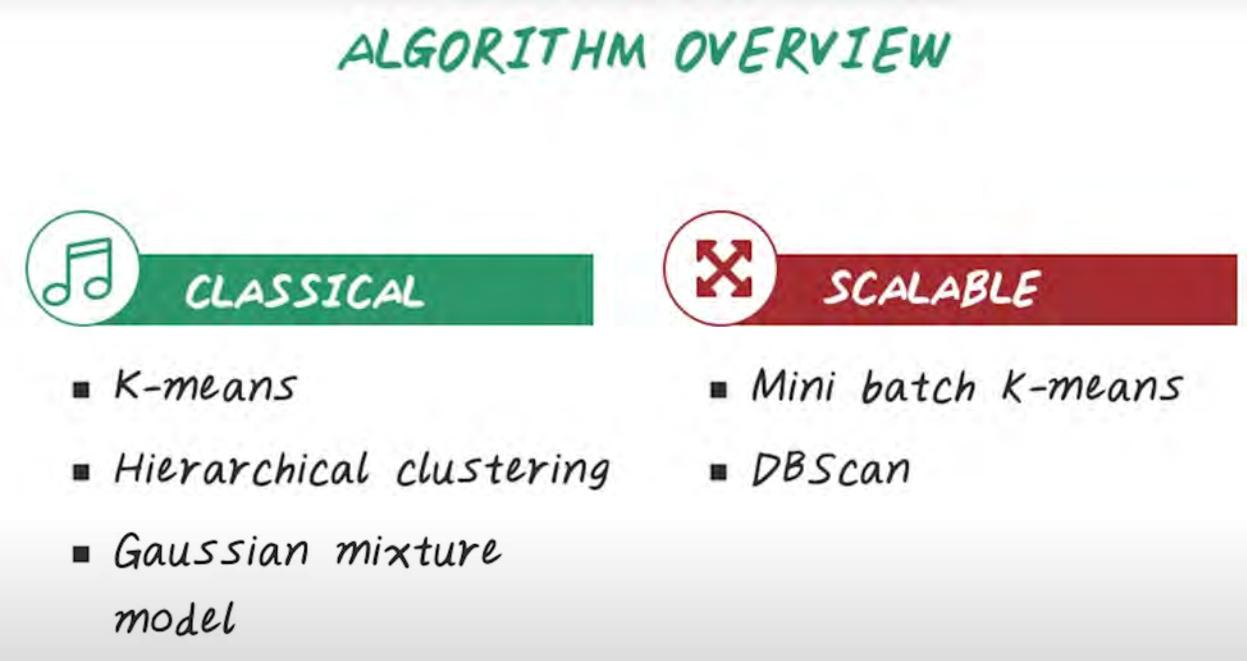
There are many healthcare applications for clustering. For example, patient stratification is about grouping patients into clusters such that patient within the same cluster share many common characteristics, and patients across cluster share very few common characteristics. And disease hierarchy discovery 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.

3. What is Clustering



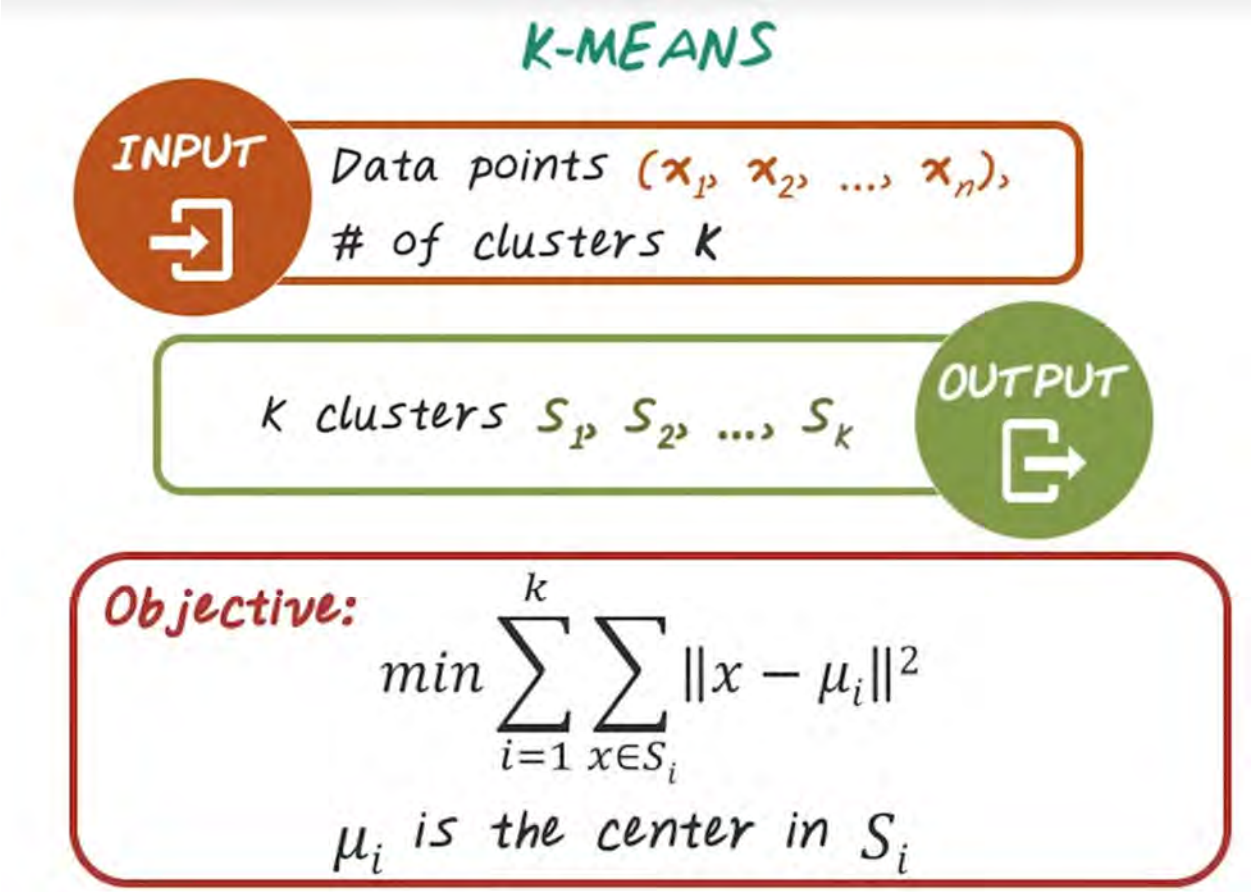
So what is clustering? Let's illustrate clustering using the following example. Imagine we have a patient by disease matrix, where the columns are all the different diseases, and rows are different patients. And other elements indicate whether a specific patient has a specific disease. If we want to 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.

4. Algorithm Overview



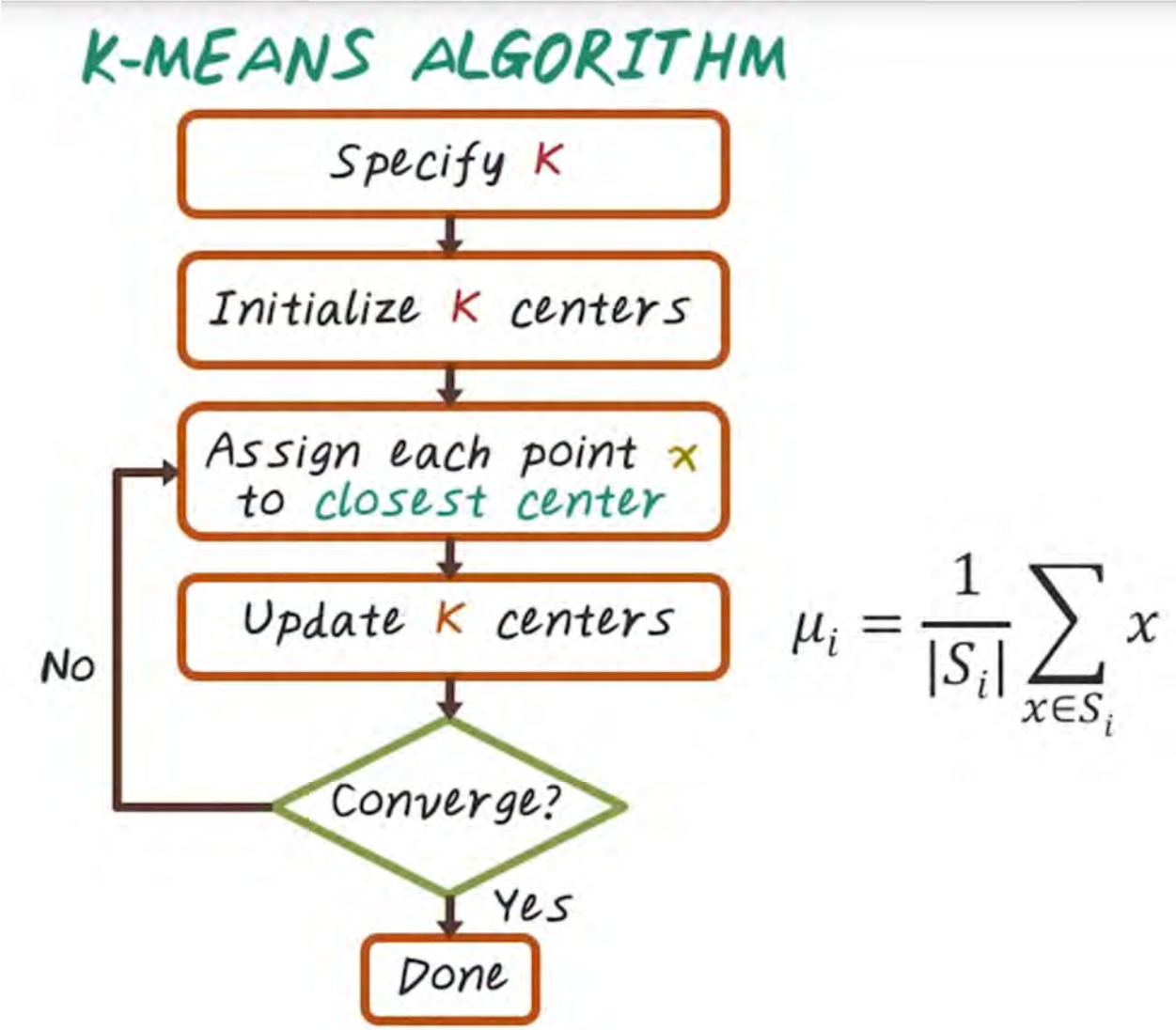
In this class we'll introduce two sets of clustering algorithms. The classical clustering algorithms and the scalable clustering algorithms. For classical algorithms, we'll discuss K-means, hierarchical clusterings, and Gaussian mixture model. For scalable algorithms, we'll introduce mini batch K-means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

5. K Means



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.

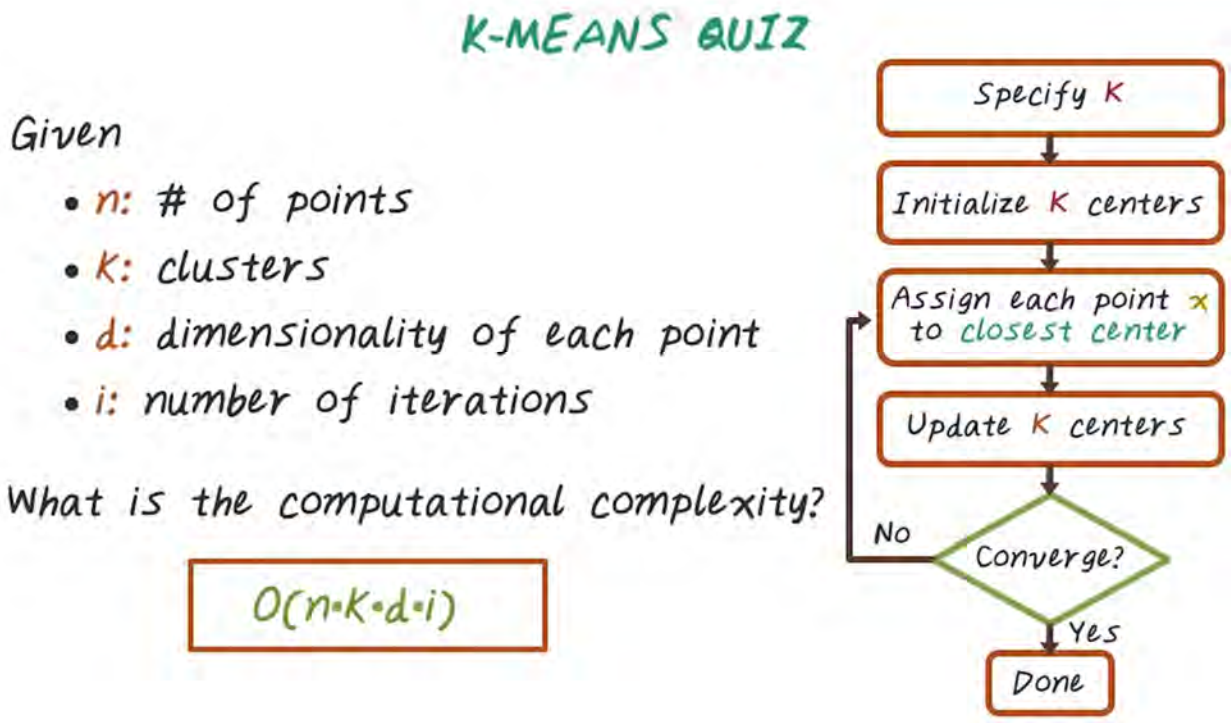
Then we iterate this process over three iterations, and two had converged. This is the final result when we want K means with setup points, with K equal to two.



6. K Means Quiz

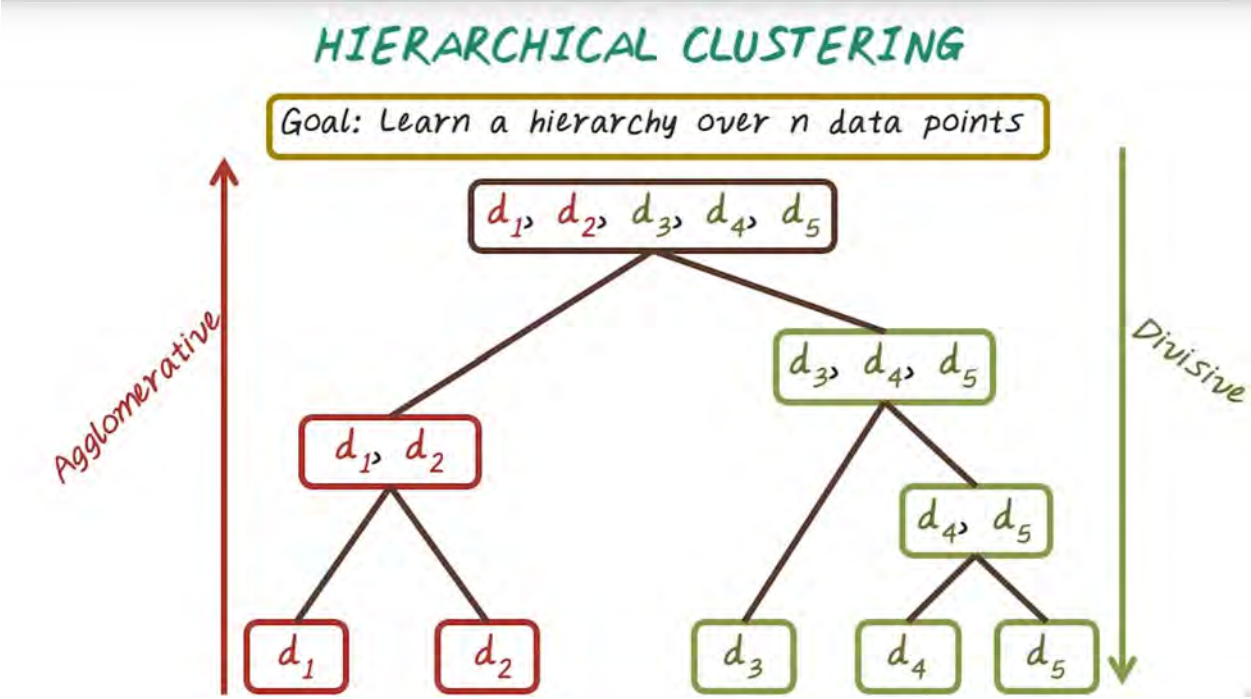
Now let's do a quiz on K-means. Given n is the number of data points, k is the number of clusters, and d is dimensionality of each data points, and i is the number of iteration of the K-means algorithm, what is the computational complexity of K-means? This is the algorithmic

illustration to help you find the answer and put your answer in this box.



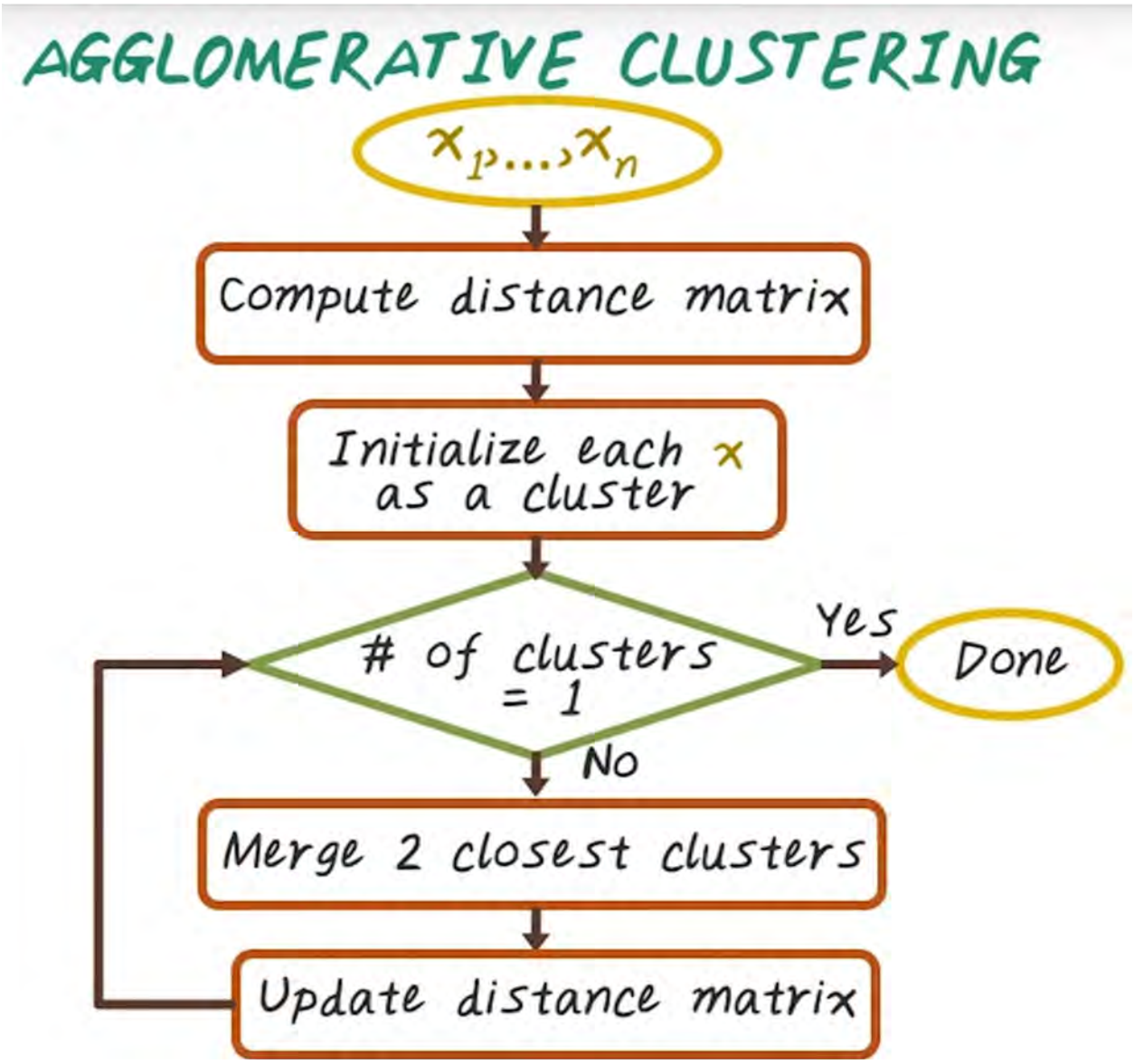
And the answer is big O of n times k times d times i. Now let's see how we got this answer. When we run k-means algorithm most of the computation goes to clustering assignment and center update. For clustering assignment, we need to compare end data points to K centers. So that takes n times k comparisons. 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.

7. Hierarchical Clustering



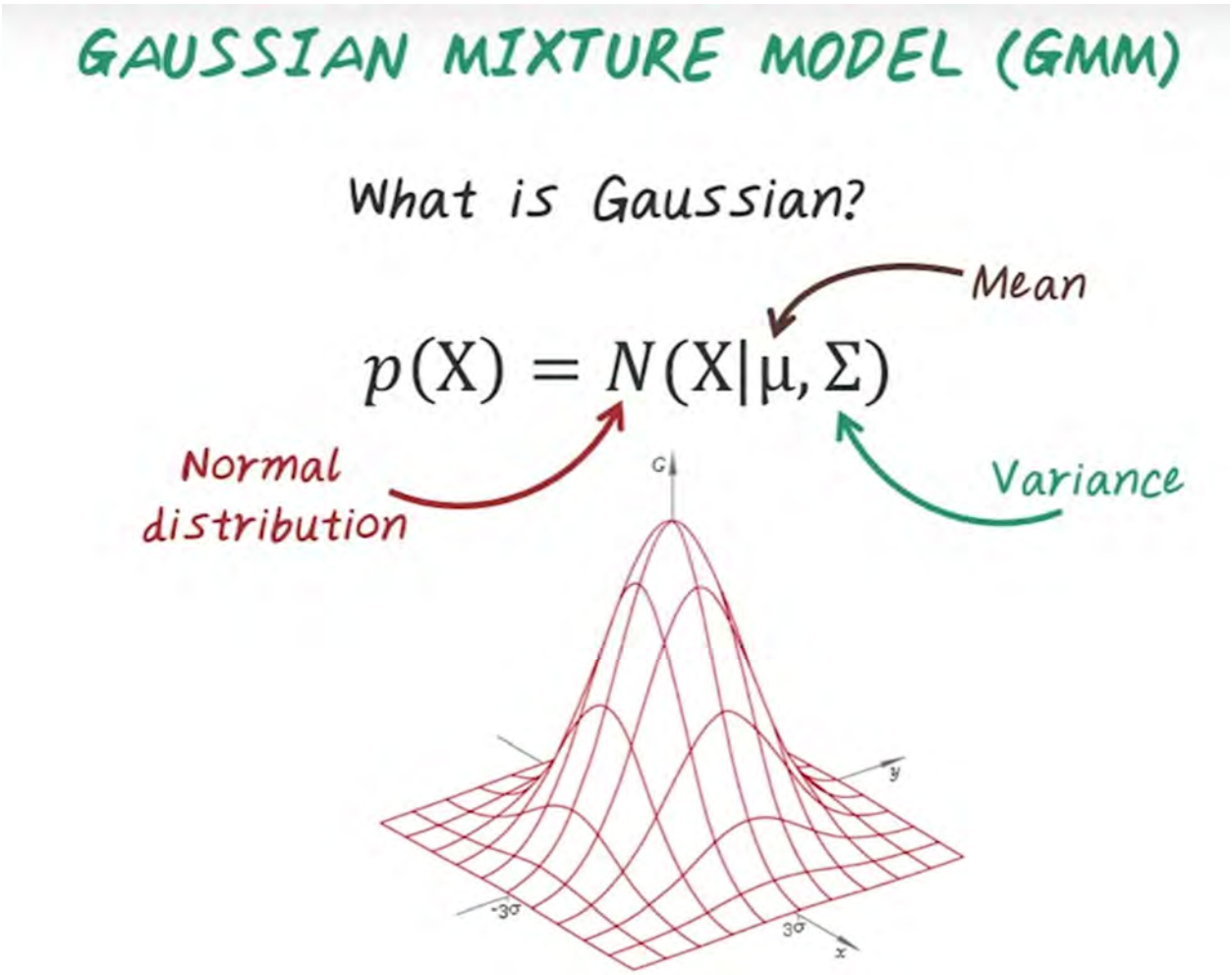
Next, let's introduce hierarchical clustering. The goal of hierarchical clustering is to learn hierarchy over a set of data points. For example, given five different diseases, we want to construct such a hierarchy so that similar diseases are grouped together into this tree structure. There are two main 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 grouped 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.

8. Agglomerative Clustering



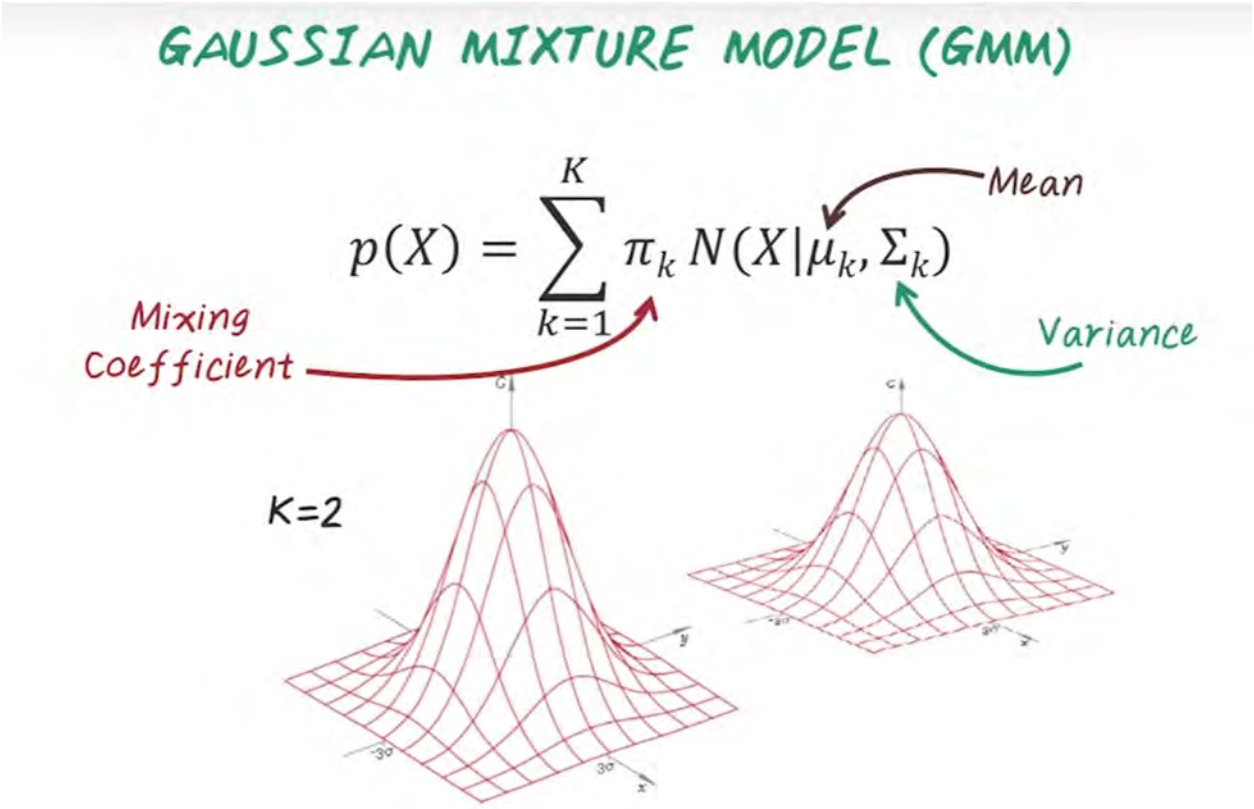
Now let's talk about agglomerative clustering algorithm. We'll start with a set of data points x1 to xn. First we compute all paired distance matrix. This will be a n by n matrix, and every element in this matrix corresponding to the distance between the pair of points. Then we initialize each data points 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.

9. Gaussian Mixture Model



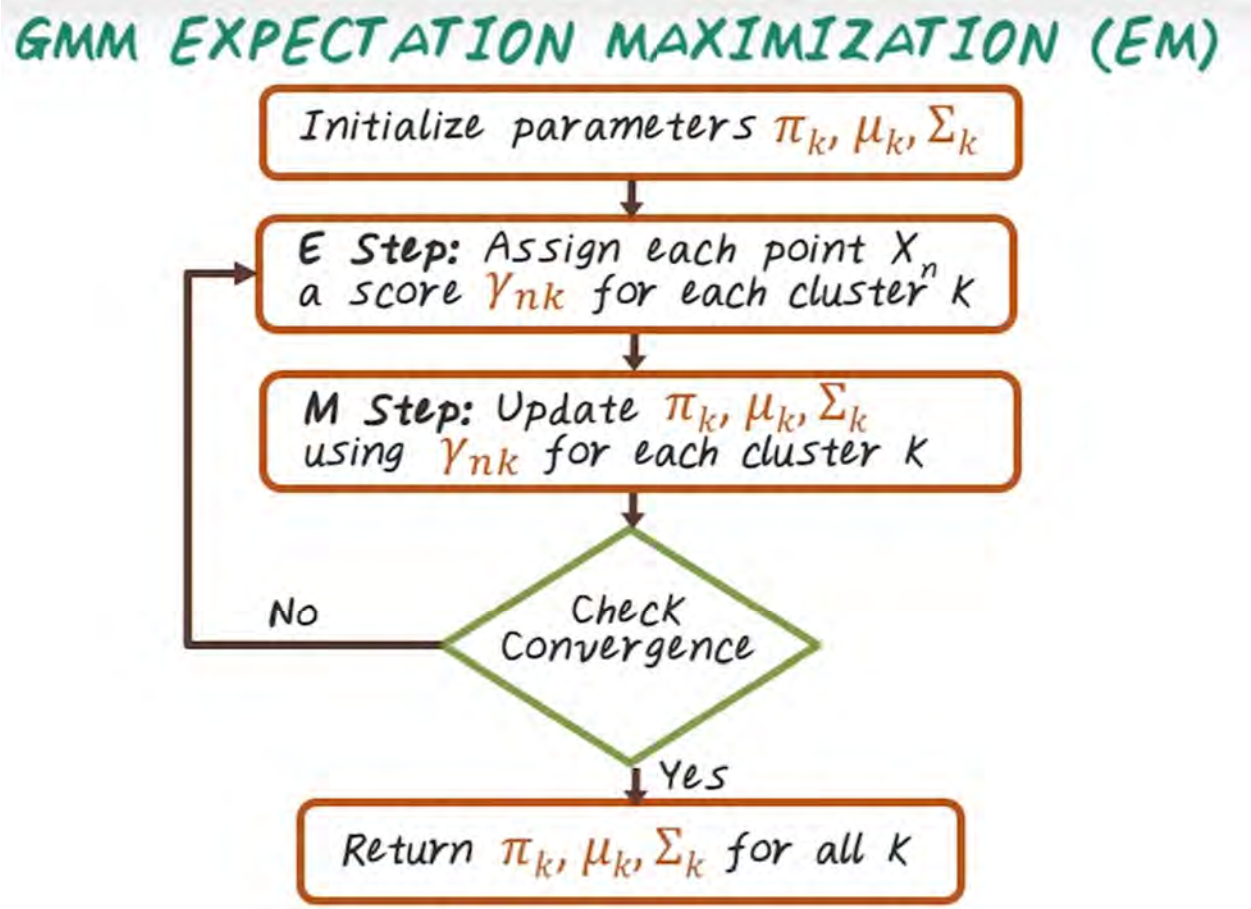
Next we'll talk about Gaussian mixture model. So far we have talked about two clustering algorithm, K-means and hierarchical clustering. They are both hard clustering method, which means every data points only belong to a single cluster, and Gaussian mixture model is a soft clustering method. >> Soft? >> 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.



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.

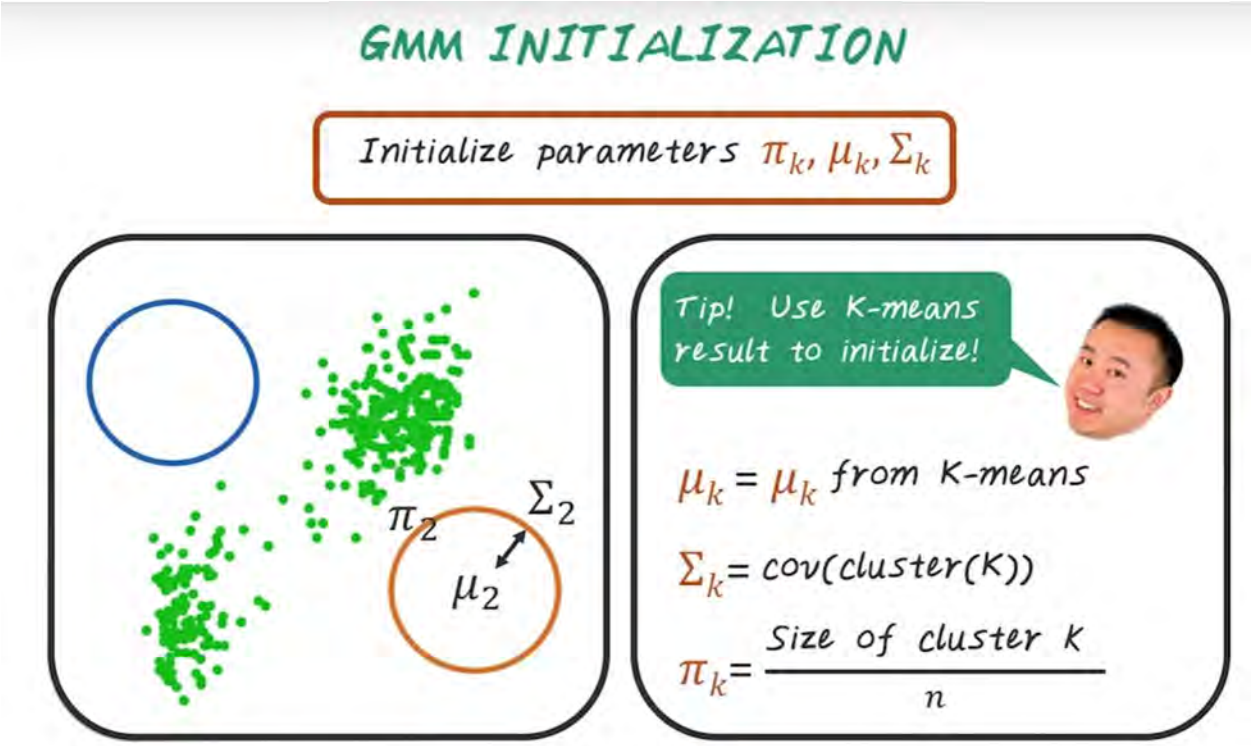
10. GMM Expectation Maximization

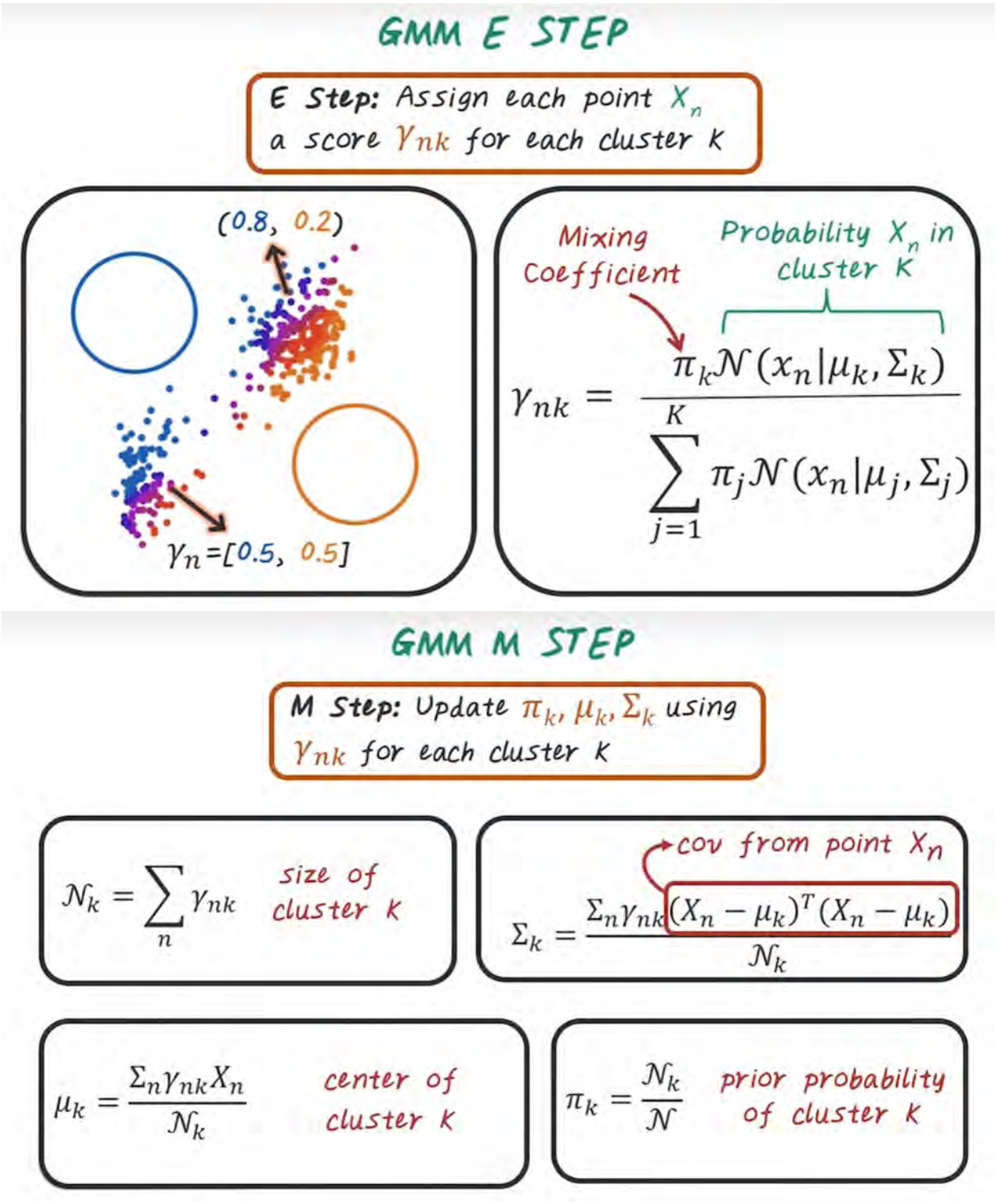


11. GMM Steps

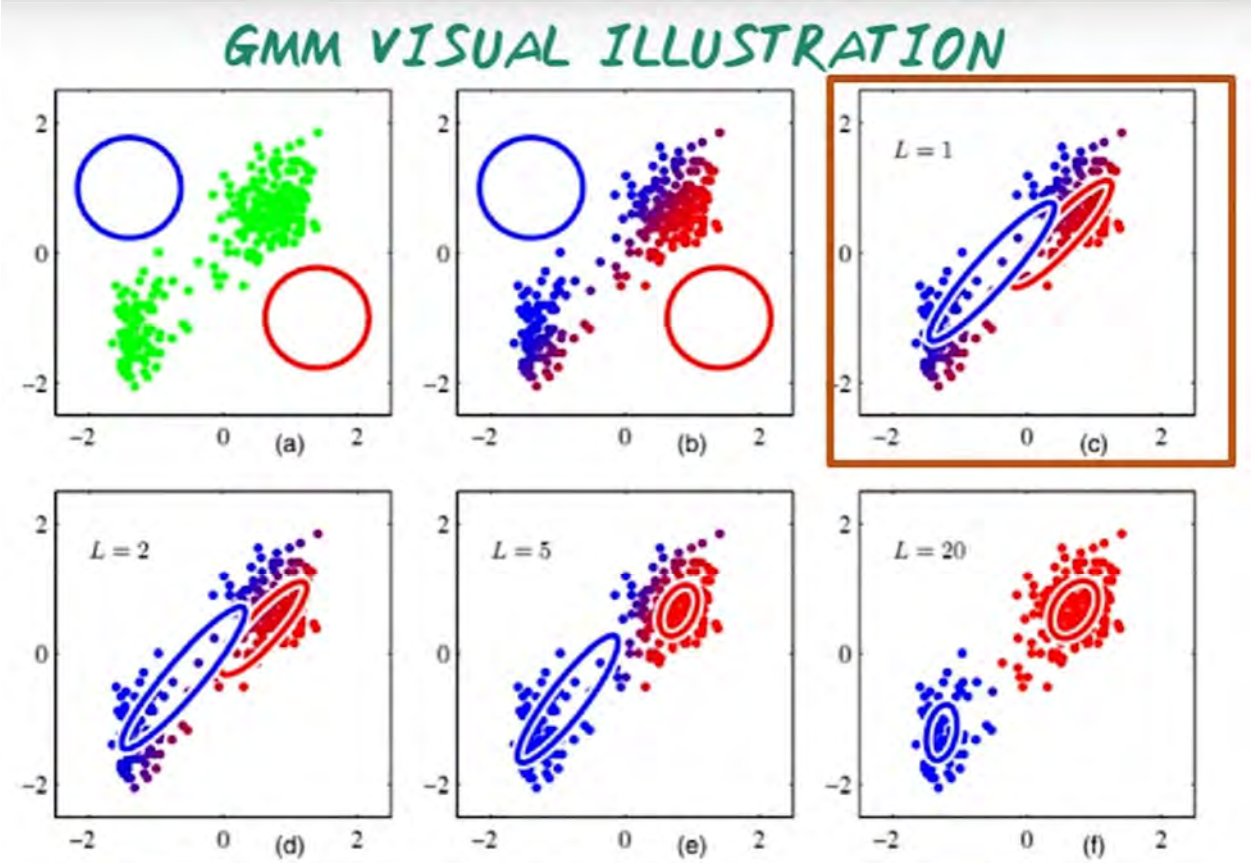
In order to run Algorithms for Gaussian mixture model, we have to initialize all those parameters. In particular, we have to find the mixing coefficients, the centers, and variance for each clusters. And this initialization step is very important. A bad initialization can cause many subsequent iterations 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 going 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.



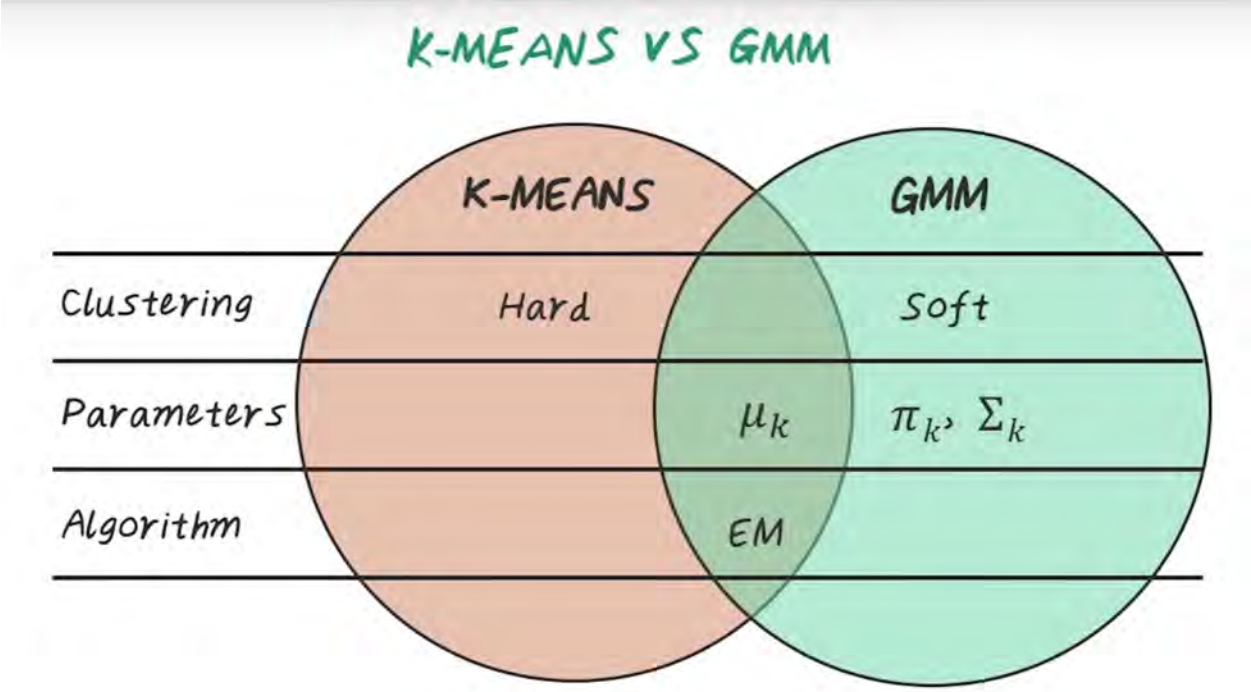


12. GMM Visual Illustration



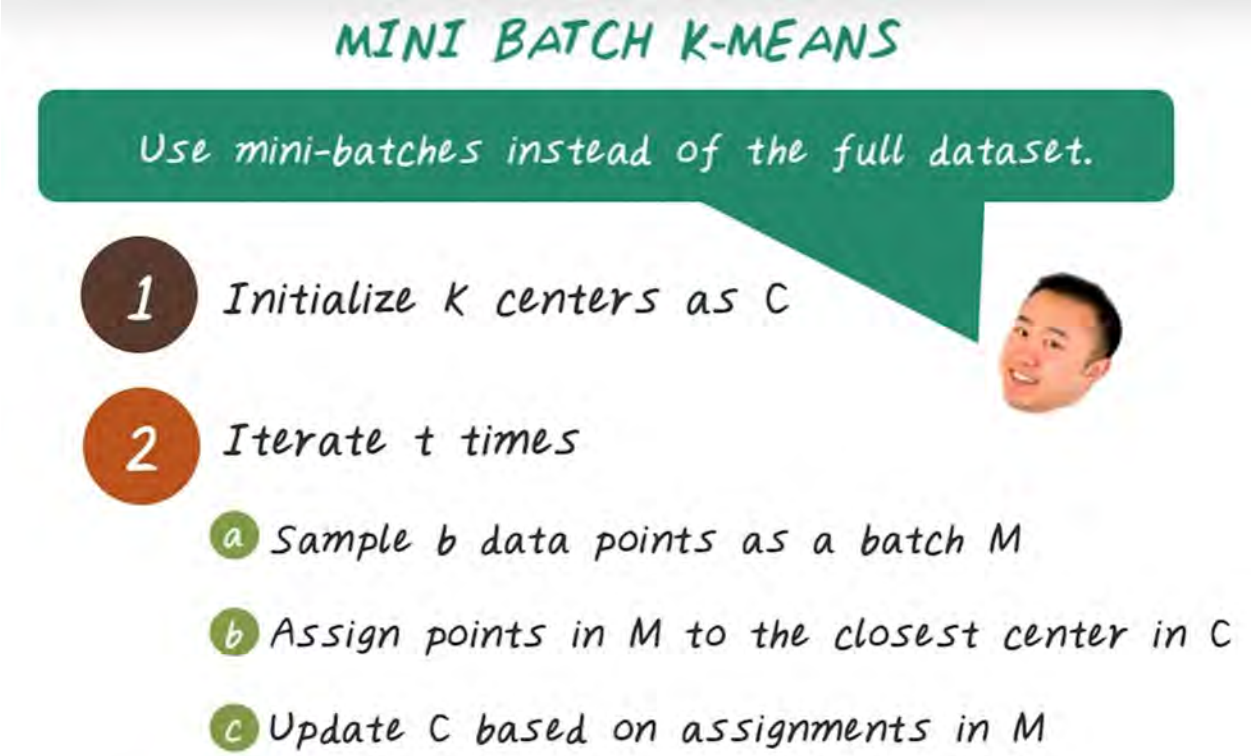
Now let's visually illustrate how Gaussian mixture model works. The goal is to run Gaussian mixture model over this set of points. We first initialize two Gaussians indicated by the blue and red circles. Then we compute assignment scores for each data points. The bluer points indicate the assignment 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

13. K Means Vs GMM

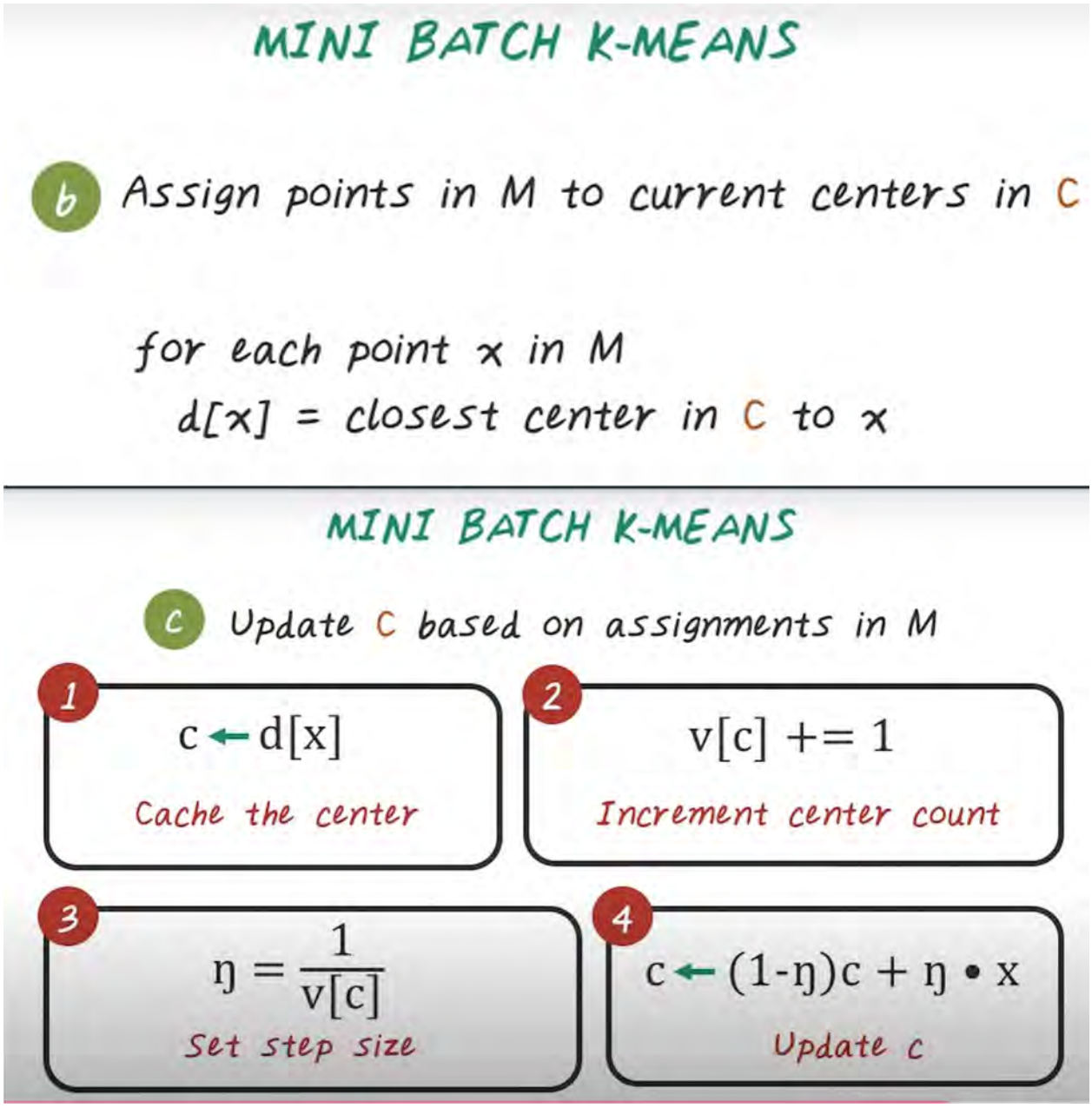


Now let's compare two clustering method. K-means and Gaussian Mixture Model. In term of clustering algorithm itself, K-mean is a hard clustering algorithm. Each data points only belong to one cluster. Gaussian Mixture is a soft clustering algorithm. Each data points can belong to multiple clusters with 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.

14. Mini Batch K Means

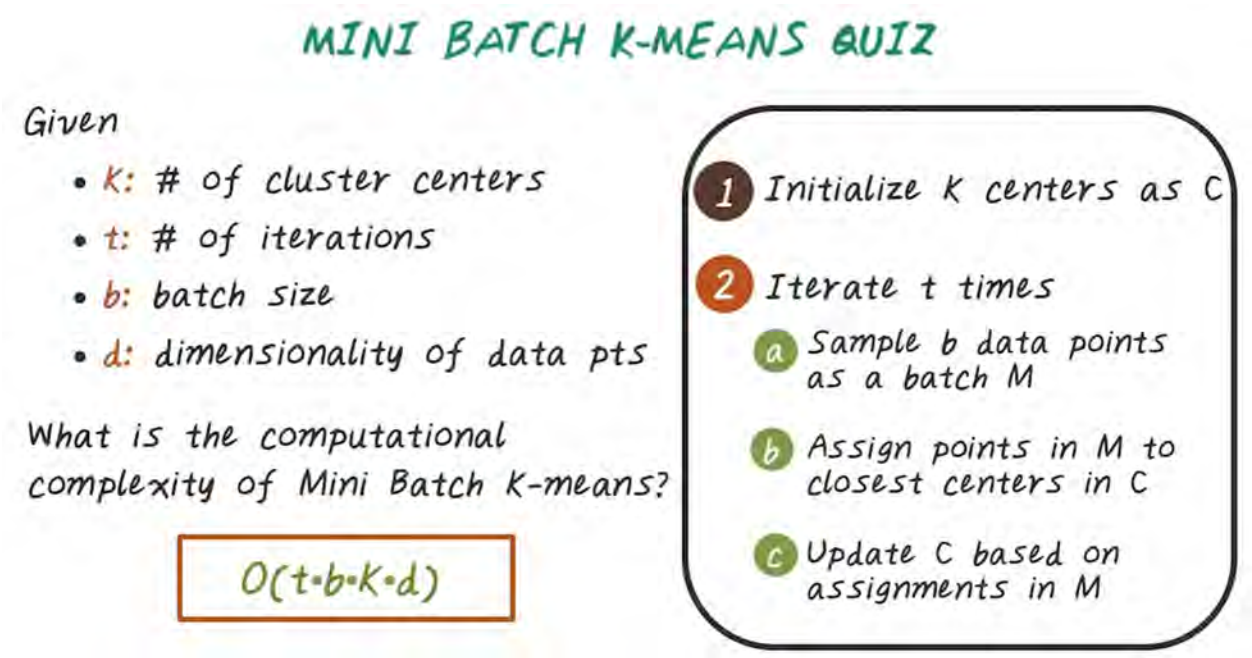


So far, we'll have talked about three classical clustering algorithm. K-mean, hierarchical clustering, and Gaussian mixture model. Next, we'll describe two scalable clustering algorithm. Mini batch K-means and DBScan. So one problem with K-mean's algorithm is that it needs to assign all the data points 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.



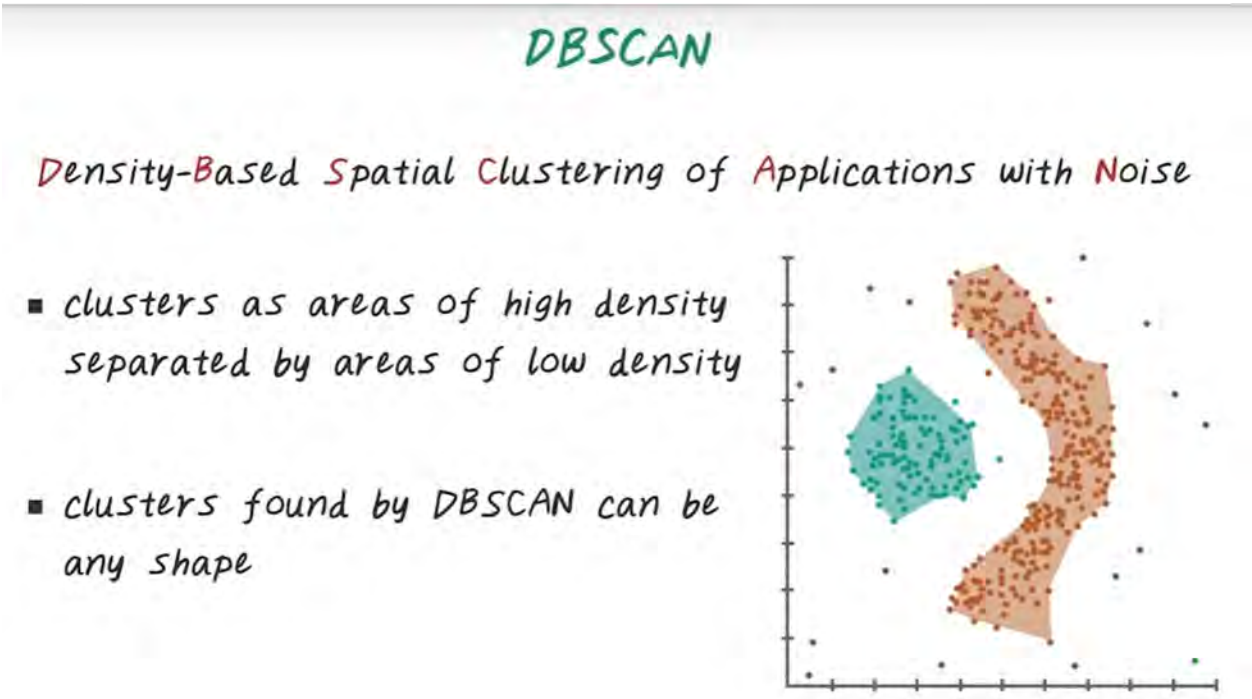
15. Mini Batch K Means Quiz

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.



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.

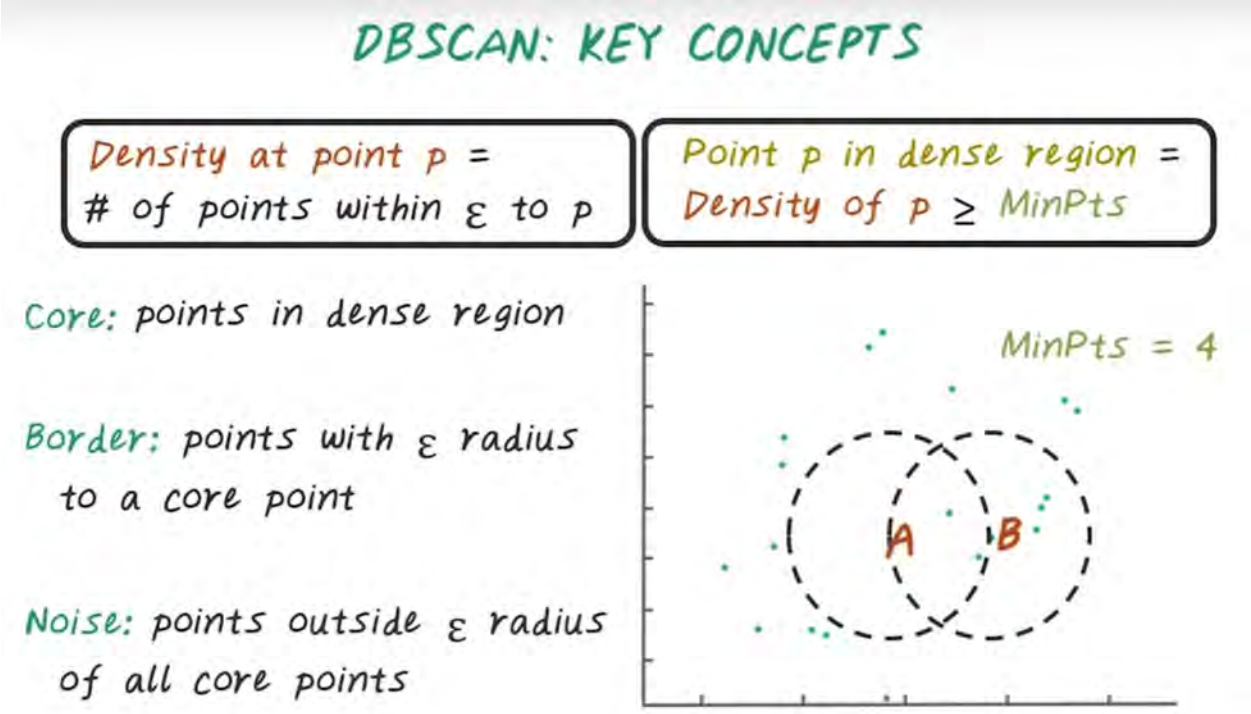
16. DBScan



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

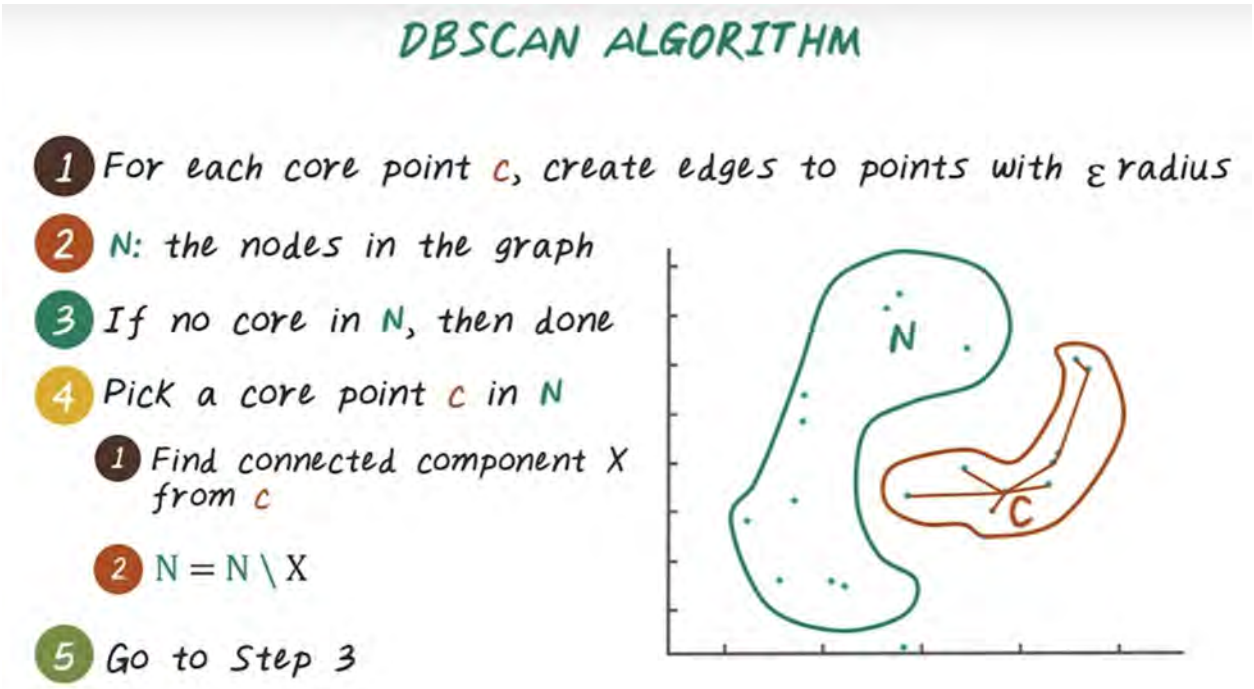
high densities separated by areas of low density. As a result, oftentimes the cluster found by DBSCAN can be of any shape. Here's an example. If we run DBSCAN on this data set, we'll have two clusters. The blue areas is one cluster and the red area is another cluster. Both are defined by high-density regions, and they are separated by low-density regions. More details about the algorithm can be found in the instructor notes.

17. DBScan Key Concepts



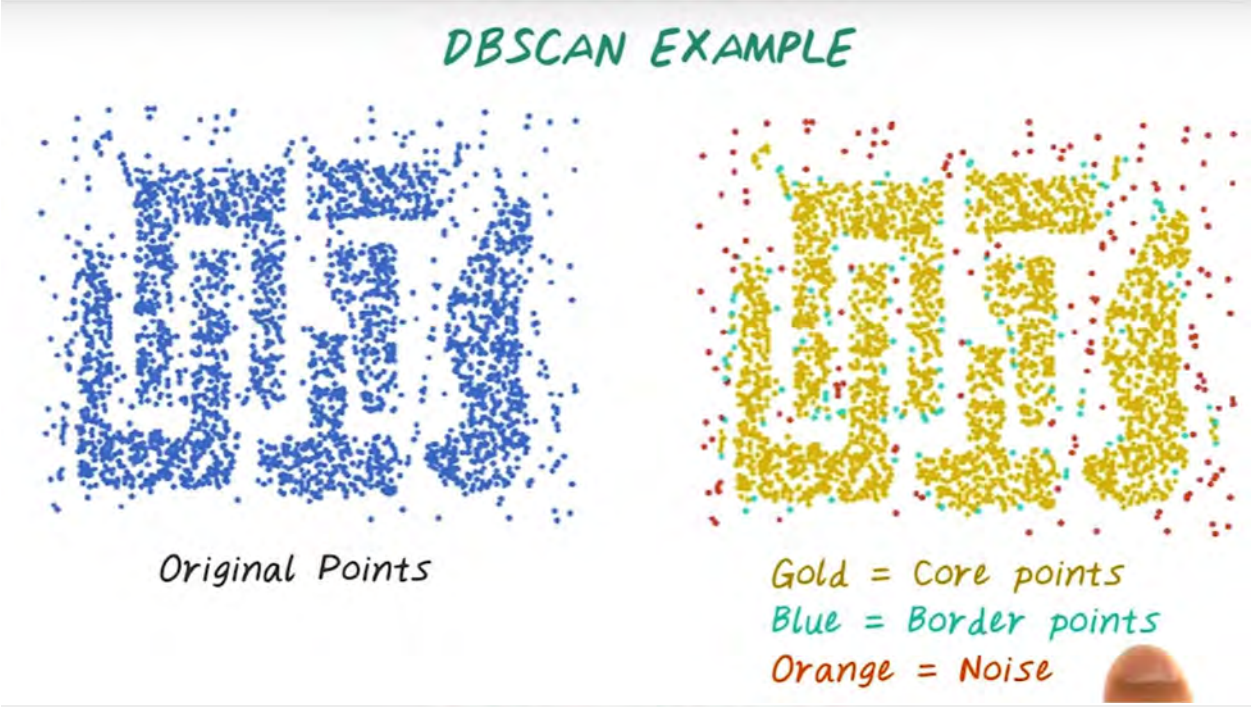
In order to explain DBSCAN algorithm we have to introduce some key concepts. First, how do we measure the density? In DBSCAN, the density at a point p is defined as the number of points within absolute distance to p. For example, here are a set of points in two dimensional space. This point A was equal 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.

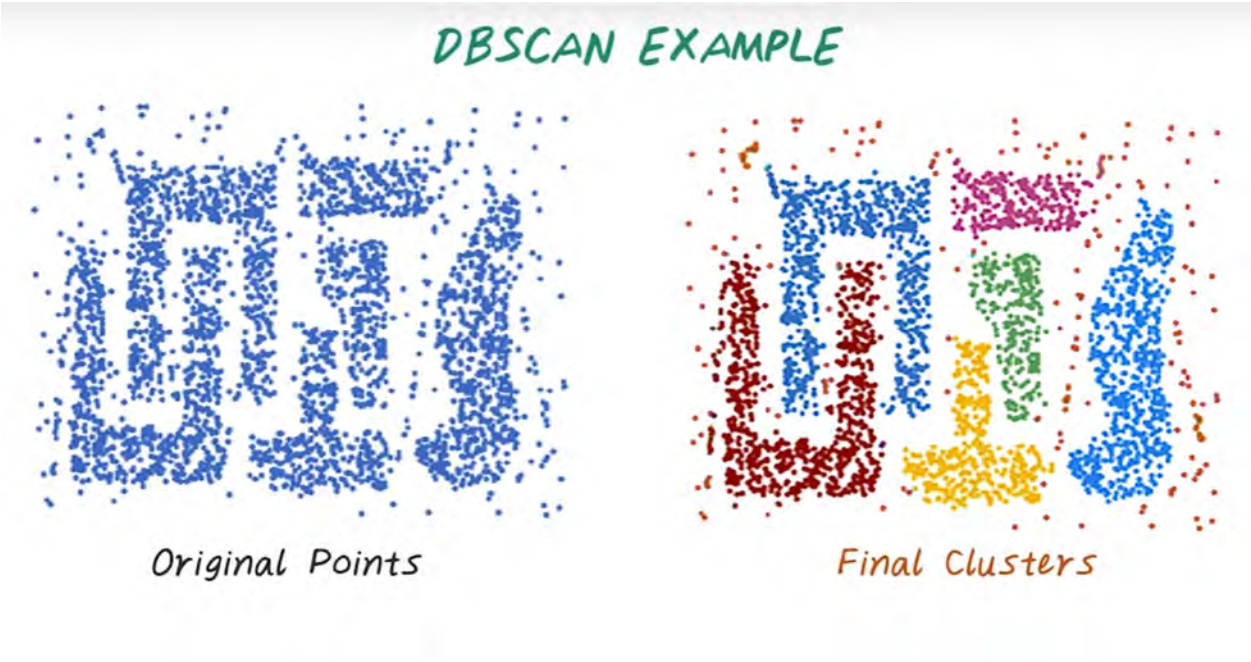
18. DBScan Algorithm



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.

19. DBScan Example

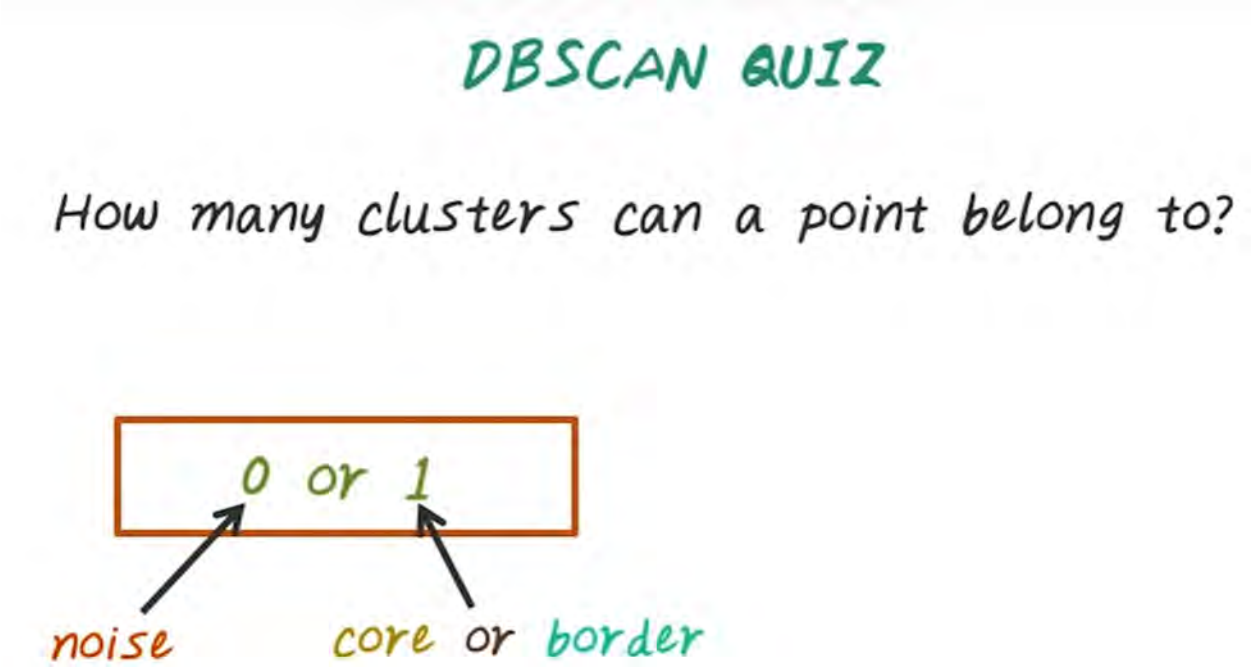




Now let's look at some examples of DBSCAN giving a set of two dimensional data points. Look like this. We want to find a set of clusters using DBSCAN. The first step of DBSCAN will try to classify all those data points into this three categories. The core points are those gold color points. And the border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

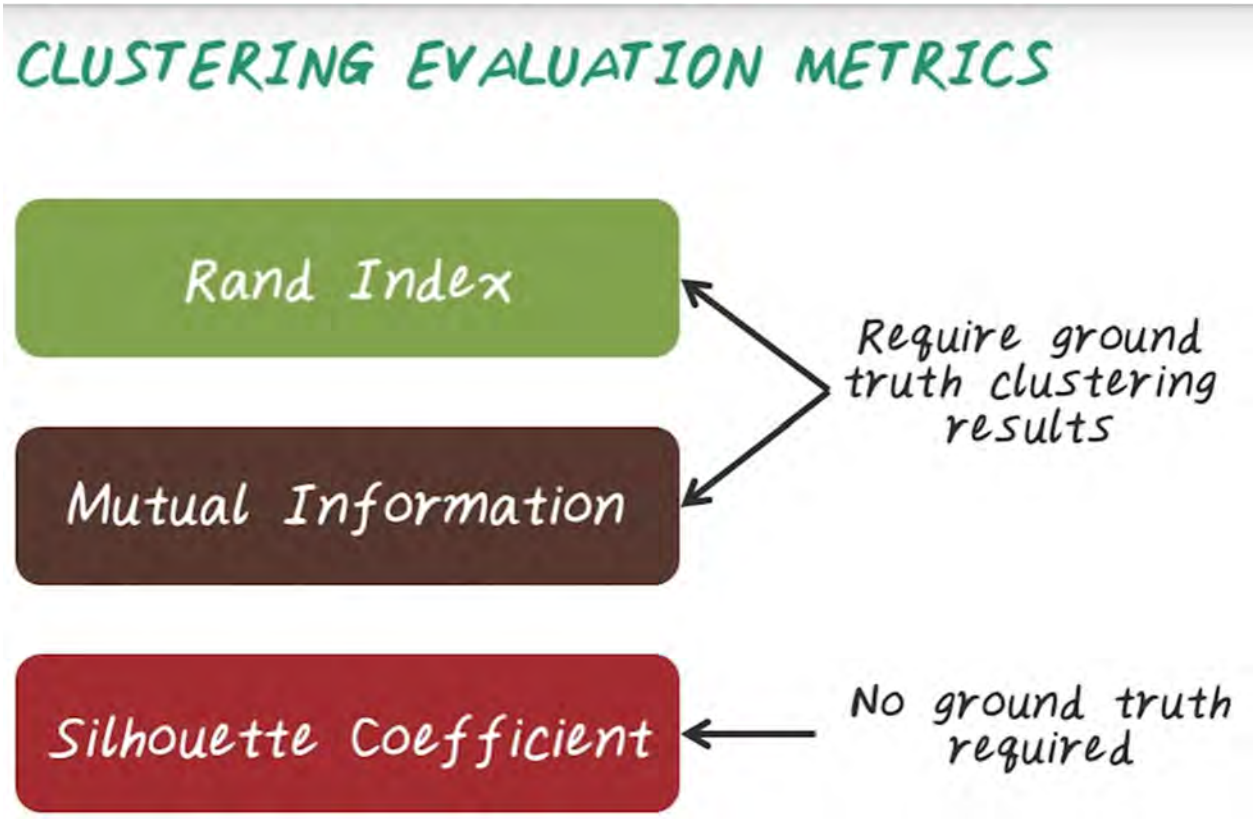
20. DBScan Quiz

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.



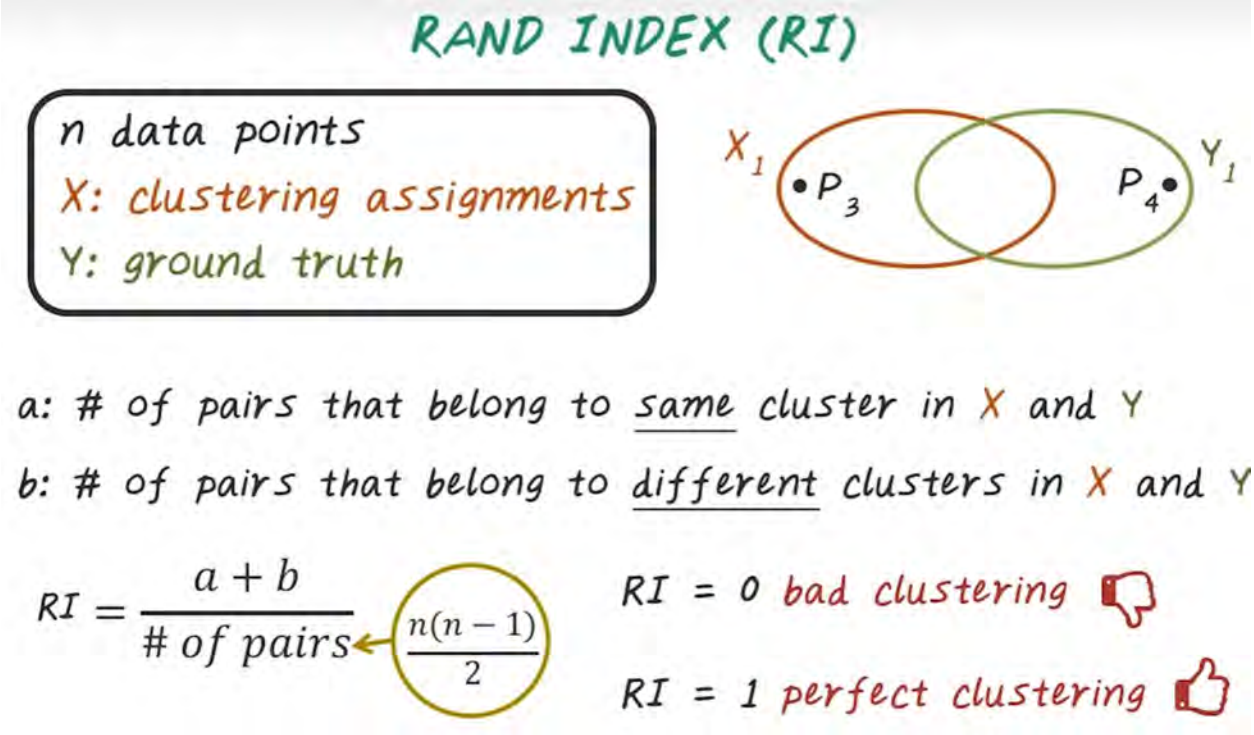
And the answer is 0 or 1. If the data point is a noise, it won't belong to any cluster, so it will be 0. If the data point is a core points or a border point, then it will belong to 1 cluster. So the key here is all the noises are not assigned to any cluster.

21. Clustering Evaluation Metrics



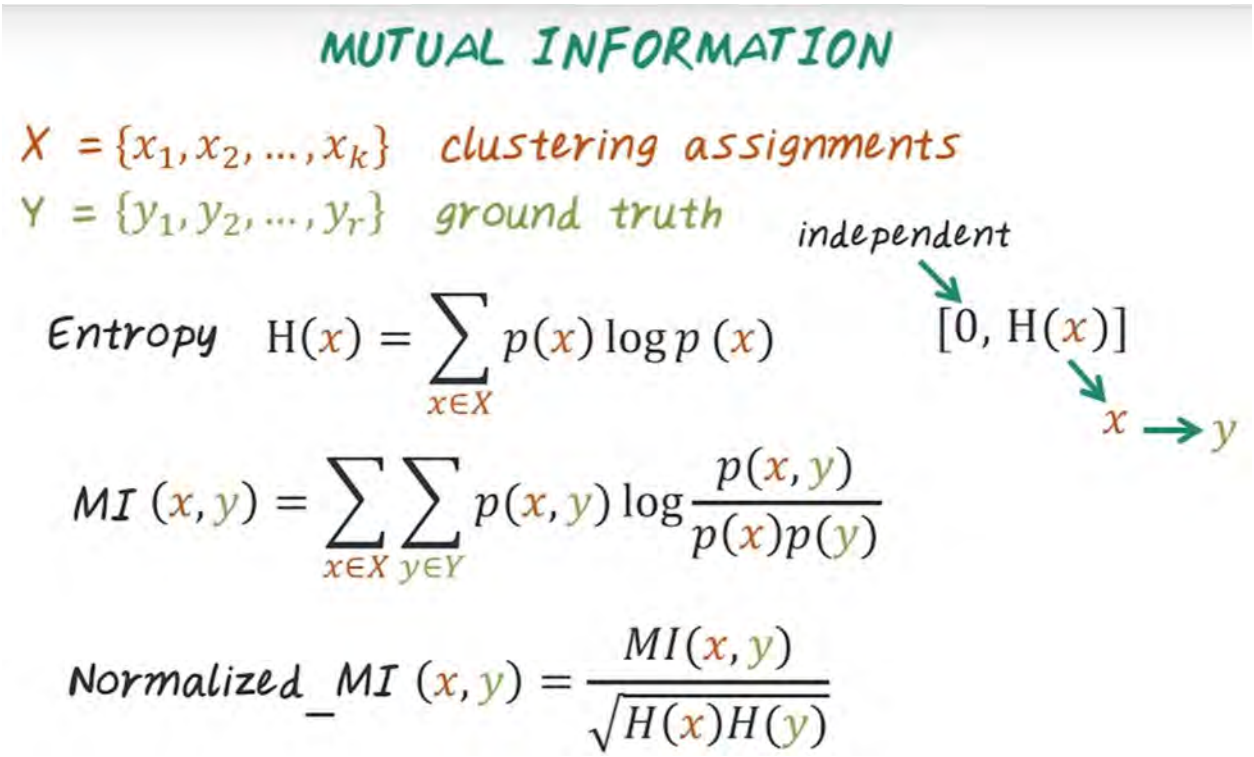
So far we have talked about clustering algorithms. Next we introduce a set of evaluation metrics for clustering. In particular, we'll talk about rand index, mutual information, and silhouette coefficient. And rand index and mutual information requires we know the ground truths of the clustering result.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

22. Rand Index



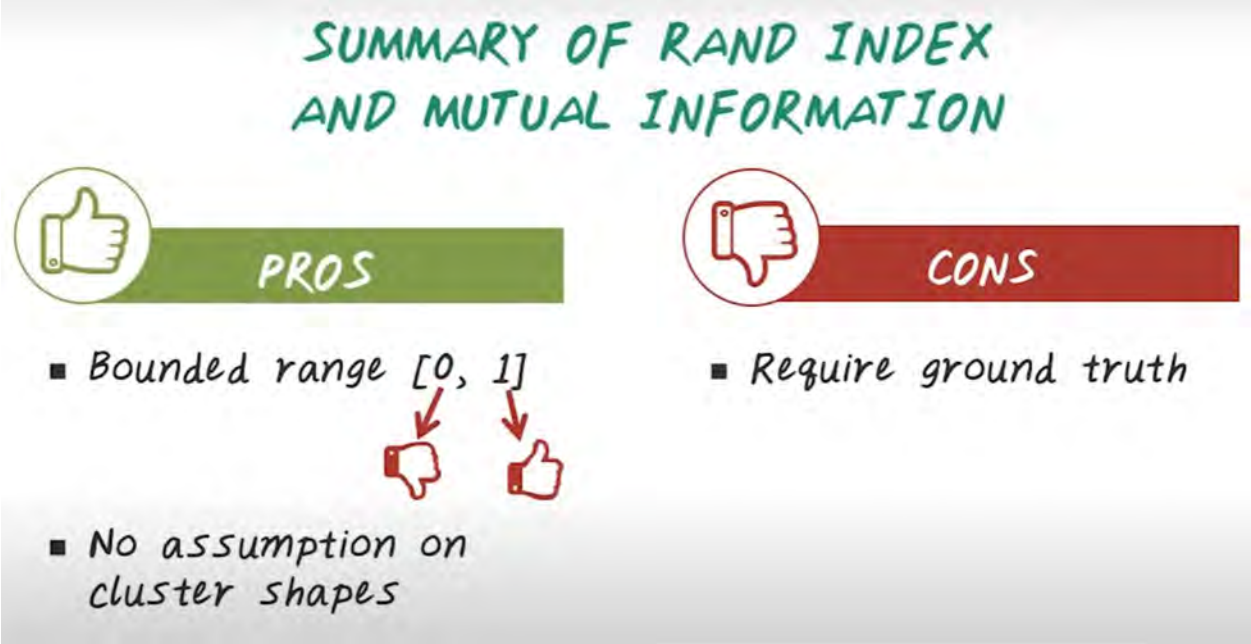
Now let's talk about Rand Index or RI. Given n data points, let's say X is the clustering assignment from the algorithm and y is the ground truth. In here, we illustrate a cluster X1 that comes from the algorithm and cluster Y1 come from the ground truth. Then we compute the term a, which is the number 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.

23. Mutual Information



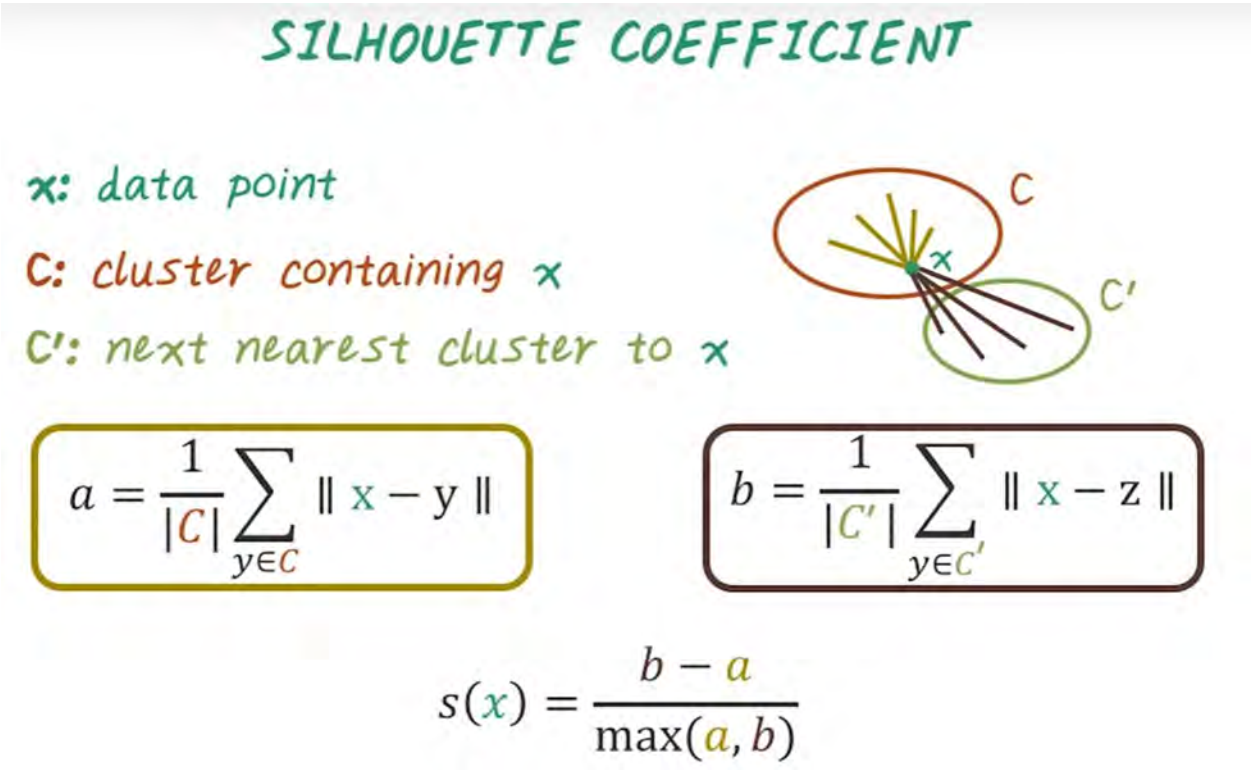
Like grand index, mutual information is another way to measure clustering quality. And mutual information is a concept from information theory which measures the mutual dependency of two random variables. In this case, the two random variables are clustering assignment x and the ground truth assignment 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.

24. Summary of RI and MI



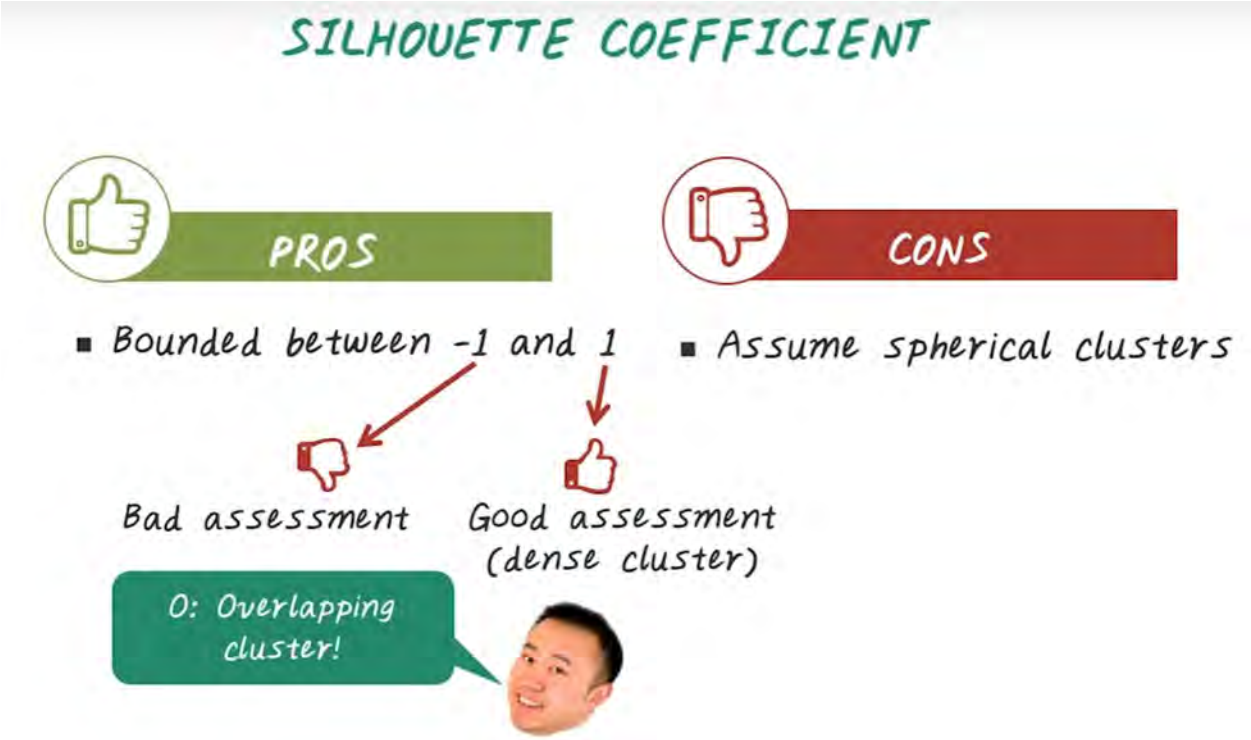
There are a lot of commonality between Rand index and mutual information. So what are the pros and cons of this Q evaluation metric. They both provided bounded ranges when the metric is close to 0, which means bad clustering assignment. When it's close to one, means good cluster assignment. And there's 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.

25. Silhouette Coefficient



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.

26. Silhouette Coefficient Pros and Cons



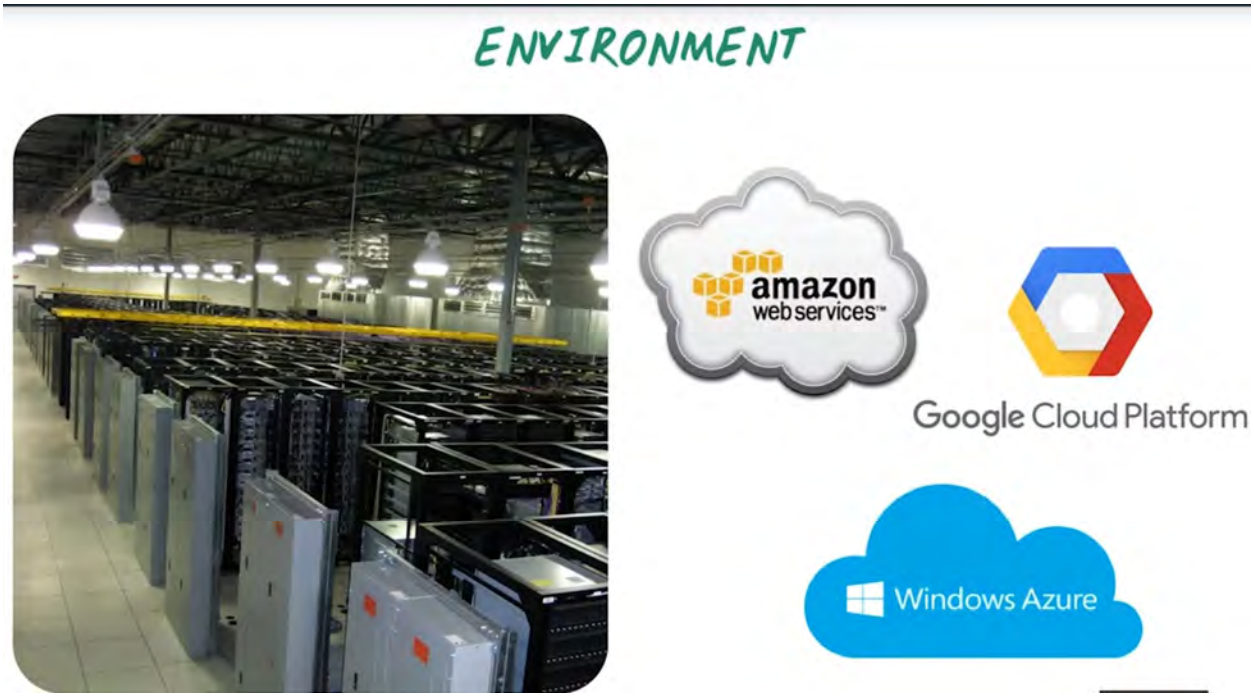
In summary, here are the pros and cons for a silhouette coefficient. Again, it provided a bounded range between minus 1 and 1. And minus 1 means very bad clustering assignment, 1 means very good clustering assignment, and 0, in this case, means overlapping cluster. The limitation here is silhouette coefficient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

Spark

1. 1. Introduction to Spark

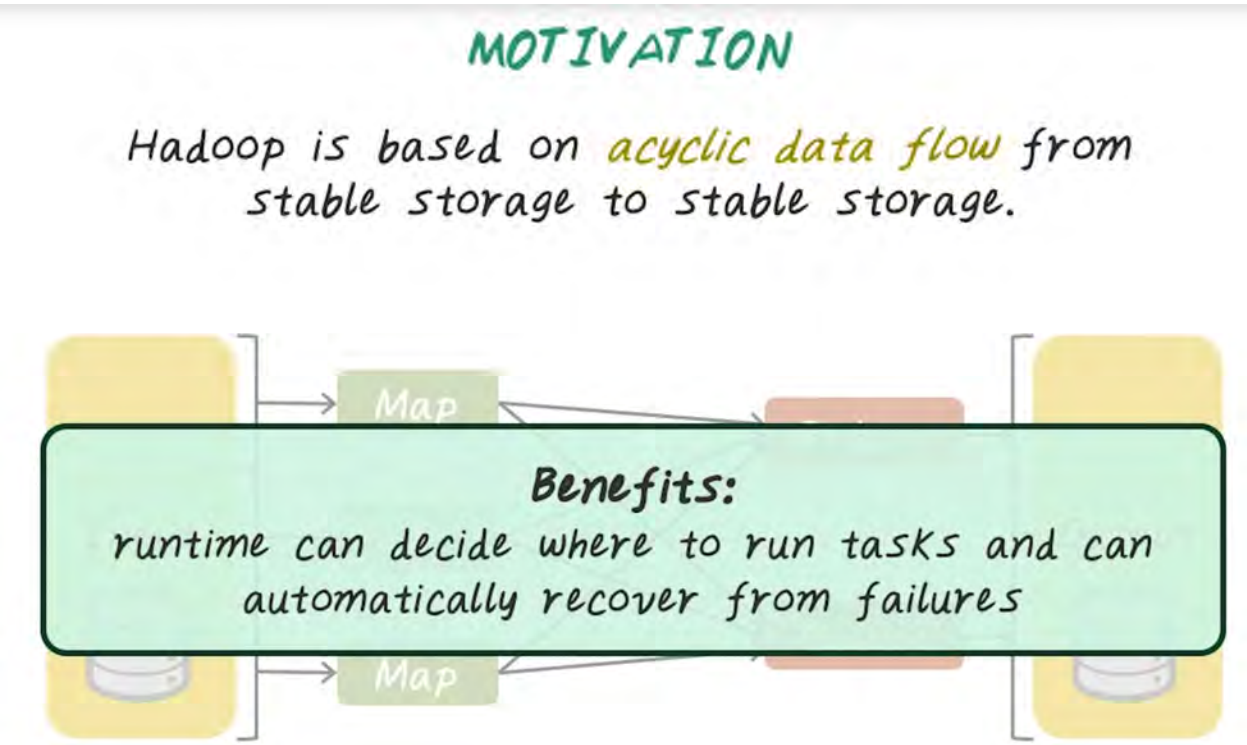
Previously, we have talked about MapReduce, a distributed fault tolerance system, for processing large data set. However, MapReduce, is not efficient for supporting iterative workload as many machine learning algorithm require. Today, we'll introduce Spark, another big data system that provides better 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.

2. 2. Environment

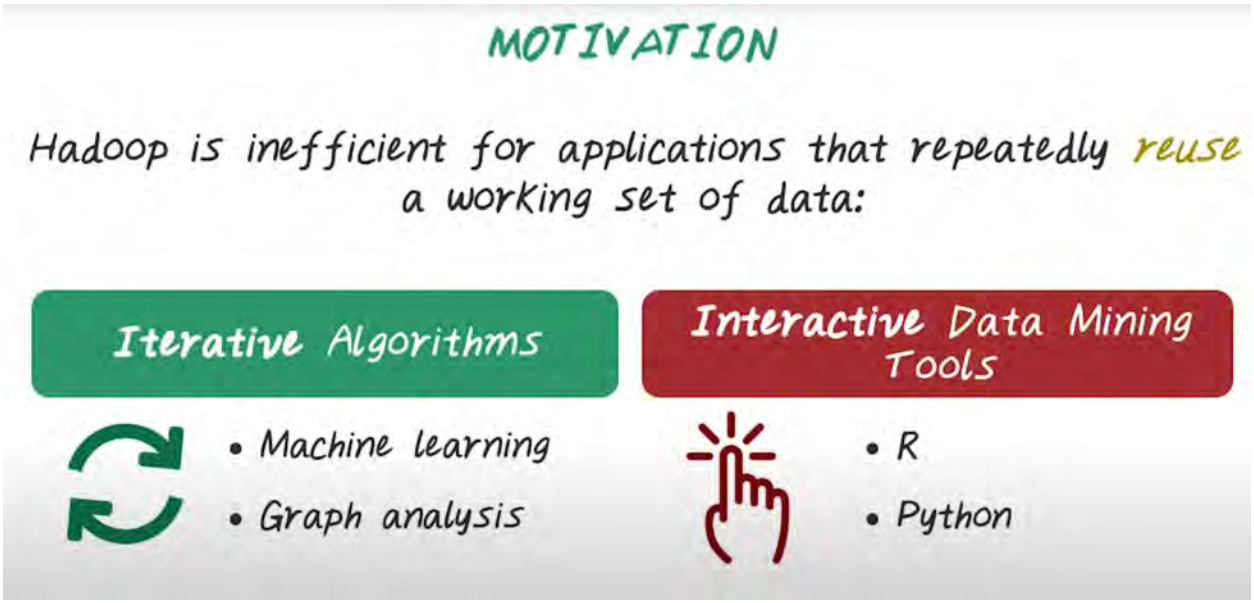


Before we introduce the big data system Spark, let's first remind ourself of the computing environment we're using here. For big data analytics, we usually need to perform all the analytics in a data center look like this. Many racks of servers that are interconnected through Internet. A lot of time 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

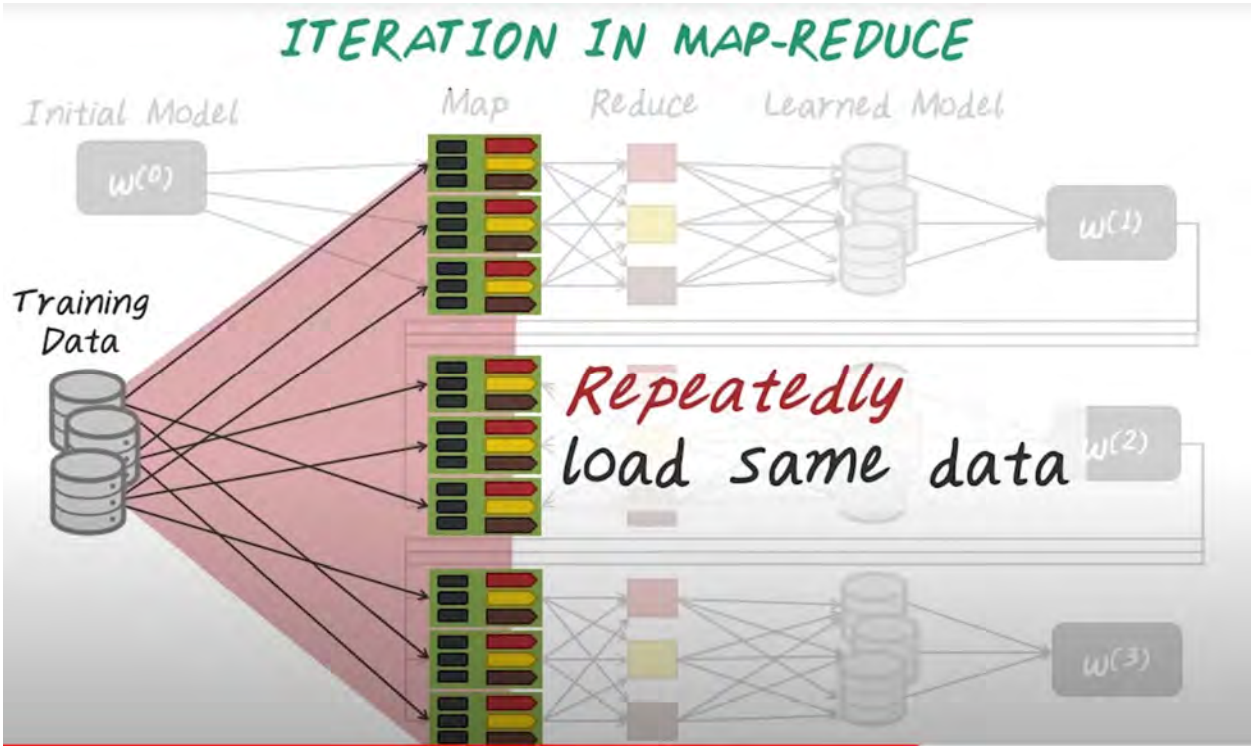
3. 3. Motivation

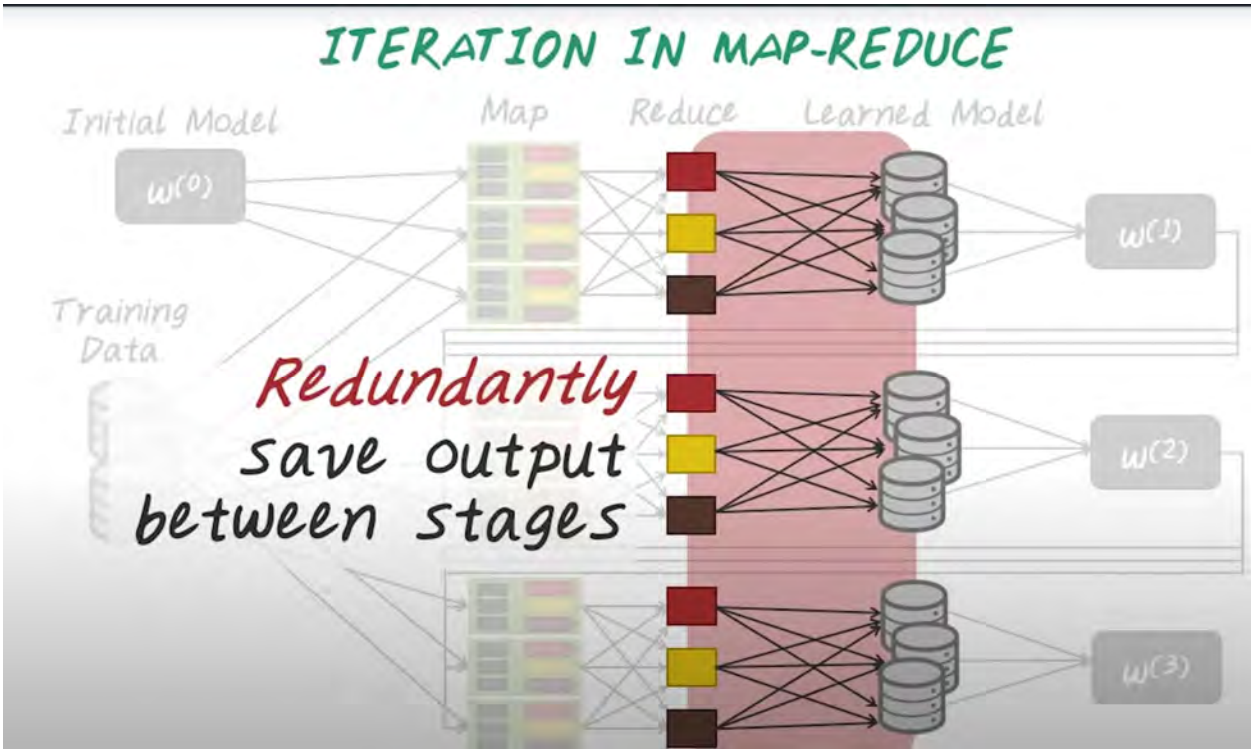


In the previous lecture we have introduced Hadoop and MapReduce. Hadoop is a big data system that operates on acyclic data flow graphs from stable storage to stable storage. So the input and output of Hadoop jobs are from the stable storage system. For example, hard disks in the distributed environment. 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.



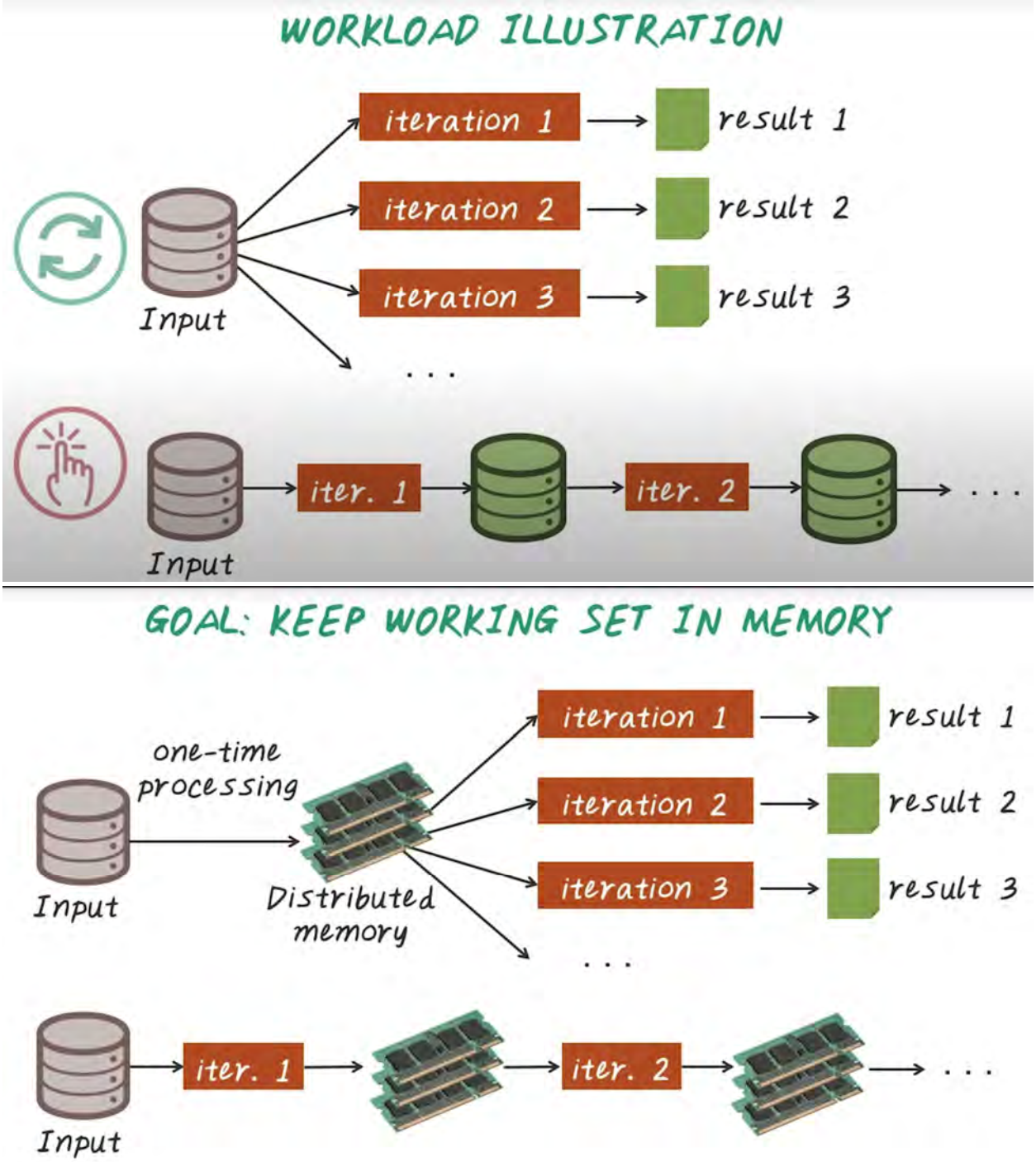
4. [4. Iteration in Map Reduce]





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.

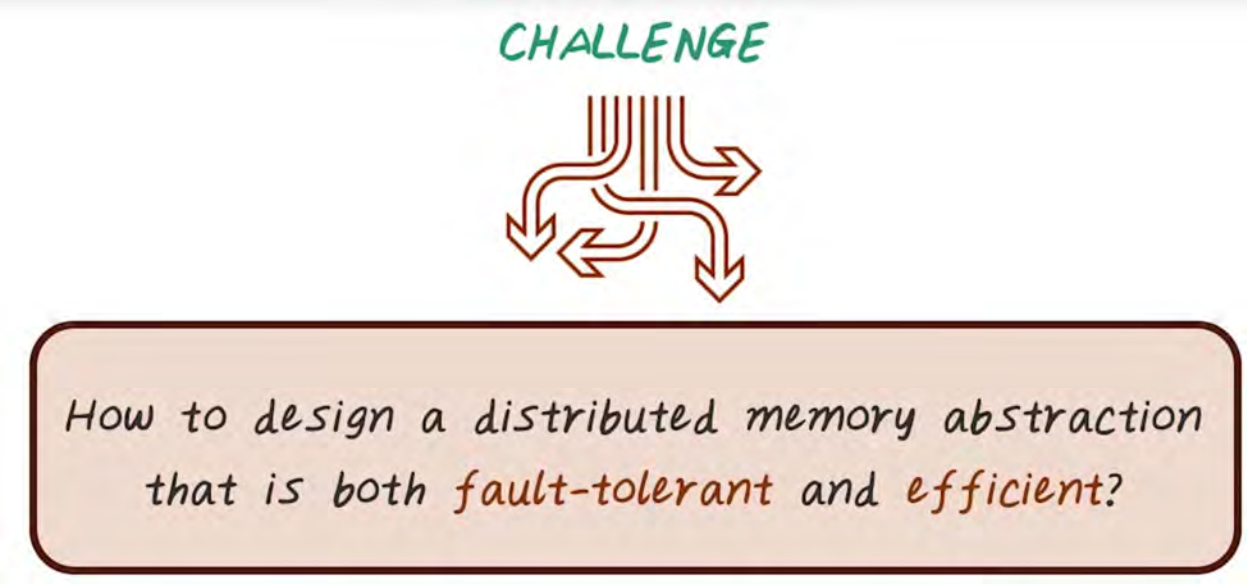
5. 5. Workload Illustration

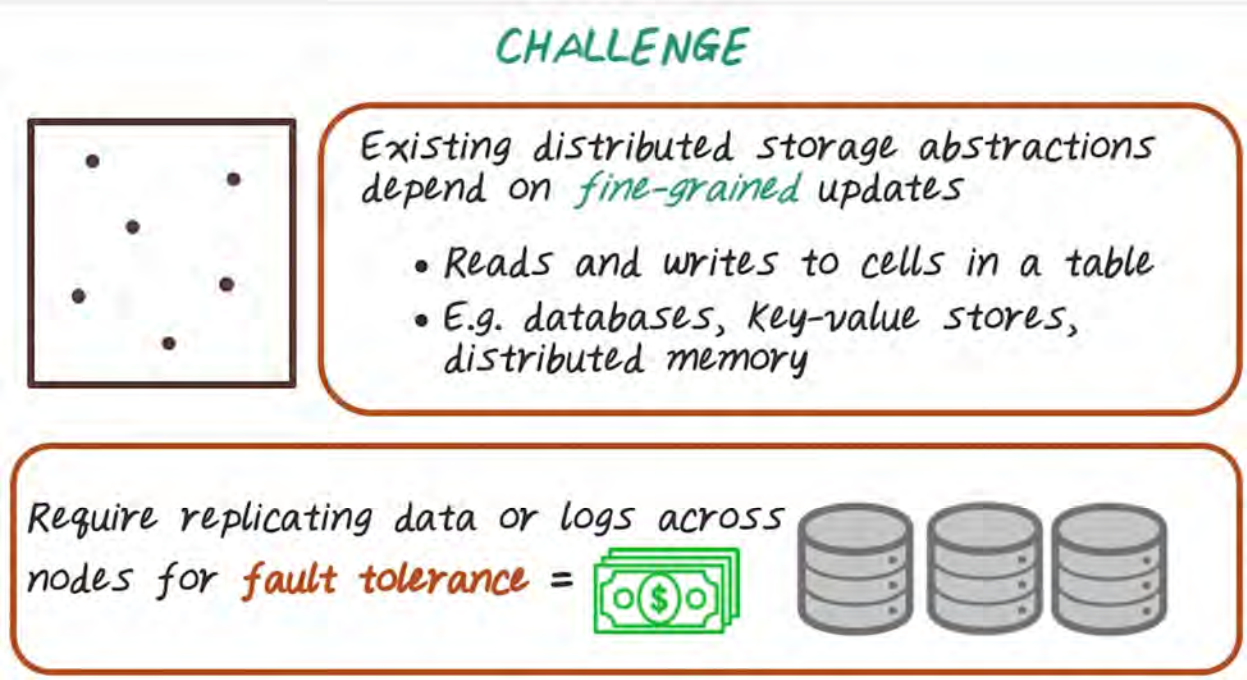


Next, let's illustrate these two types of workload, iterative algorithm and interactive computation. For iterative algorithm, we apply the same computation logic on the same input data set again and again to generate different iteration of result set. And those result are often corresponding to the model 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 a 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.

6. 6. Challenge

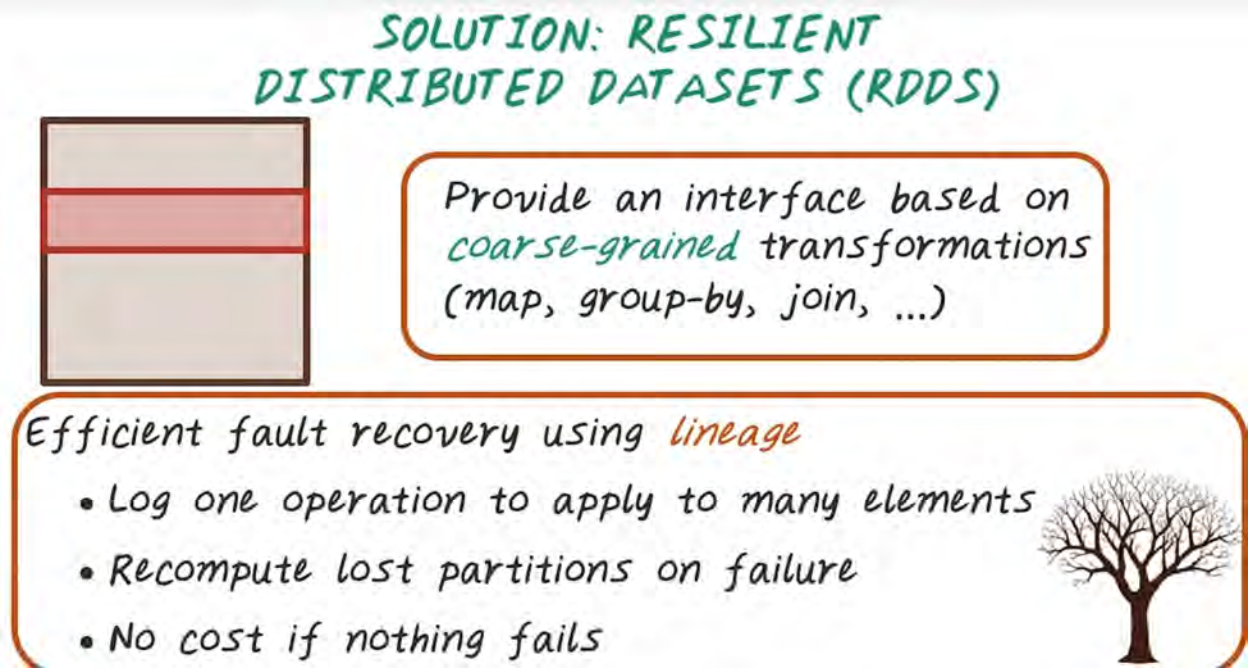




So what's the challenge about keeping everything in memory? The key challenge is how do we design a distributed memory abstraction that is both fault-tolerant and efficient. Hadoop

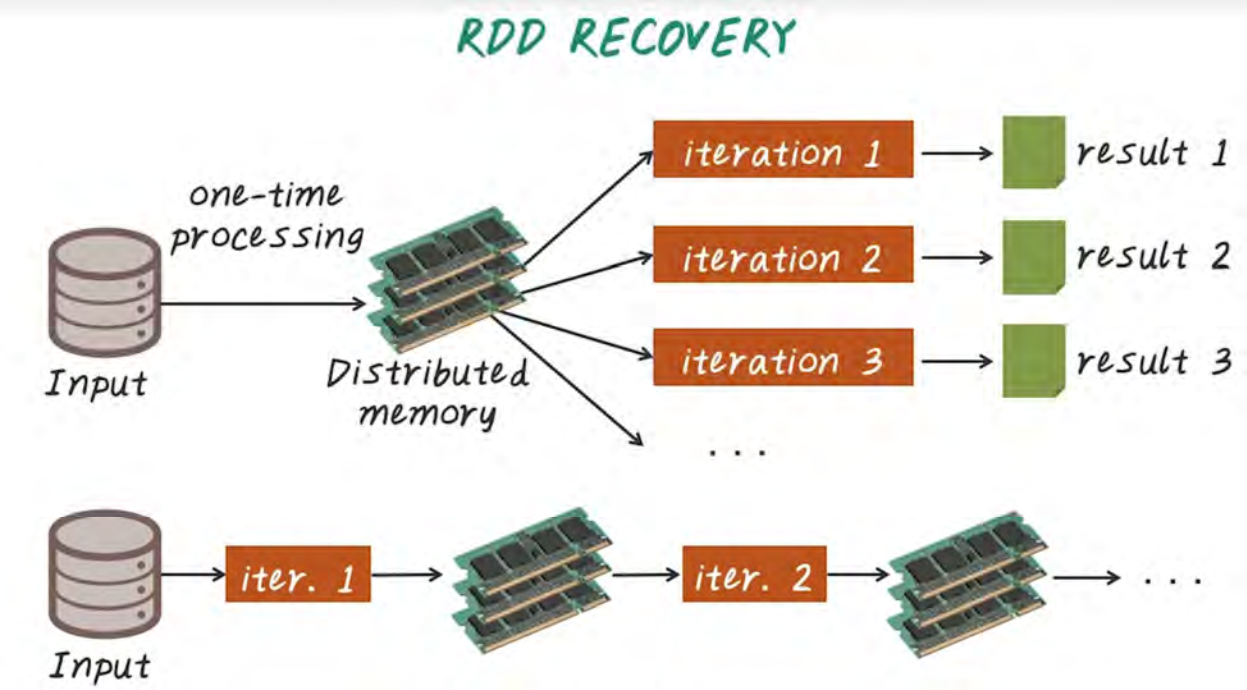
guarantee fault- tolerance by using disk so that they can keep data and intermediate result in a stable storage. But, memory is not a stable storage. It's, by definition, efficient, but it's not fault-tolerant. To guarantee both fault-tolerance and efficiency is the challenge for such a system. Next, 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.

7. 7. Solution RDDs



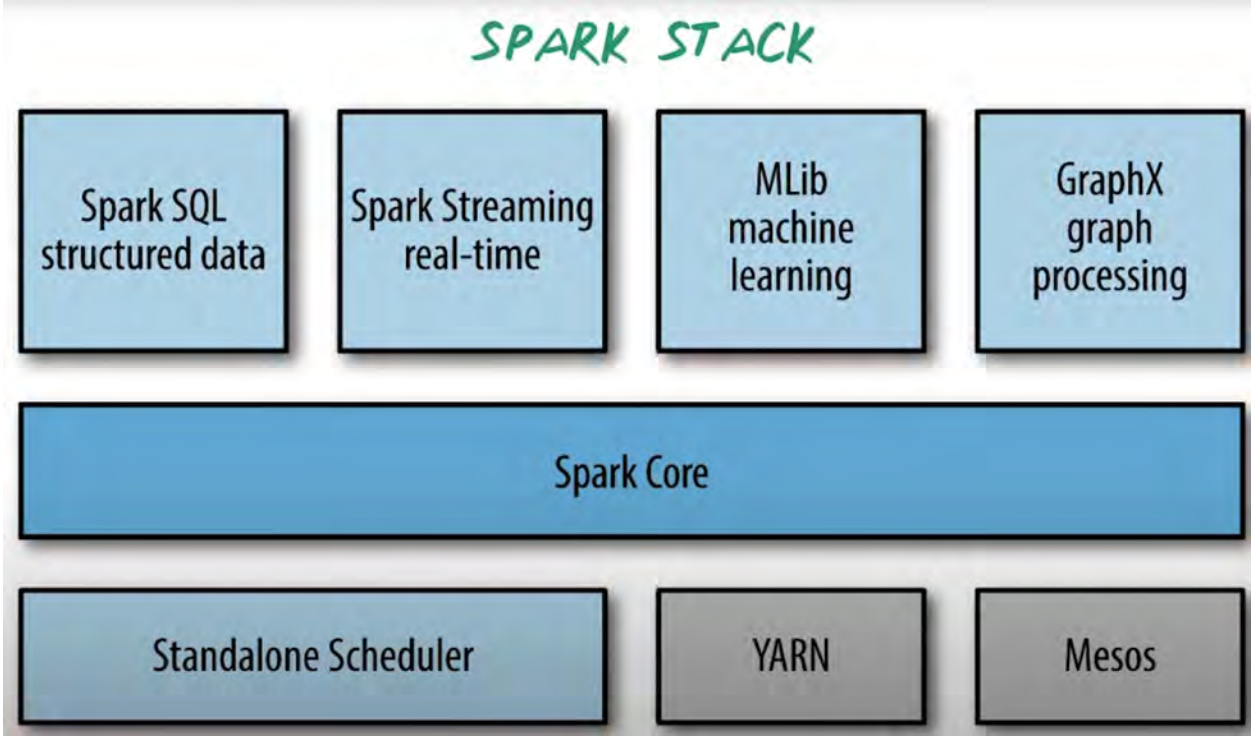
To solve this problem, spark decides to strike a balance between granularity of the computation and the efficiency for enabling fall tolerance. In this case, RDD provide an interface based on coarse-grained transformations of the entire data set. For example, a map function can be performed on every 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.

8. 8. RDD Recovery



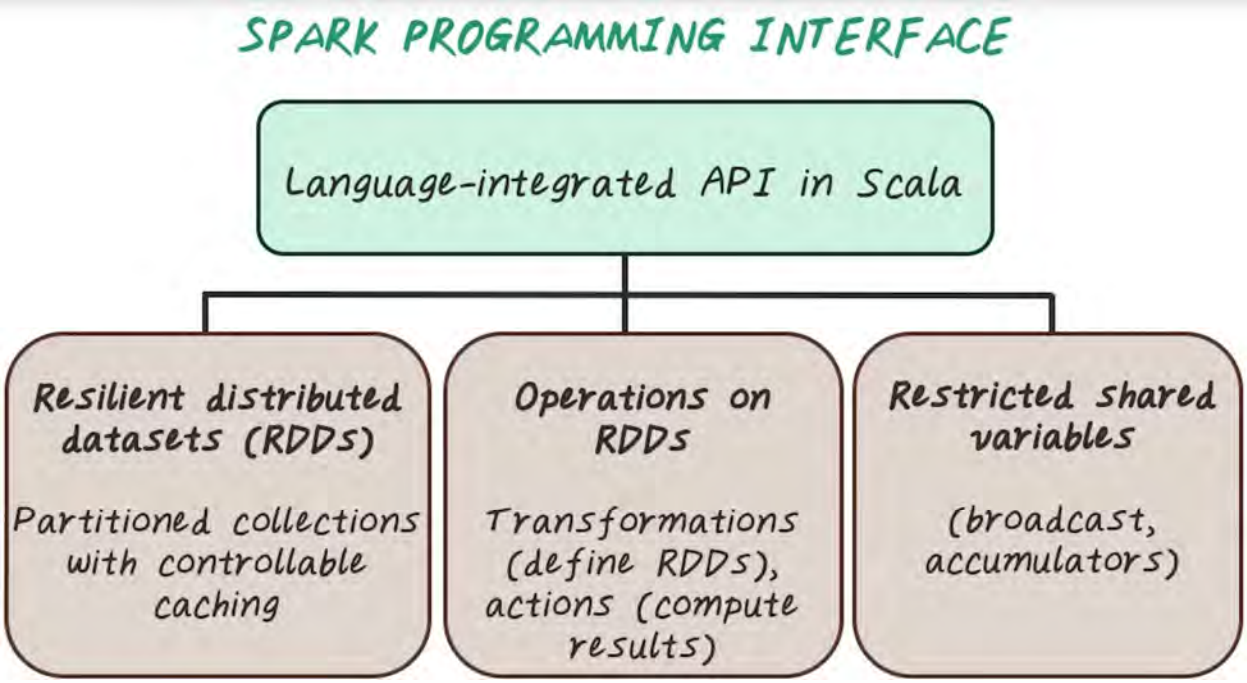
Next, let's illustrate how RDD can recover from failure. Again, we have these two different scenarios. This is for iterative algorithms and this is for interactive computation. It's quite obvious for the interactive computation. For example, the result from the second iteration failed. In this case, 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.

9. 9. Spark Stack



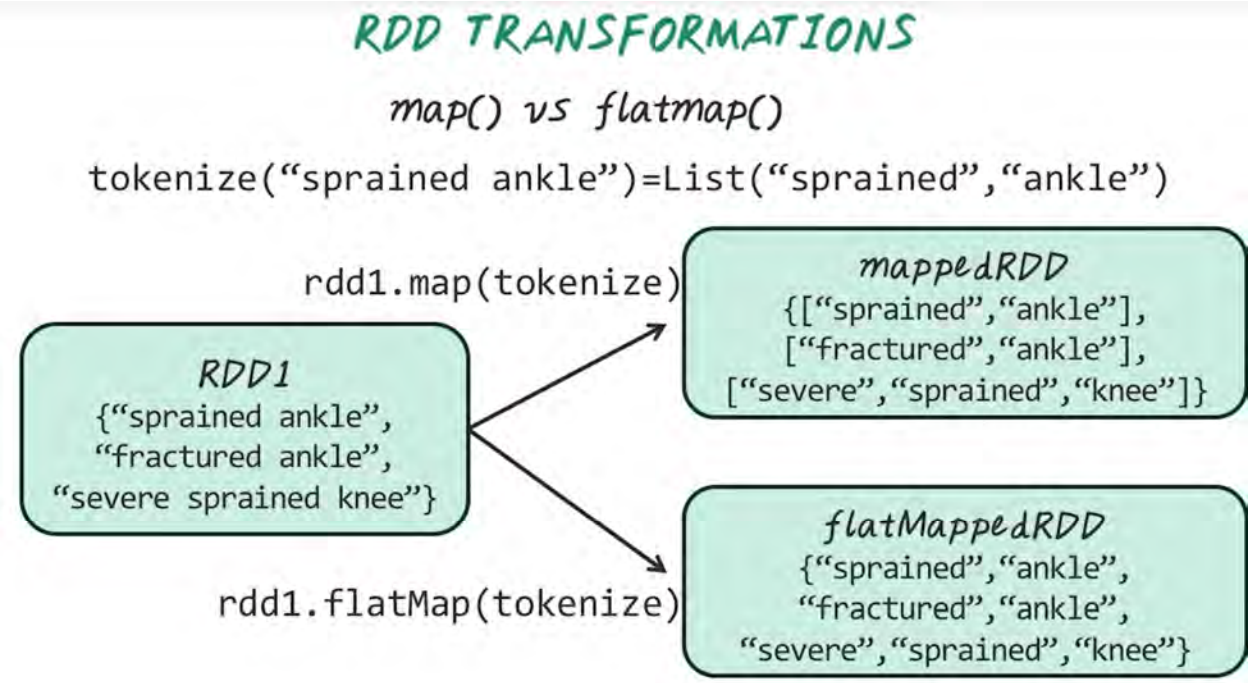
Software stack for Spark already become quite rich. The Spark core contains the basic functionality of Spark including components for scheduling, memory management, fault recovery, and interacting with storage systems and more. And Spark Core is a home API for RDD and this RDD, as we just illustrated, 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 oncologies. 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.

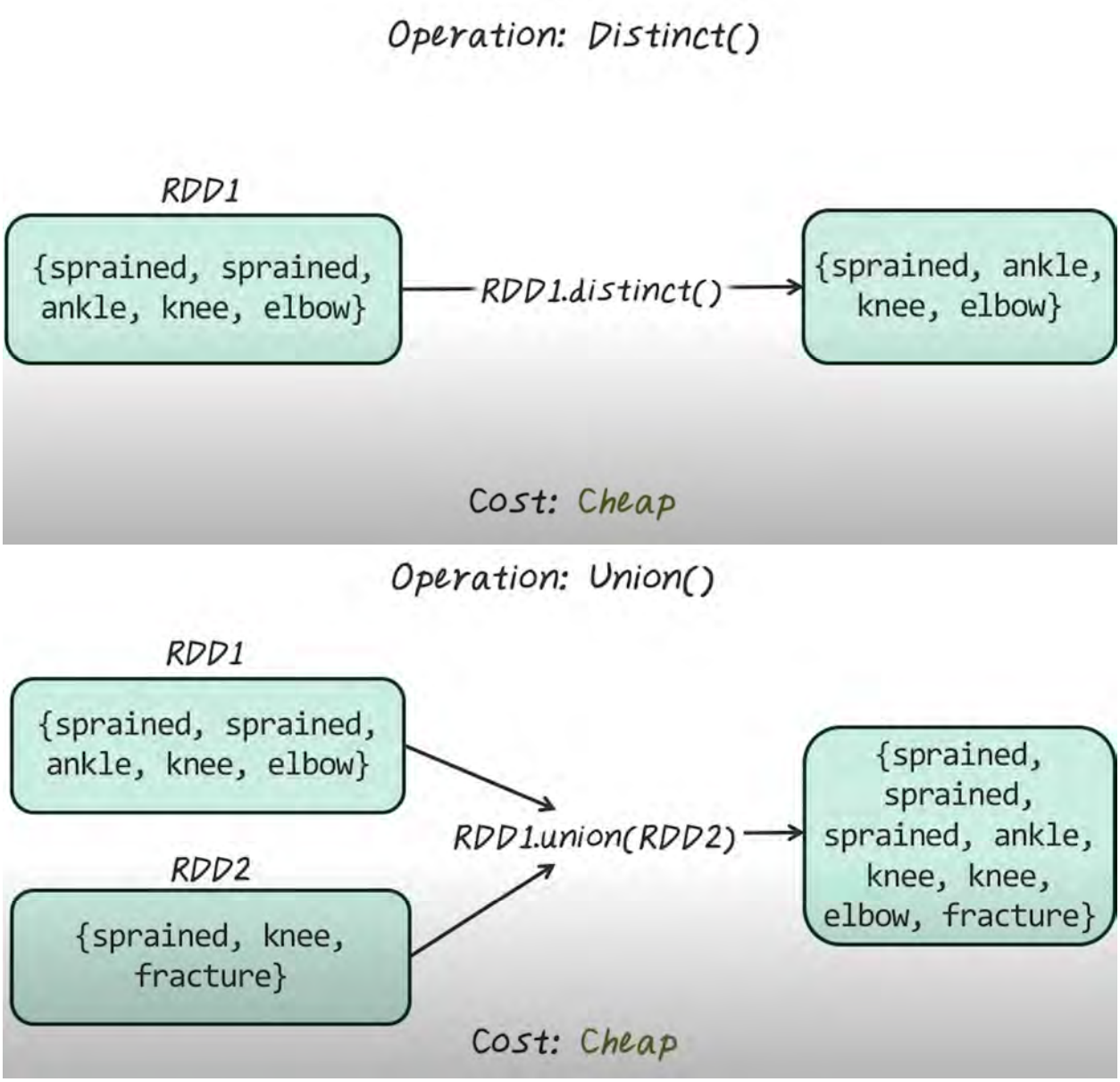
10. 10. Spark Programming Interface

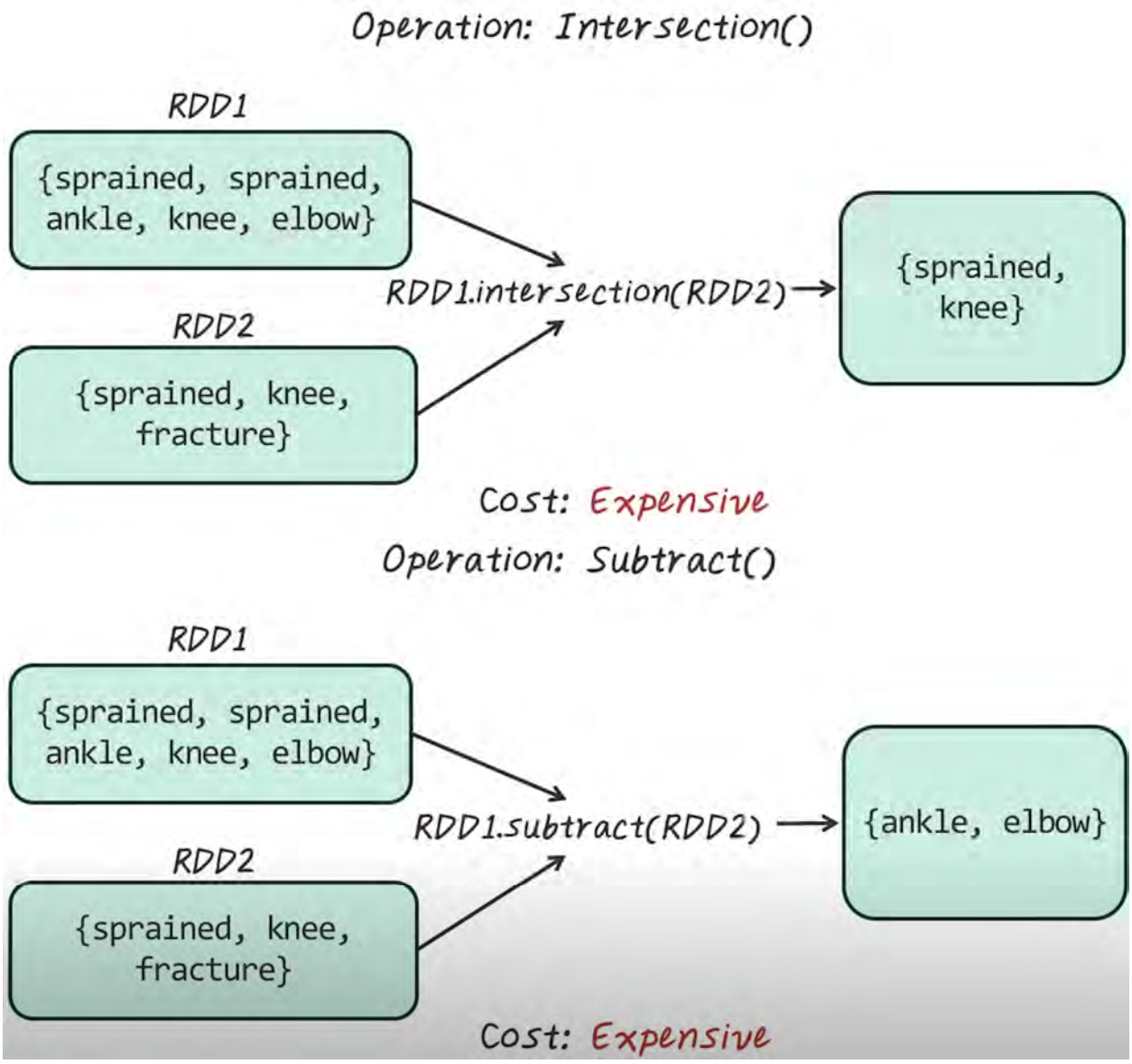


Next, let's talk about programming interface for Spark. The core interface is written in Scala. It has several other different languages being supported such as Python and Java. There are three different types of API. There's one set of API is about creating and manipulating RDD. Then we have these different 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.

11. [11. RDD Transformations]




Operation: Distinct()
RDD1
{sprained, sprained, ankle, knee, elbow}
RDD1.distinct()
{sprained, ankle, knee, elbow}
Cost: Cheap
Operation: Union()
RDD1
{sprained, sprained, ankle, knee, elbow}
RDD2
{sprained, knee, fracture}
RDD1.union(RDD2)
{sprained, sprained, sprained, ankle, knee, knee, elbow, fracture}
Cost: Cheap



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 [INAUDIBLE] 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.

12. 12. Basic RDD Transformations

Table 3-2. Basic RDD transformations on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`map()`	Apply a function to each element in the RDD and return an RDD of the result.	`rdd.map(x => x + 1)`	`{2, 3, 4, 4}`
`flatMap()`	Apply a function to each element in the RDD and return an RDD of the contents of the iterators returned. Often used to extract words.	`rdd.flatMap(x => x.to(3))`	`{1, 2, 3, 2, 3, 3, 3}`
`filter()`	Return an RDD consisting of only elements that pass the condition passed to `filter()`.	`rdd.filter(x => x != 1)`	`{2, 3, 3}`
`distinct()`	Remove duplicates.	`rdd.distinct()`	`{1, 2, 3}`
`sample(withReplacement, fraction, [seed])`	Sample an RDD, with or without replacement.	`rdd.sample(false, 0.5)`	Nondeterministic

Table 3-3. Two-RDD transformations on RDDs containing {1, 2, 3} and {3, 4, 5}

Function name	Purpose	Example	Result
`union()`	Produce an RDD containing elements from both RDDs.	`rdd.union(other)`	`{1, 2, 3, 3, 4, 5}`
`intersection()`	RDD containing only elements found in both RDDs.	`rdd.intersection(other)`	`{3}`
`subtract()`	Remove the contents of one RDD (e.g., remove training data).	`rdd.subtract(other)`	`{1, 2}`
`cartesian()`	Cartesian product with the other RDD.	`rdd.cartesian(other)`	`{(1, 3), (1, 4), … (3,5)}`

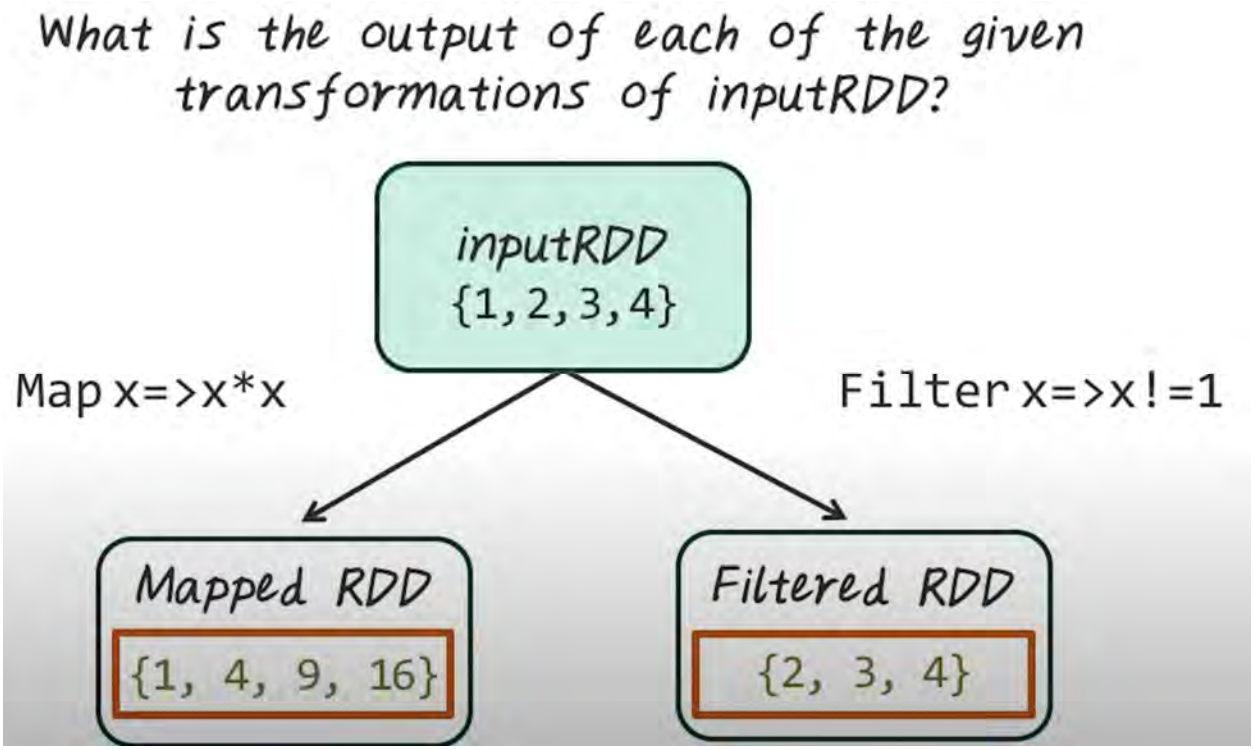
Table 3-4. Basic actions on an RDD containing {1, 2, 3, 3}

Function name	Purpose	Example	Result
`collect()`	Return all elements from the RDD.	`rdd.collect()`	`{1, 2, 3, 3}`
`count()`	Number of elements in the RDD.	`rdd.count()`	`4`
`countByValue()`	Number of times each element occurs in the RDD.	`rdd.countByValue()`	`{(1, 1), (2, 1), (3, 2)}`

Function name	Purpose	Example	Result
`take(num)`	Return num elements from the RDD.	`rdd.take(2)`	`{1, 2}`
`top(num)`	Return the top num elements the RDD.	`rdd.top(2)`	`{3, 3}`
`takeOrdered(num)(ordering)`	Return num elements based on provided ordering.	`rdd.takeOrdered(2)(myOrdering)`	`{3, 3}`
`takeSample(withReplacement, num, [seed])`	Return num elements at random.	`rdd.takeSample(false, 1)`	Nondeterministic
`reduce(func)`	Combine the elements of the RDD together in parallel (e.g., sum).	`rdd.reduce((x, y) => x + y)`	9
`fold(zero)(func)`	Same as `reduce()` but with the provided zero value.	`rdd.fold(0)((x, y) => x + y)`	9
`aggregate(zeroValue)(seqOp, combOp)`	Similar to `reduce()` but used to return a different type.	`rdd.aggregate((0, 0))((x, y) => (x._1 + y, x._2 + 1), (x, y) => (x._1 + y._1, x._2 + y._2))`	`(9, 4)`
`foreach(func)`	Apply the provided function to each element of the RDD.	`rdd.foreach(func)`	Nothing

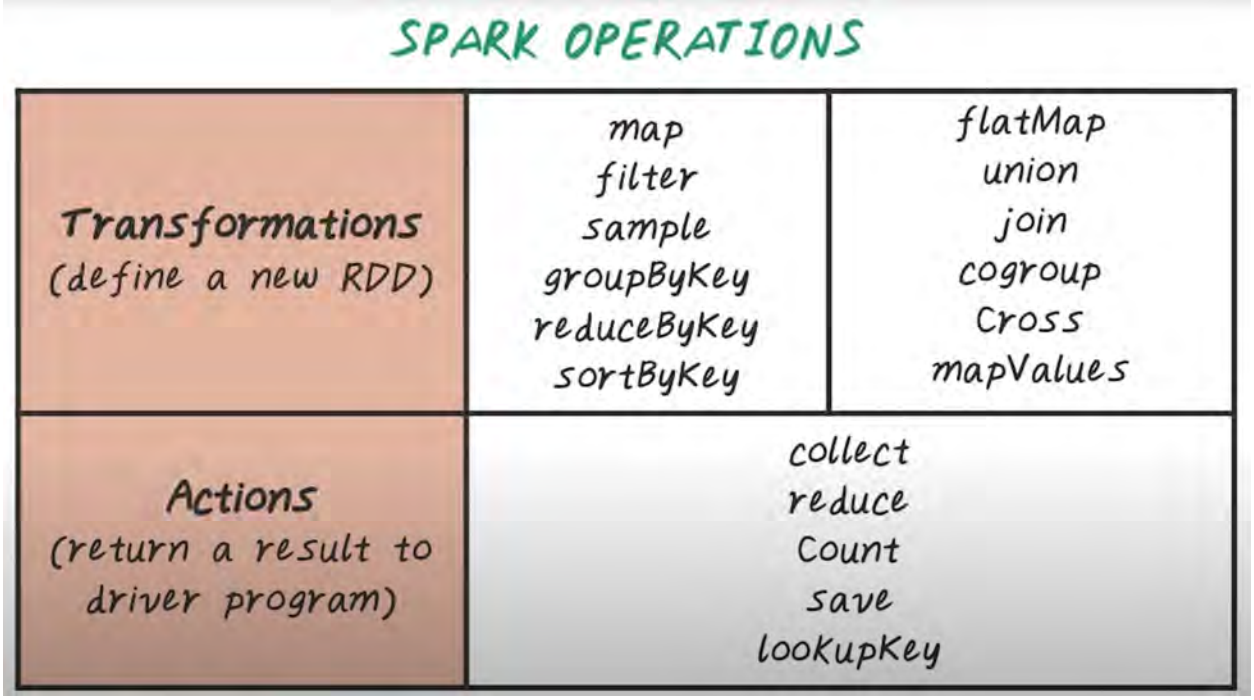
13. [13. RDD Transformations Quiz]

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.



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.

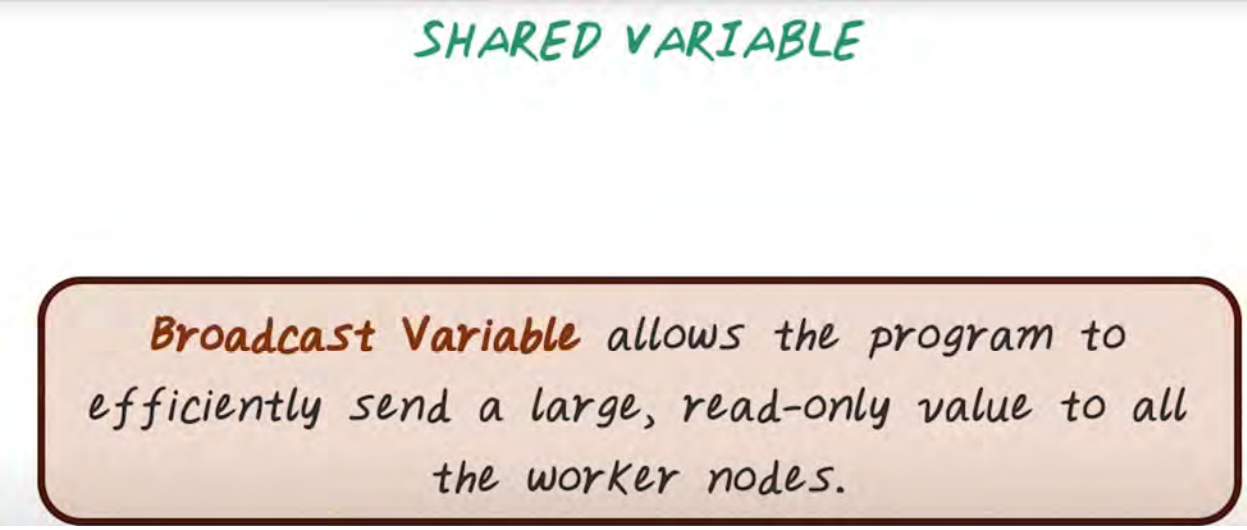
14. [14. Spark Operations]



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,

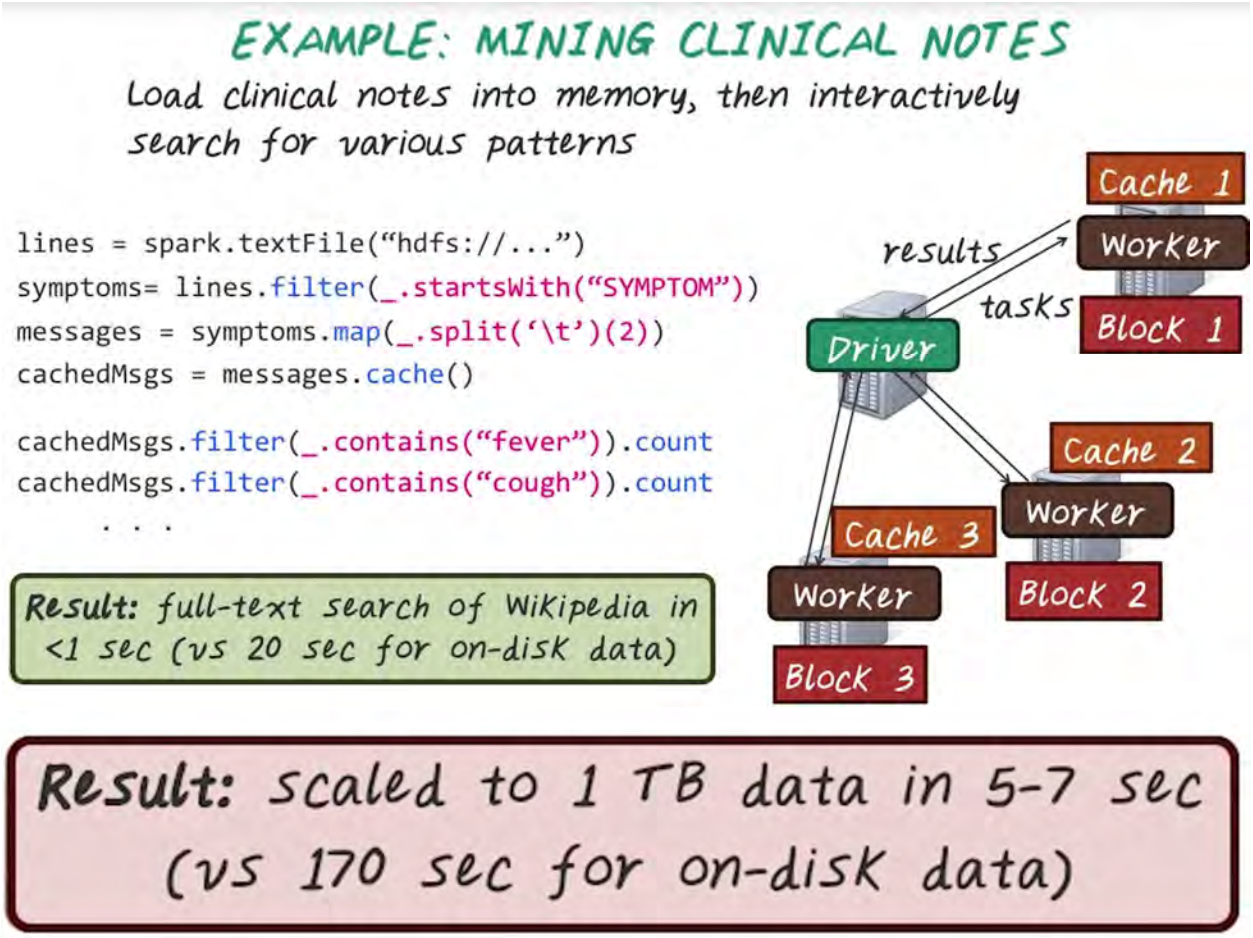
mapValues. So it's a very rich set up of transformations, way beyond what MapReduce provide us in Hadoop. Same for Actions. Action is like the reduce phase in MapReduce. So it can collect the values from RDD. We can perform a reduce function, we can count how many records, we can save it somewhere else, 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 example will be in the instructor notes.

15. 15. Shared Variable

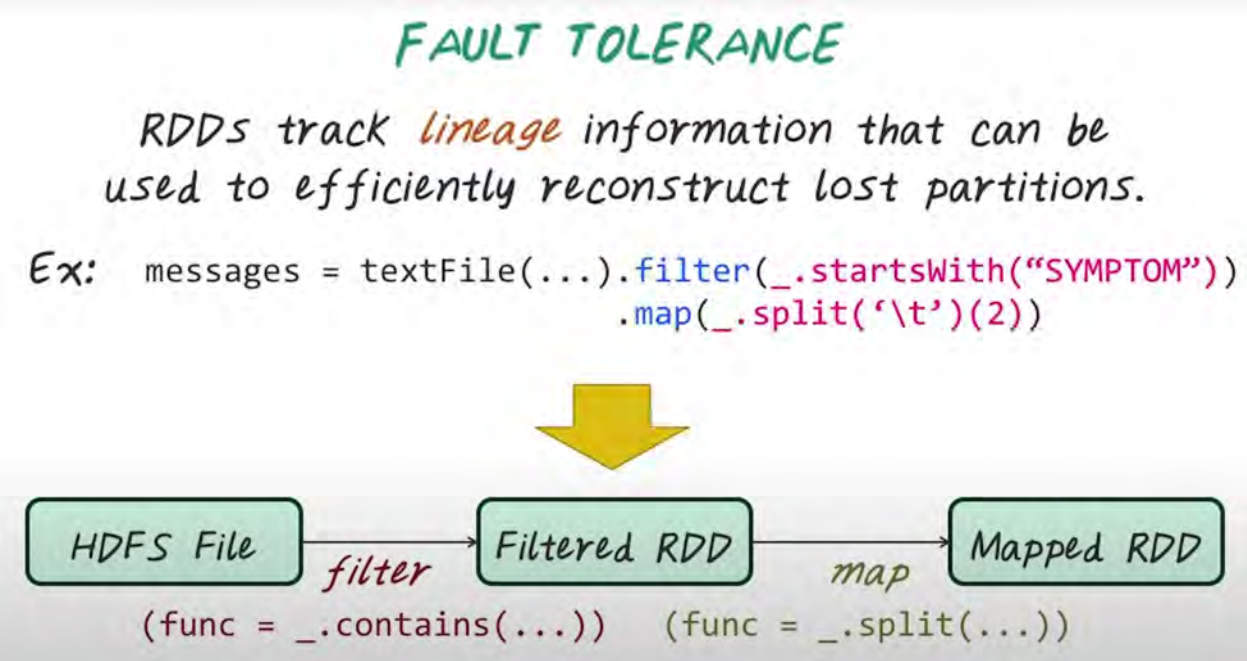


Another important concept in Spark is this shared variable. The way to create a shared variable is to use this broadcast variable that allows us to efficiently send a large, read-only values to all the worker nodes. Next, let's illustrate a Spark job using this example. Say we have a large volume of 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.

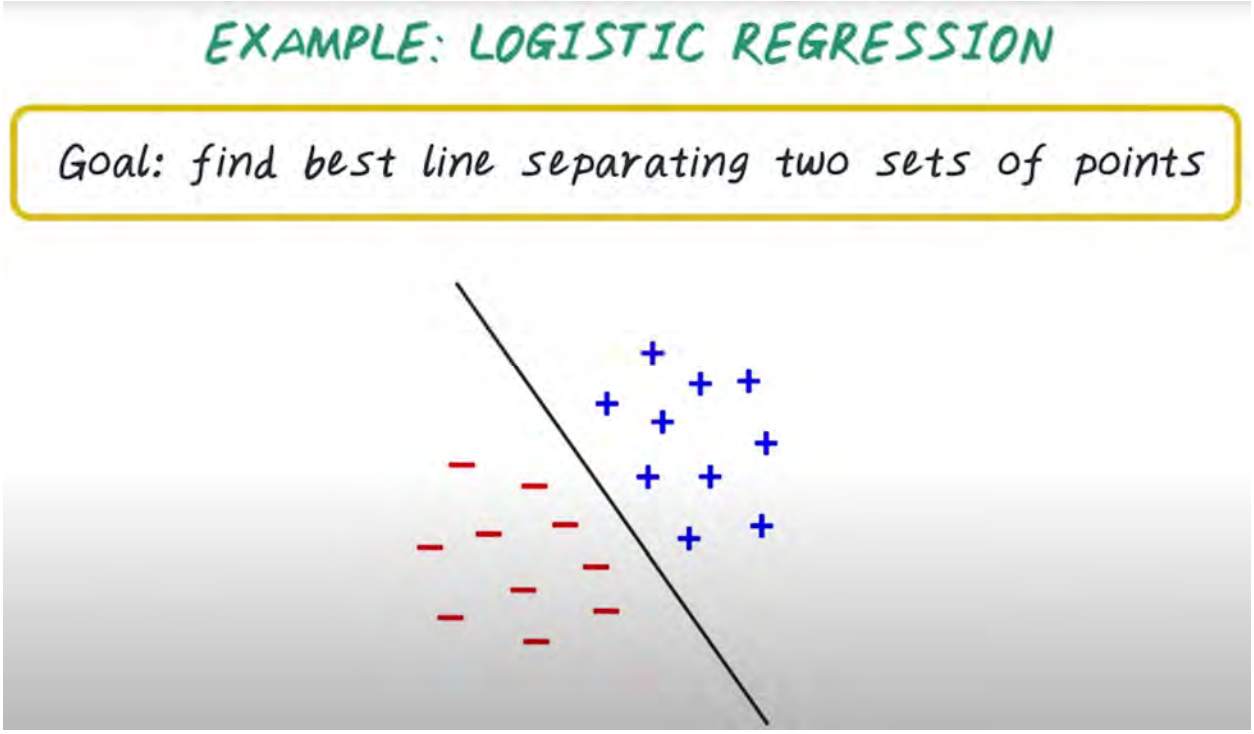


16. [16. Fault Tolerance]



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.

17. 17. Example Logistic Regression

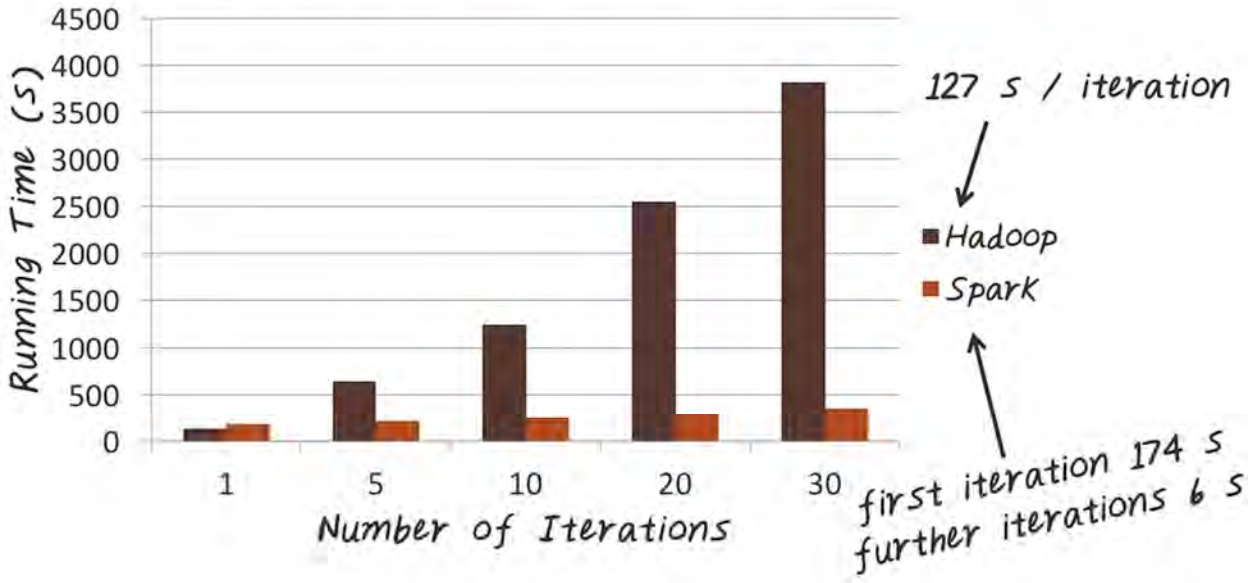


With Spark, now we can efficiently support iterative algorithms such as machine learning algorithm like logistic regression. So giving logistic regression as a classification algorithm, trying to find the best line to separate these two sets of points. And this is the target we want to learn. We start 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 an 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.

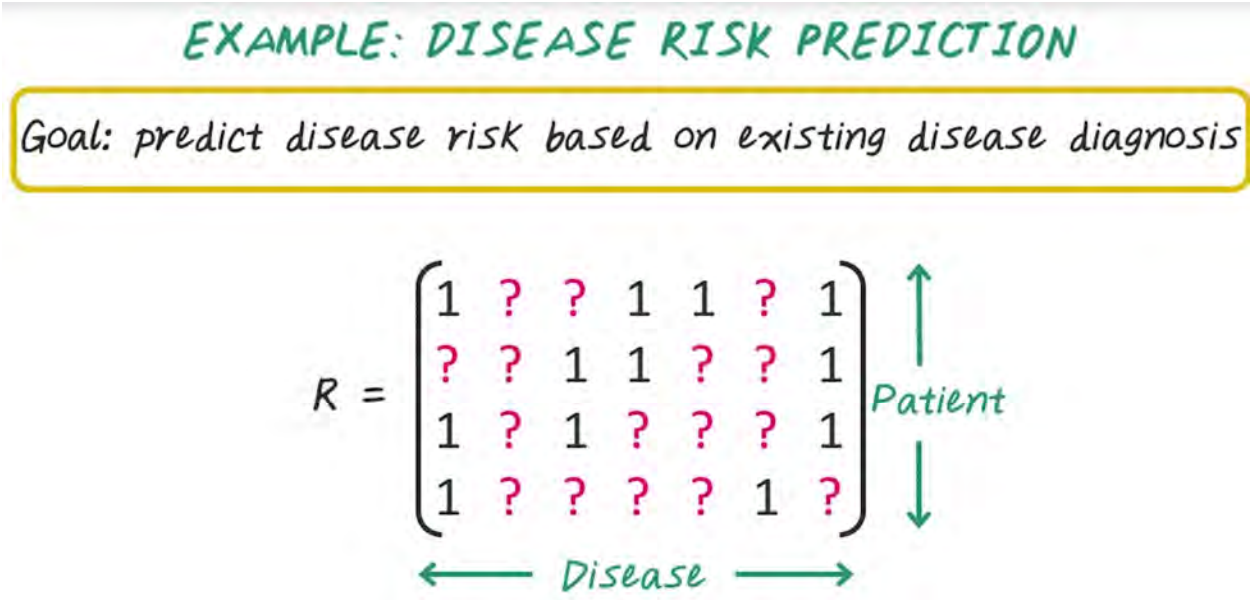
EXAMPLE: LOGISTIC REGRESSION

```
val data = spark.textFile(...).map(readPoint).cache()

var w = Vector.random(D)

for (i <- 1 to ITERATIONS) {
  val gradient = data.map(p =>
    (1 / (1 + exp(-p.y*(w dot p.x))) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= gradient
}

println("Final w: " + w)
```

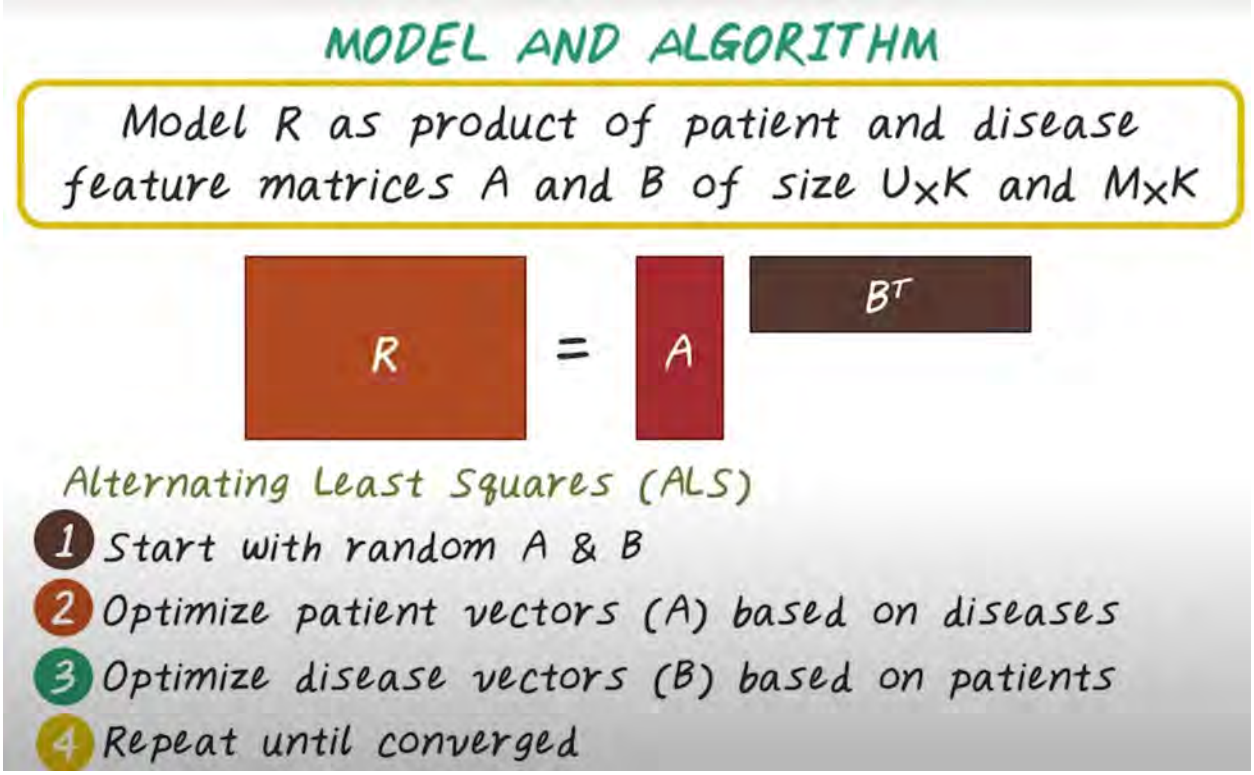
LOGISTIC REGRESSION PERFORMANCE



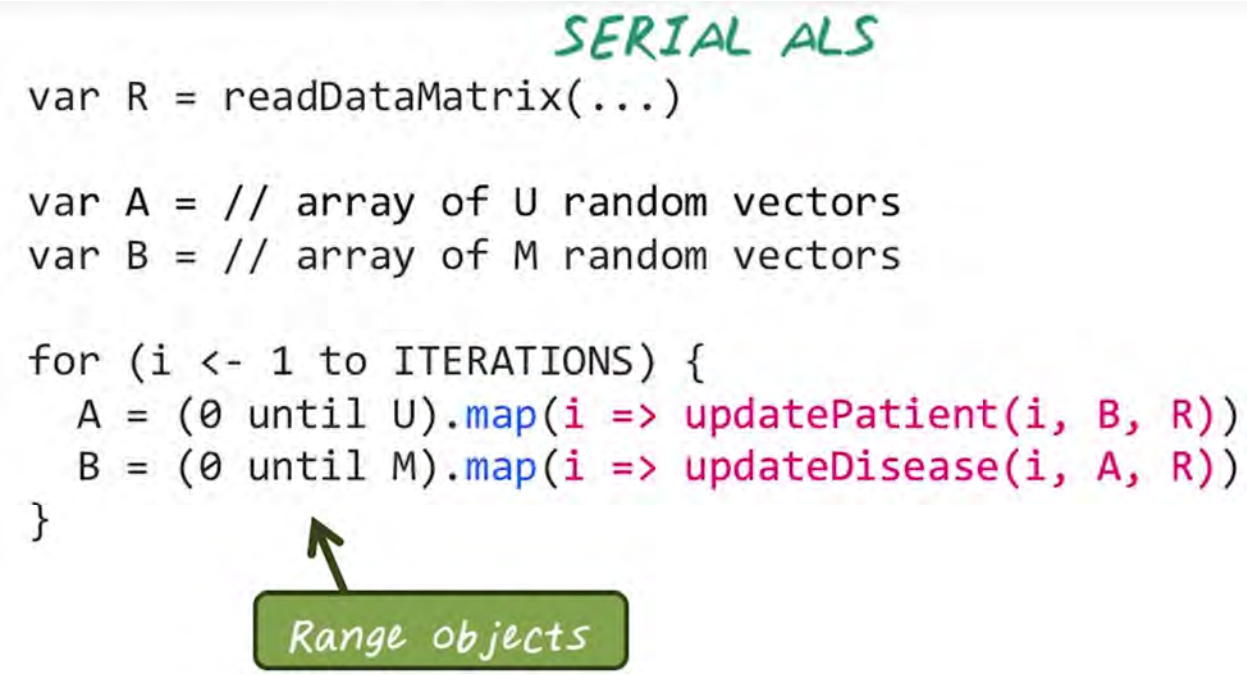
18. 18. Example Disease Risk Prediction



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.



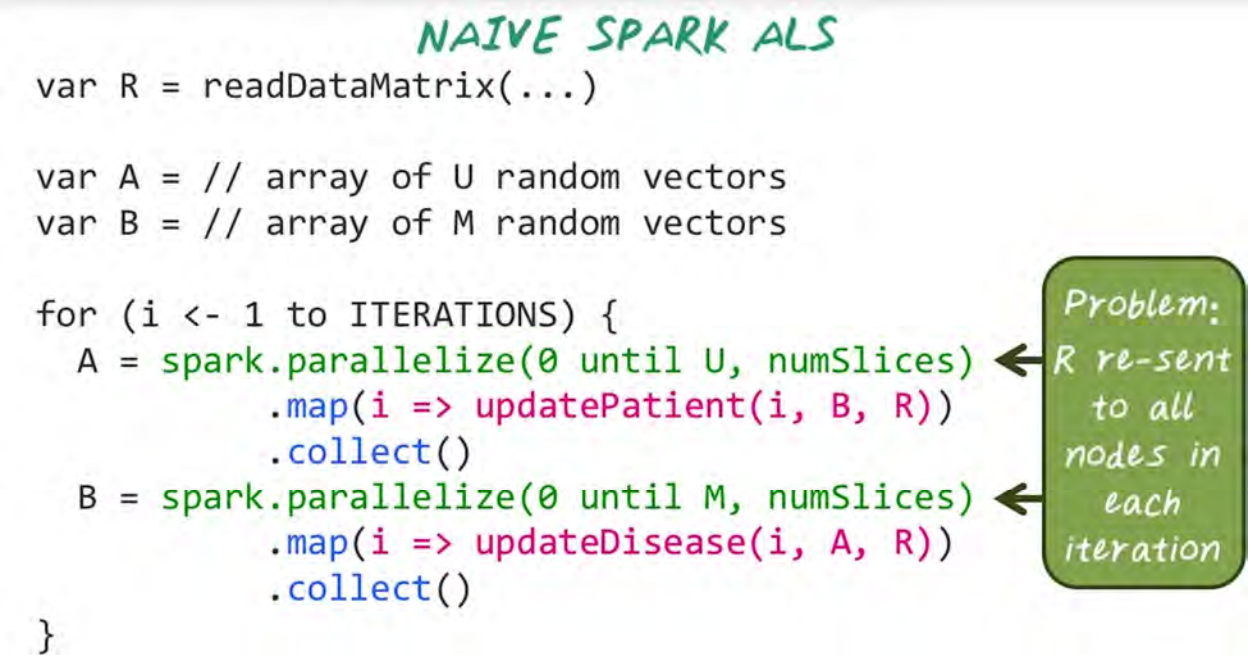
19. [19. Serial ALS]

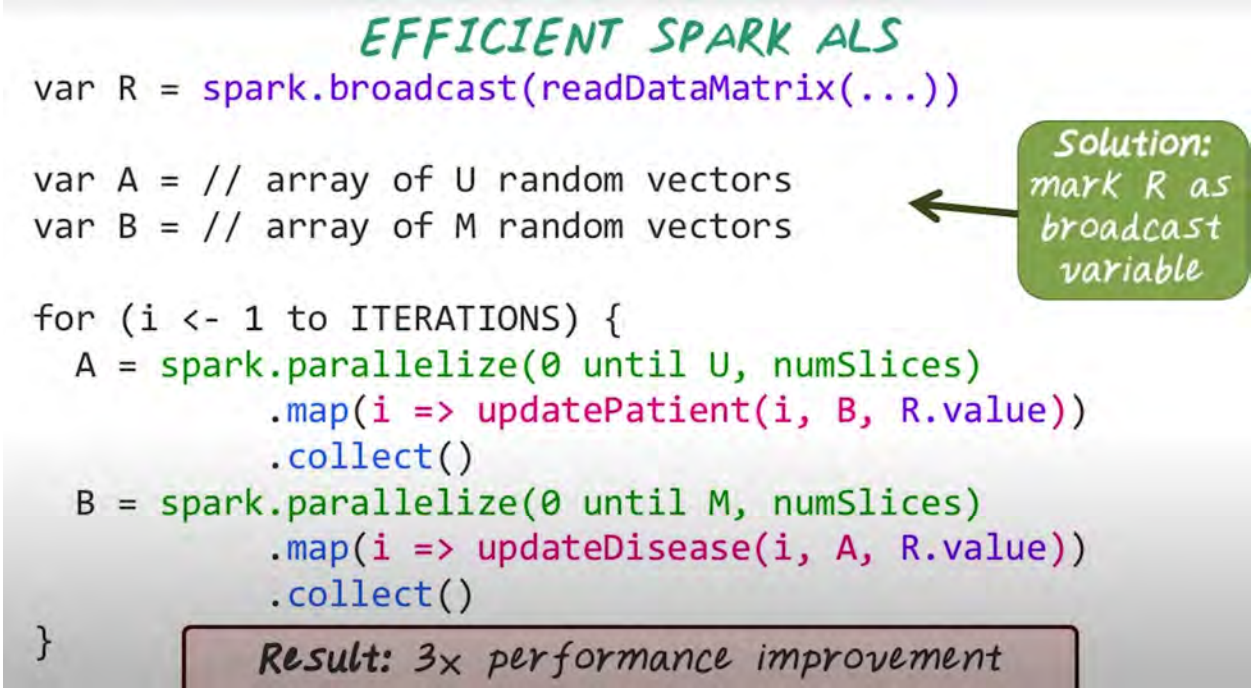


Next, let's talk about how do we do this using Spark. So, we can right this simple 0 implementation of alternating lee square. First, we can read the data matrix R. Then, we can initialize A and B randomly. After that, we can start this iterative operation. Here we want to update A and B alternatively. 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.

20. [20. Naive Spark ALS]





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.

Medical Ontology

1. 1. Introduction to Medical Ontology

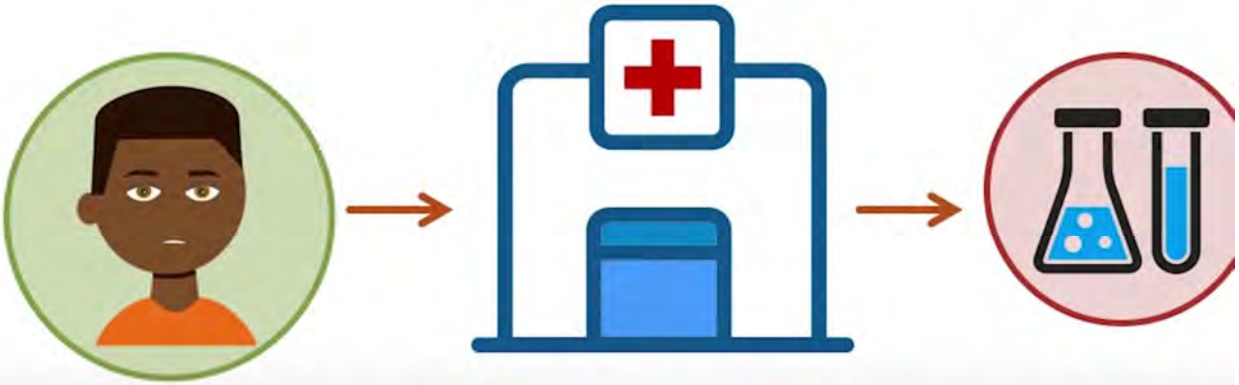
One of the things that make healthcare a unique domain for big data analytics is the existence of structured medical knowledge, which are often represented as ontologies or knowledge graphs. For historical reason, healthcare and medicine domain have already developed many ontologies for organizing diseases, medical procedures, medications, lab tests, and more. These oncologies give us great resources to understand health 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.

2. 2. Health Data Standards

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.

HEALTH DATA STANDARDS



LOINC

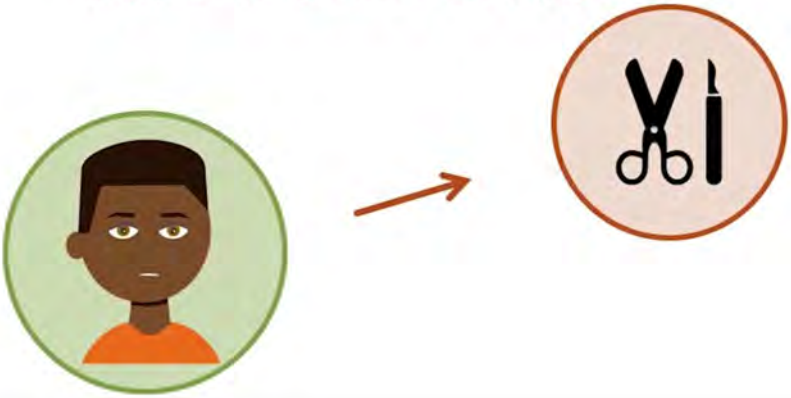
Logical Observation Identifiers Names and Codes

HEALTH DATA STANDARDS

ICD

International Classification of Disease

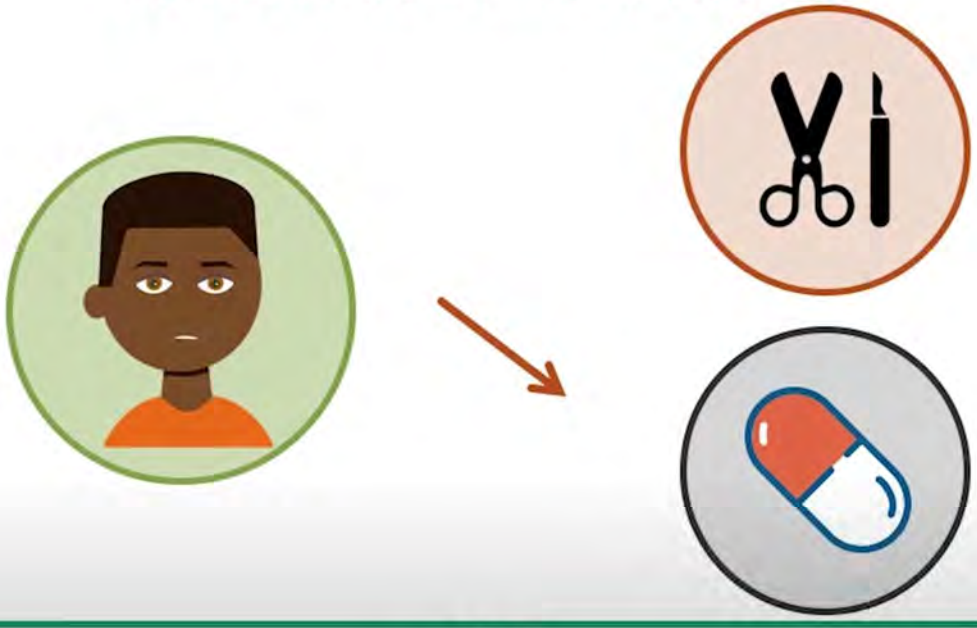
HEALTH DATA STANDARDS



CPT

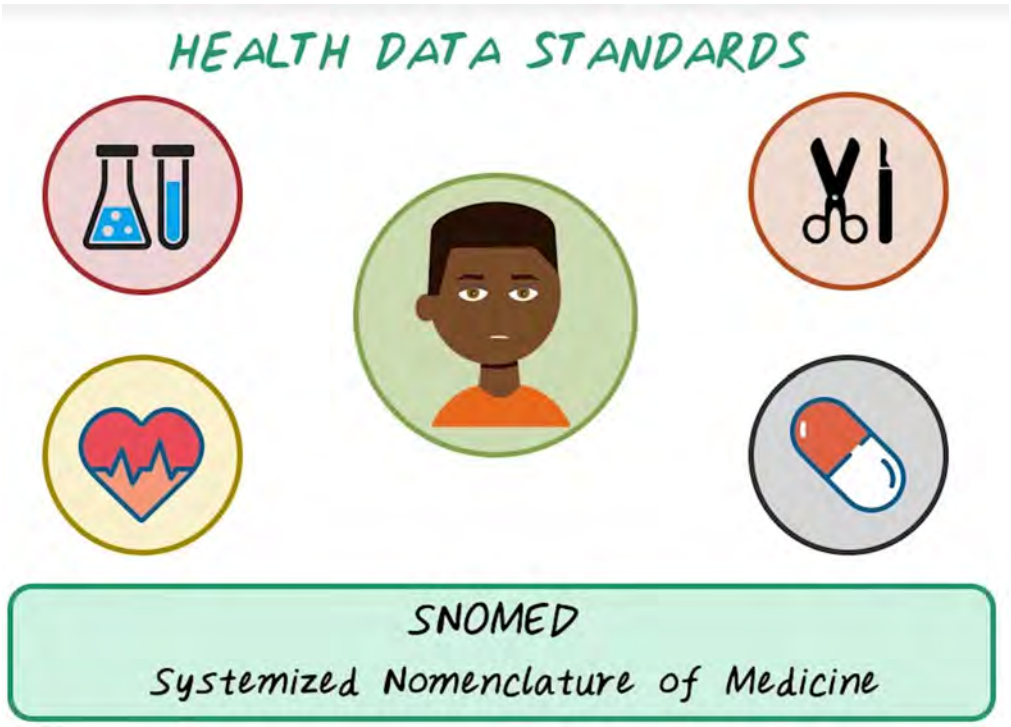
Current Procedural Terminology

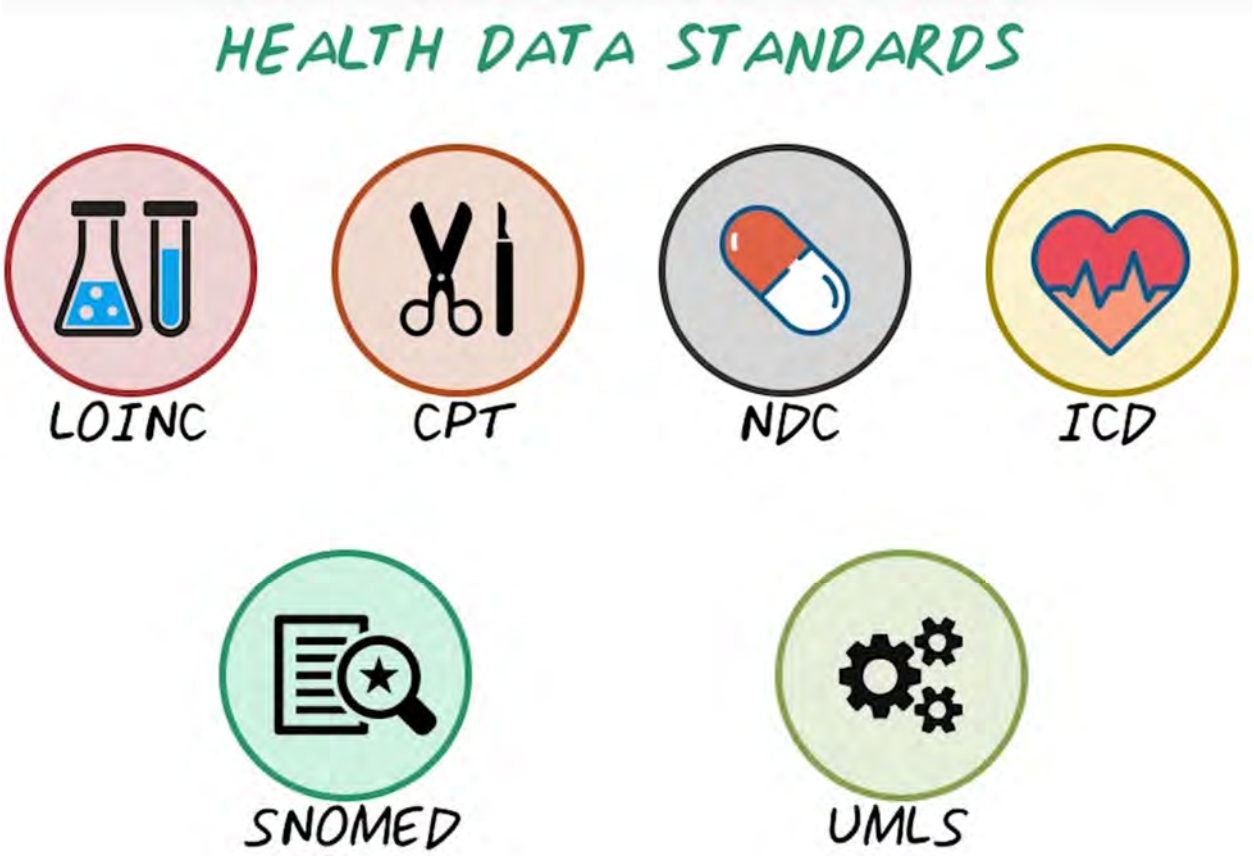
HEALTH DATA STANDARDS



NDC

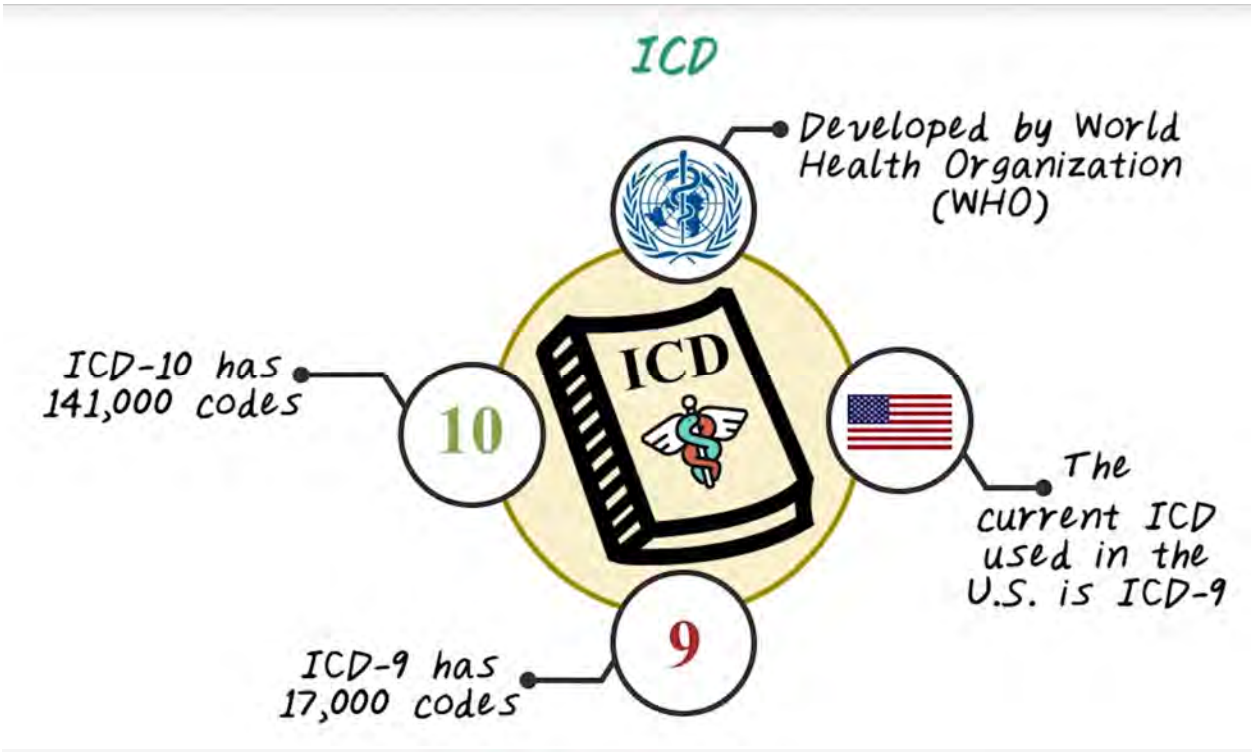
National Drug Code


HEALTH DATA STANDARDS
SNOMED
Systemized Nomenclature of Medicine

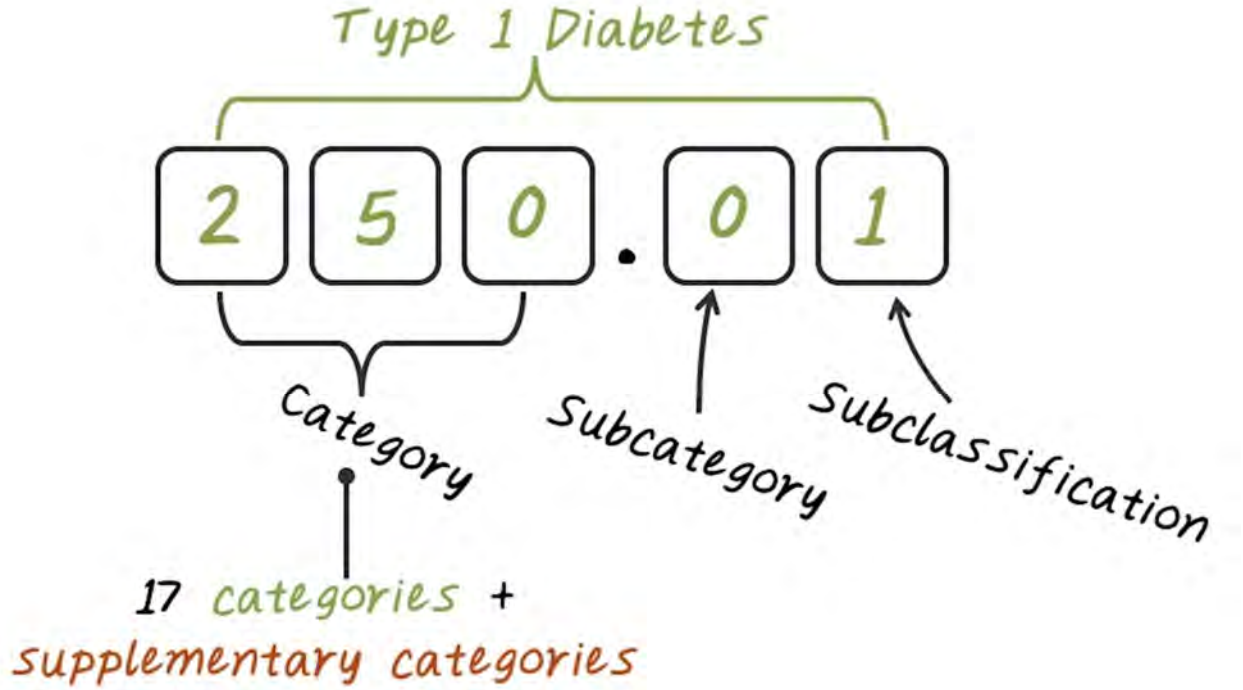

HEALTH DATA STANDARDS
LOINC
CPT
NDC
ICD
SNOMED
UMLS

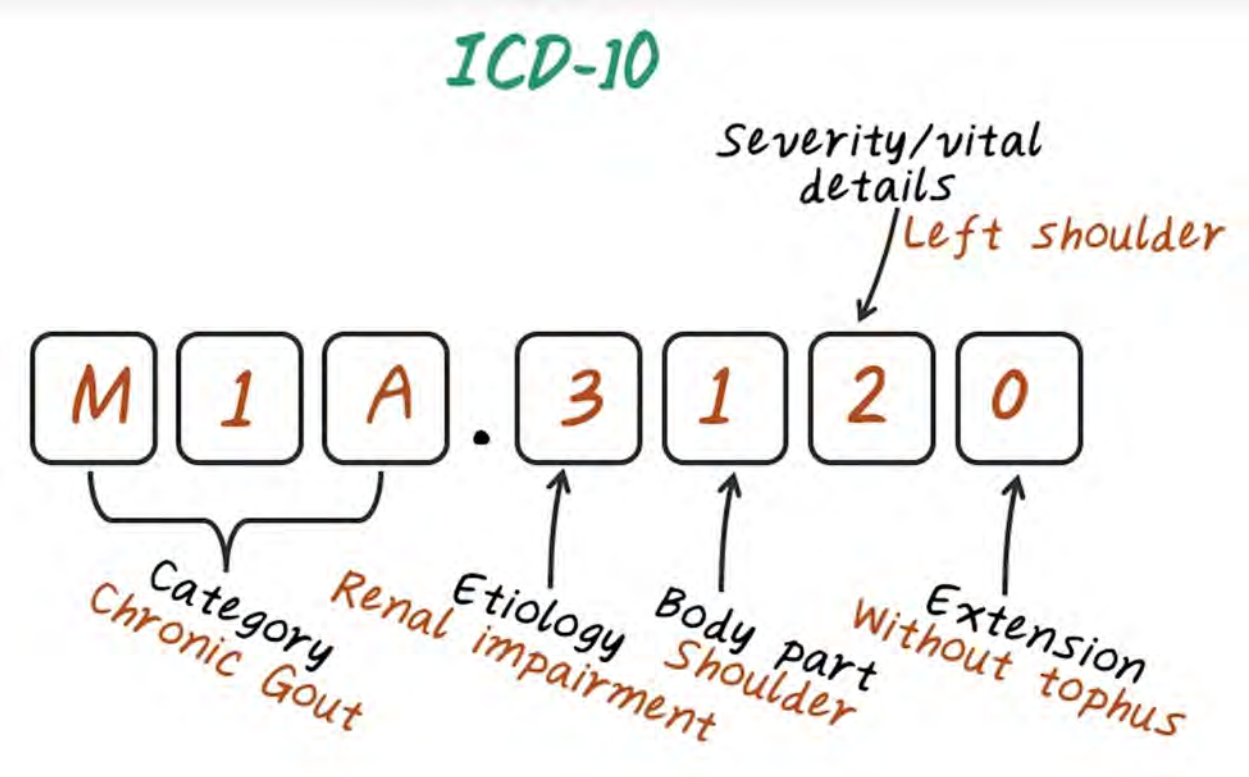
3. [3. ICD]

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 that 19. V85.1 indicate BMI between 19 and 24. And V85.2 indicate BMI between 25 and 29. Now lets 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.


ICD
Developed by World Health Organization (WHO)
ICD-10 has 141,000 codes
10
ICD
The current ICD used in the U.S. is ICD-9
9
ICD-9 has 17,000 codes

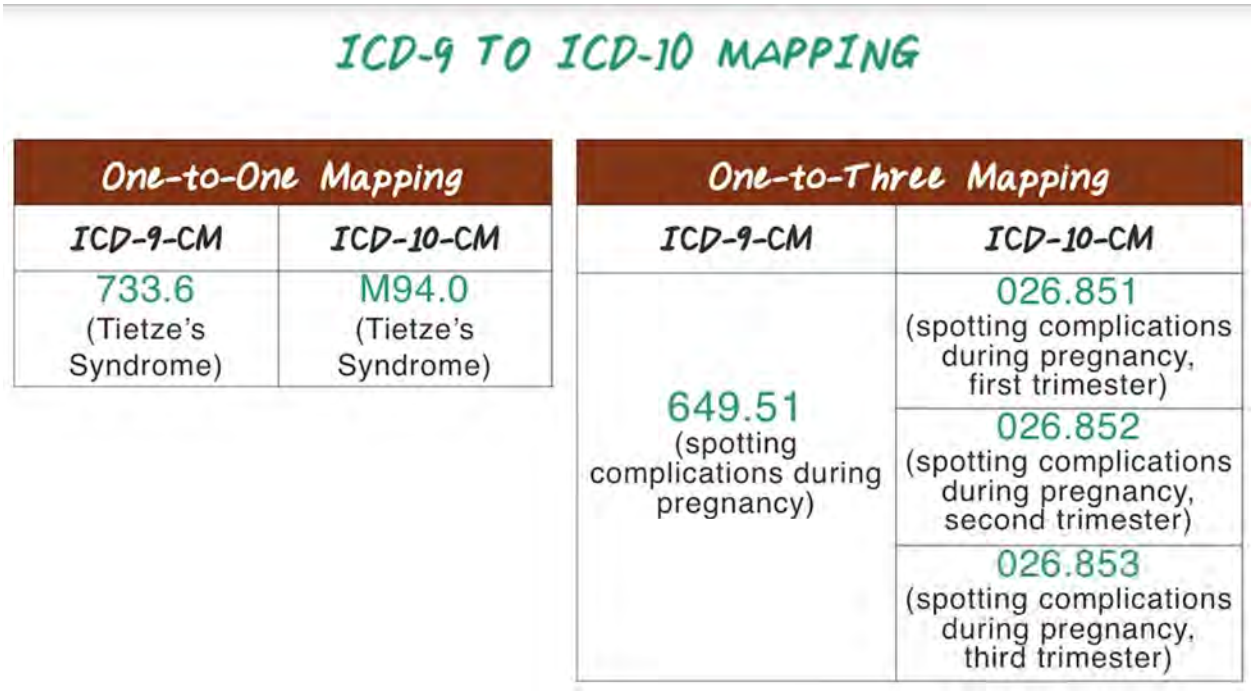
ICD-9


Type 1 Diabetes
2 5 0 . 0 1
Category
Subcategory
Subclassification
17 categories +
supplementary categories



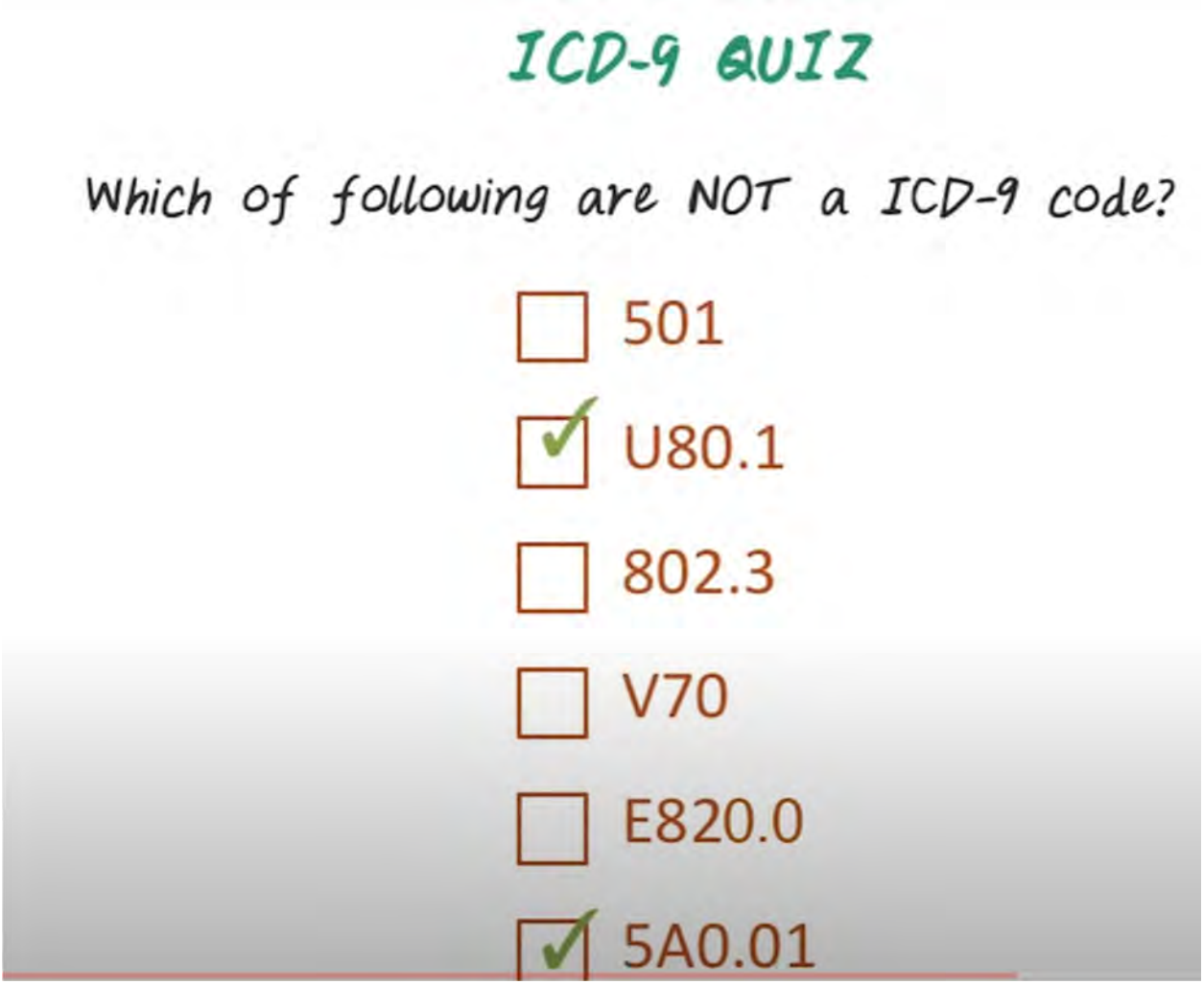
4. [4. ICD 9 to ICD 10 Mapping]

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.



5. [5. ICD 9 Quiz]

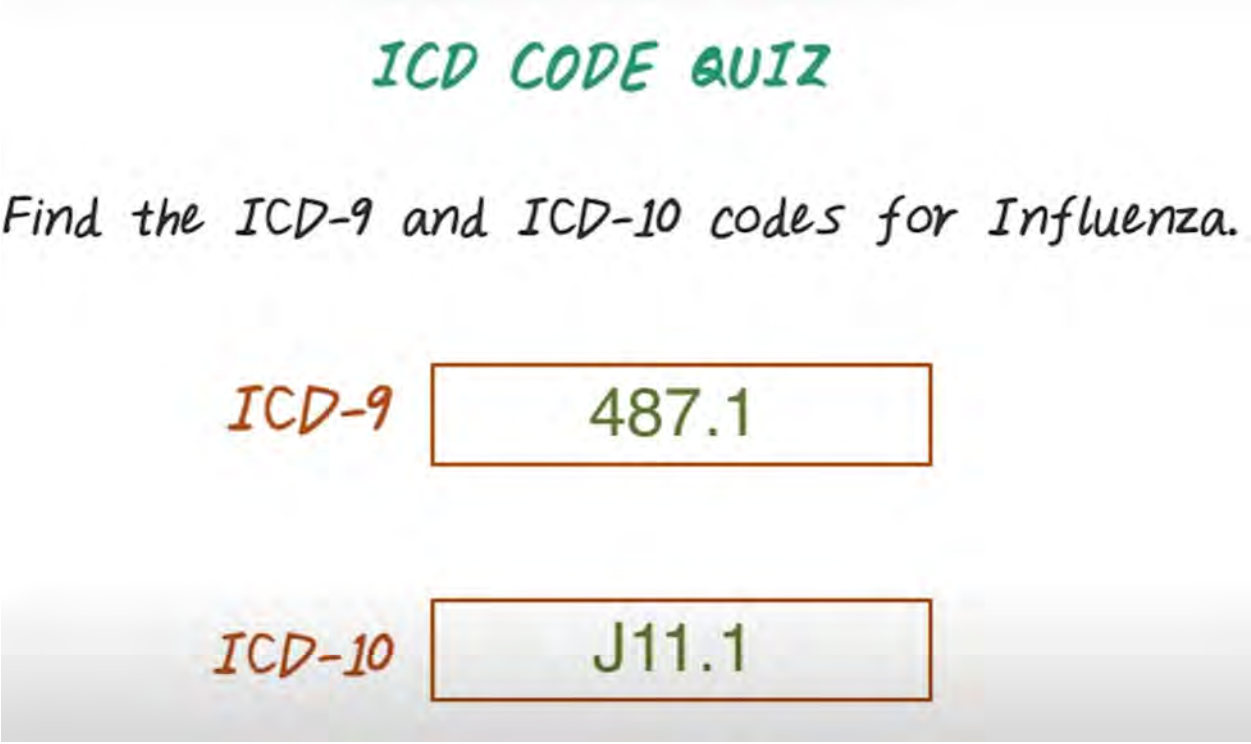
Here's a quiz question. Which of the following are not a ICD-9 code? Is this 501 or U80.1, 802.3, V70, E820.0, and 5A0.01?



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.

6. 6. ICD Code Quiz

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

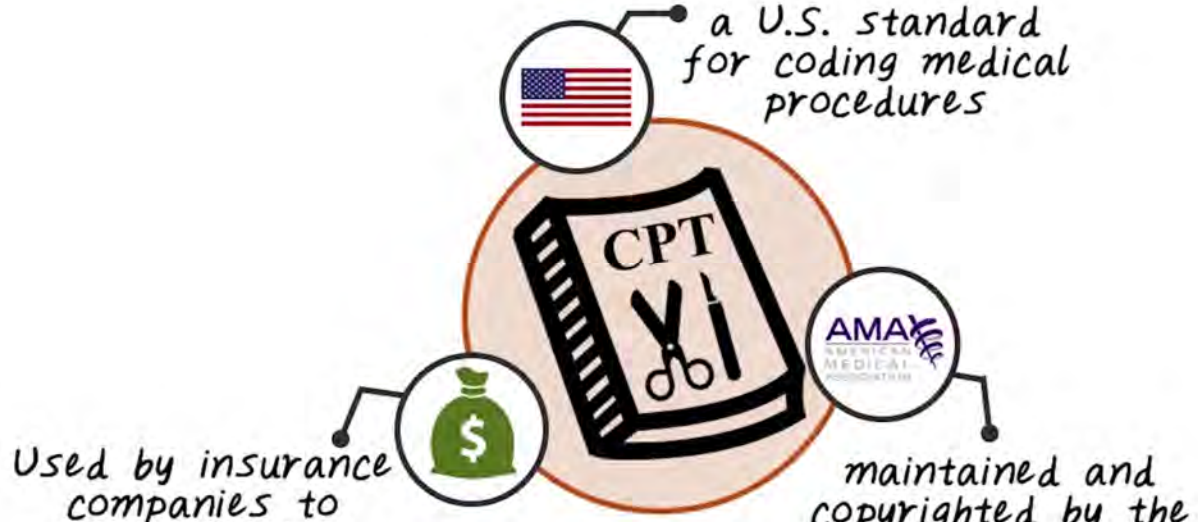


Here's the answer for ICD-9, 487.1 stands for Influenza. And for ICD-10, the answer is J11.1.

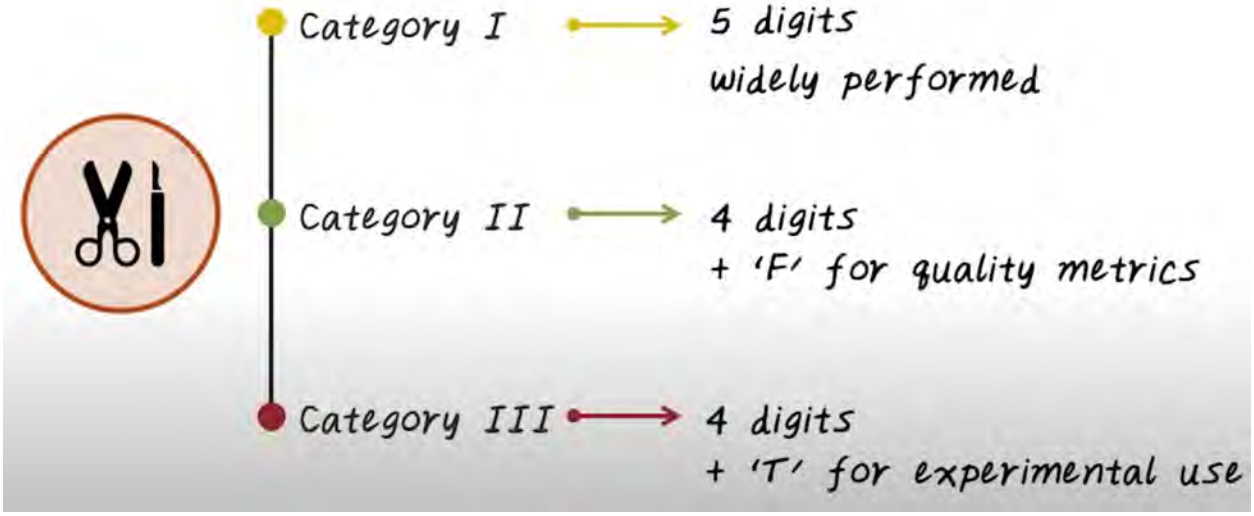
7. 7. CPT

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.

CPT


a U.S. standard for coding medical procedures
CPT
AMA
Used by insurance companies to determine how much physicians will be paid
maintained and copyrighted by the American Medical Association (AMA)

CPT


Category I
5 digits
widely performed
Category II
4 digits
+ 'F' for quality metrics
Category III
4 digits
+ 'T' for experimental use

CATEGORY I CPT

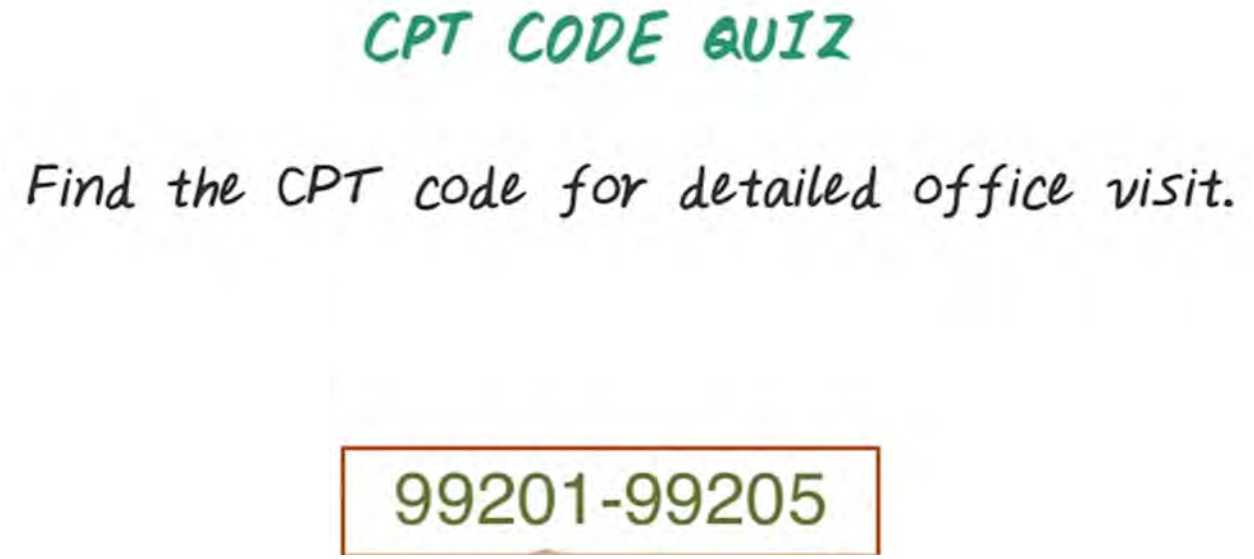
Evaluation and Management	99201-99499
Anesthesia	00100-01999; 99100-99140
Surgery	10021-69990
Radiology	70010-79999
Pathology and Laboratory	80047-89398
Medicine	90281-99199; 99500-99607

CPT CATEGORY II CODE

Composite measures	0001F-0015F
Patient management	0500F-0575F
Patient history	1000F-1220F
Physical examination	2000F-2050F
Diagnostic/screening process or results	3006F-3573F
Therapeutic, preventative or other interventions	4000F-4306F
Follow-up or other outcomes	5005F-5100F
Patient safety	6005F-6045F
Structural Measures	7010F-7025F

8. 8. CPT Code Quiz

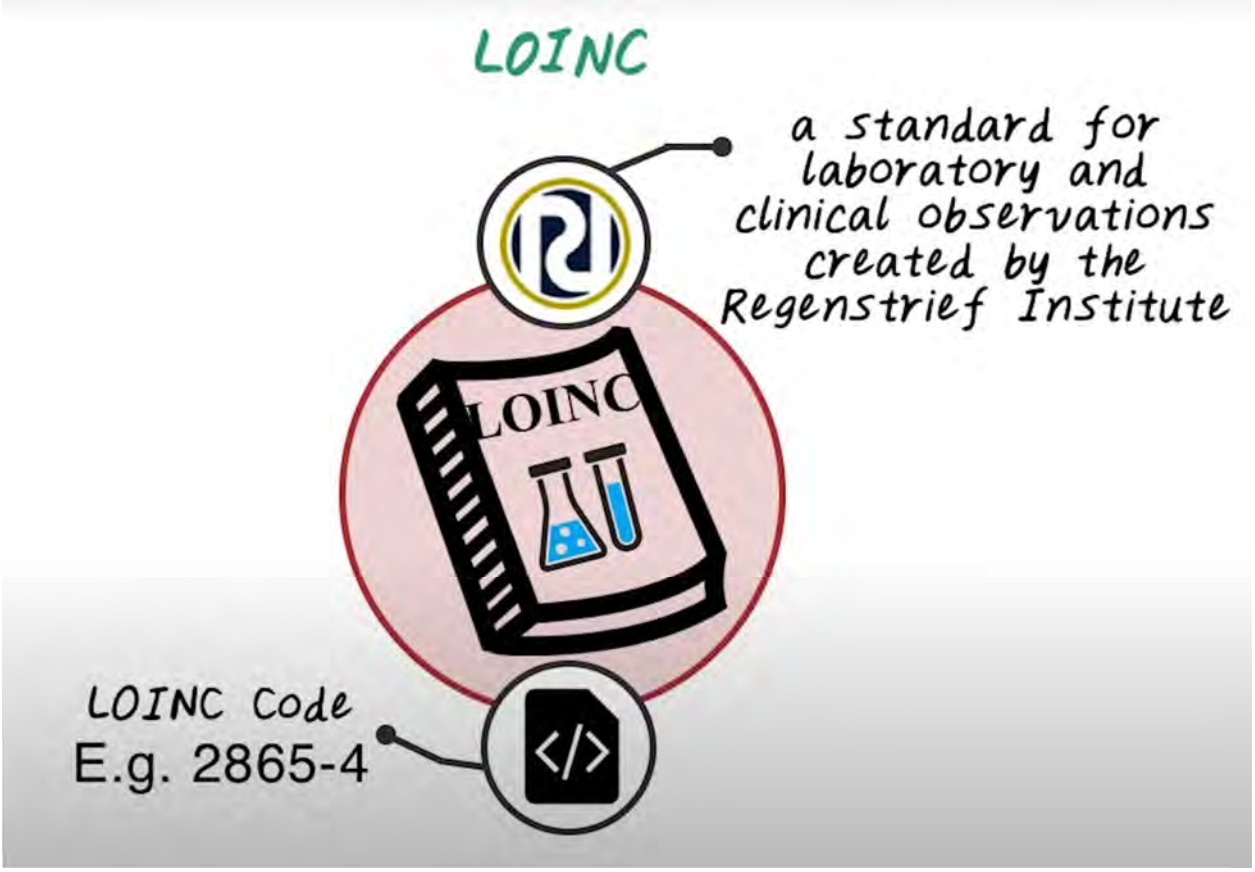
Here's a quiz on CPT code. Search online, try to find a CPT code for detailed office visit.

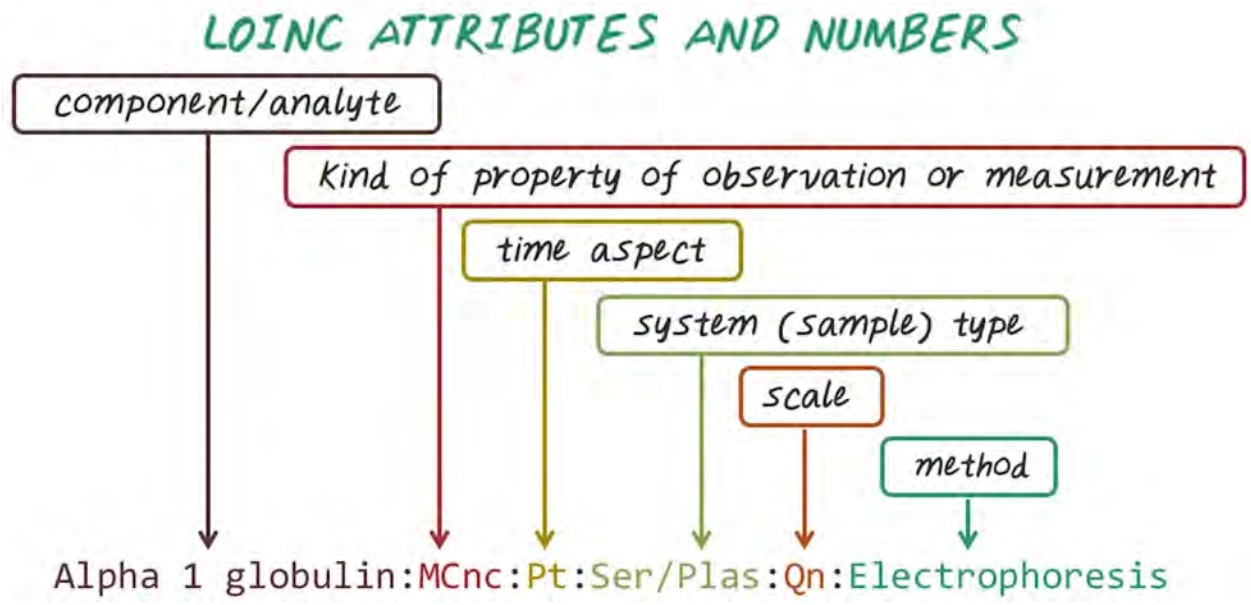


The office visit is in the range of 99201 to 99205. And all those different codes represent a office visit. However, the distinction is mainly on how much time you spent face to face with the patient. So 99201 is for ten minutes time, while 99205 corresponding to an hour time. So you can see that which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

9. 9. LOINC

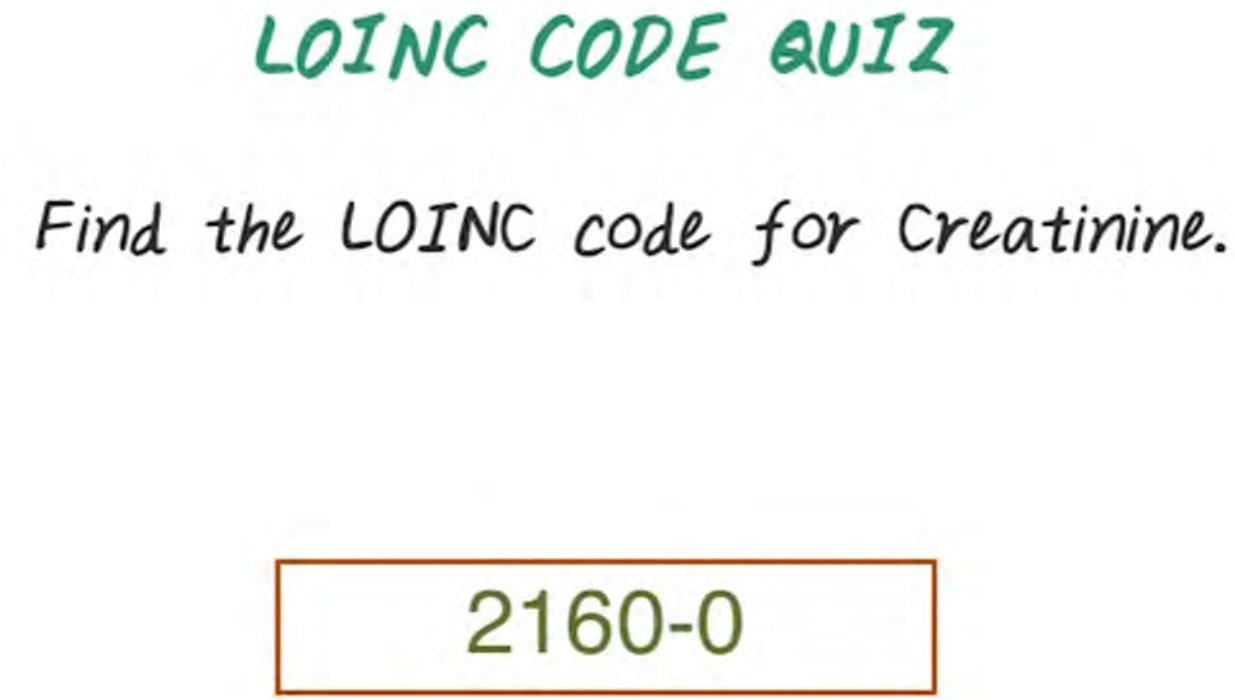
Next let's talk about LOINC. LOINC is a standard for lab and clinical observation and it's created by Regenstrief Institute which is a non-profit organization in Indiana. LOINC has been created mainly to capturing lab test. And a LOINC code contains digits like 2865-4, and this is a LOINC code. Of course 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.





10. 10. LOINC Code Quiz

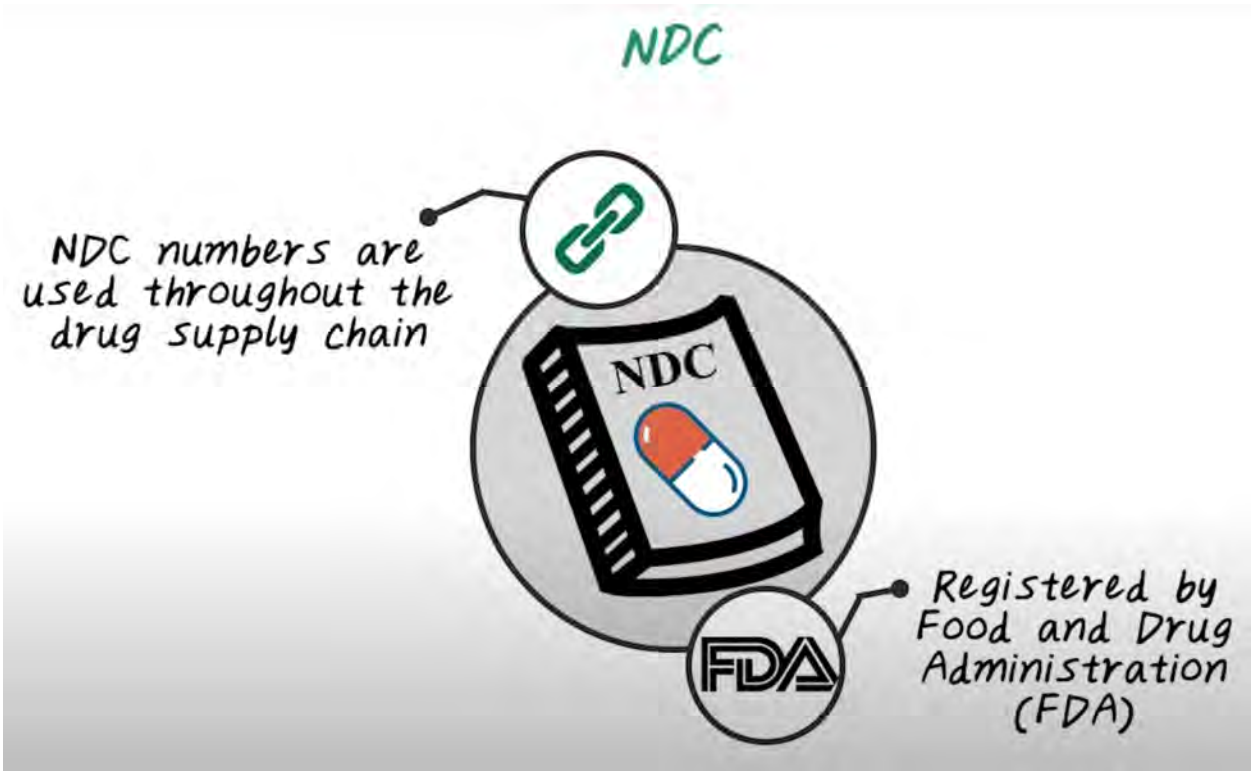
Here's the quiz on LOINC code. Try to search online to find LOINC code for lab test Creatinine.

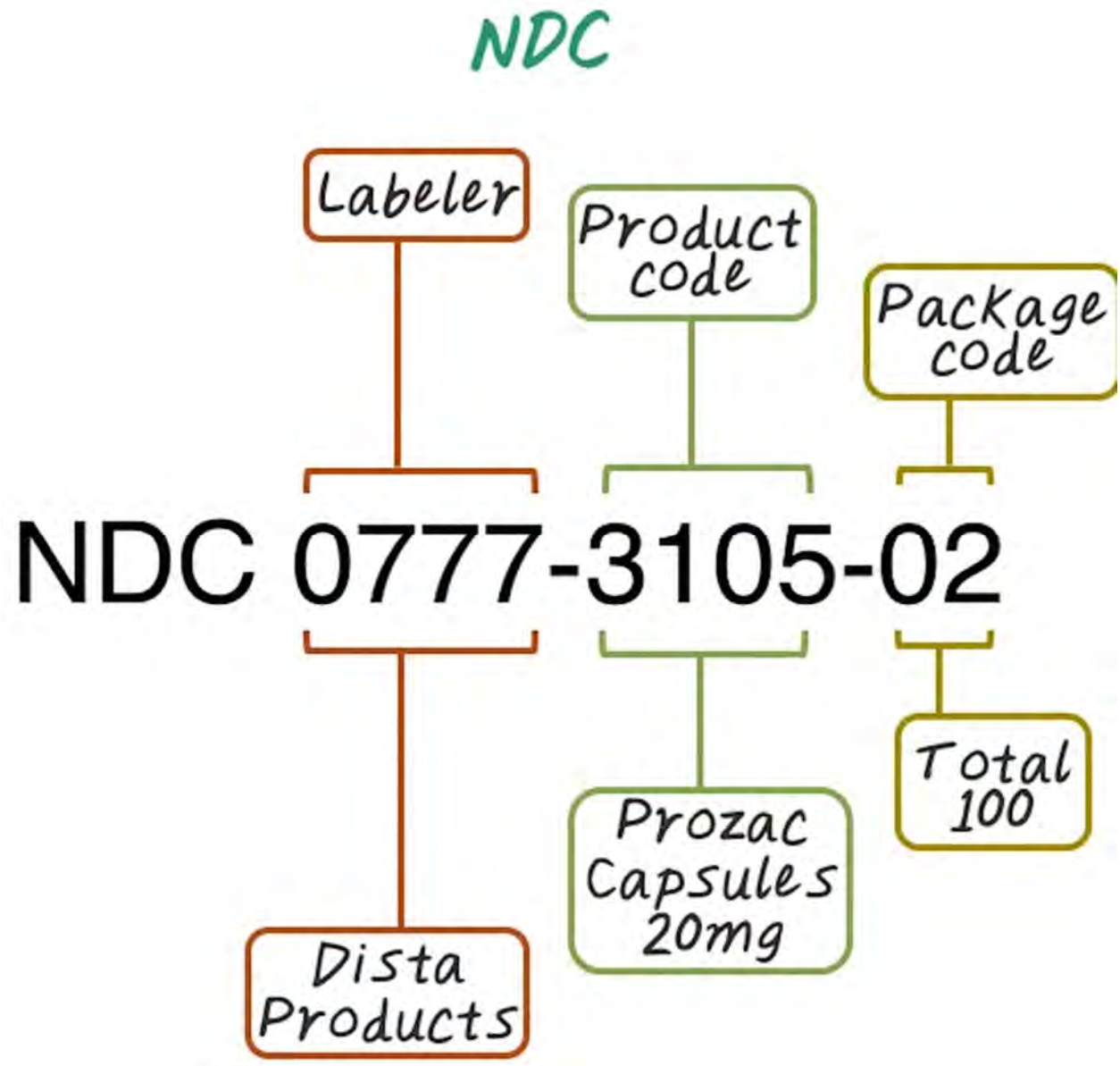


And the answer is 2160-0. You may find Azure lab tests also contains the term creatinine, and this is one of the most popular one, a test for creatinine.

11. 11. NDC

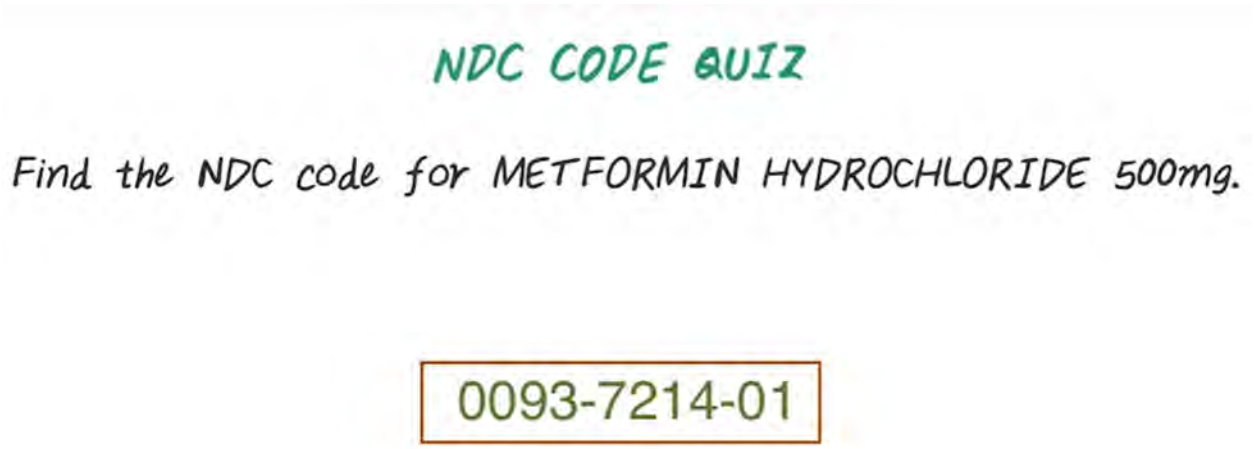
Next, let's talk about NDC code. NDC stands for National Drug Code. NDC is the standard for medications, and NDC is registered and maintained by FDA, the Food and Drug Administration. And FDA maintains a searchable database of all the NDC code on their website. NDC code are used throughout the entire 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.


NDC
NDC numbers are used throughout the drug supply chain
NDC
FDA
Registered by Food and Drug Administration (FDA)



12. [12. NDC Code Quiz]

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

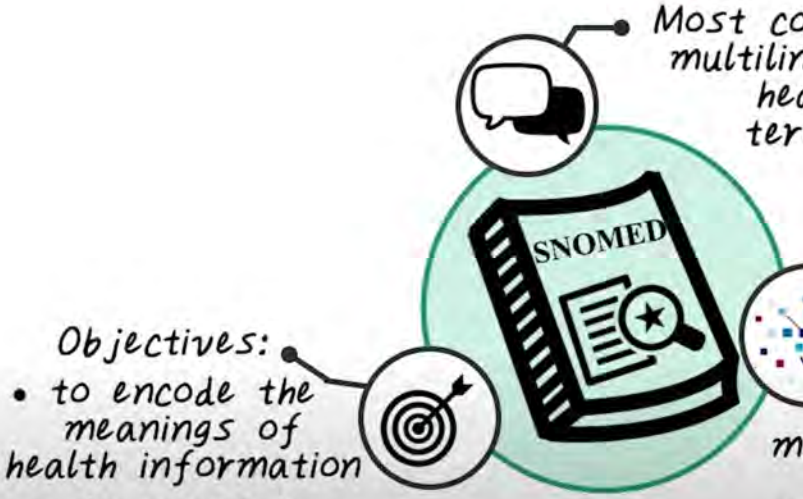


One of the answer could be, 0093-7214-01. Your answer could be different from this depending on who is the labeler, and what's the product code and this can look different, even for the same drug.

13. 13. SNOMED

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.

SNOMED



Most comprehensive, multilingual clinical healthcare terminology

Objectives:
- to encode the meanings of health information
- to support effective clinical recording of data

maintained by the IHTSDO, an international non-profit organization in Denmark

SNOMED DEVELOPMENT PROCESS

IHTSDO | Members | Reference Sets | Implementation | Users | Purposes

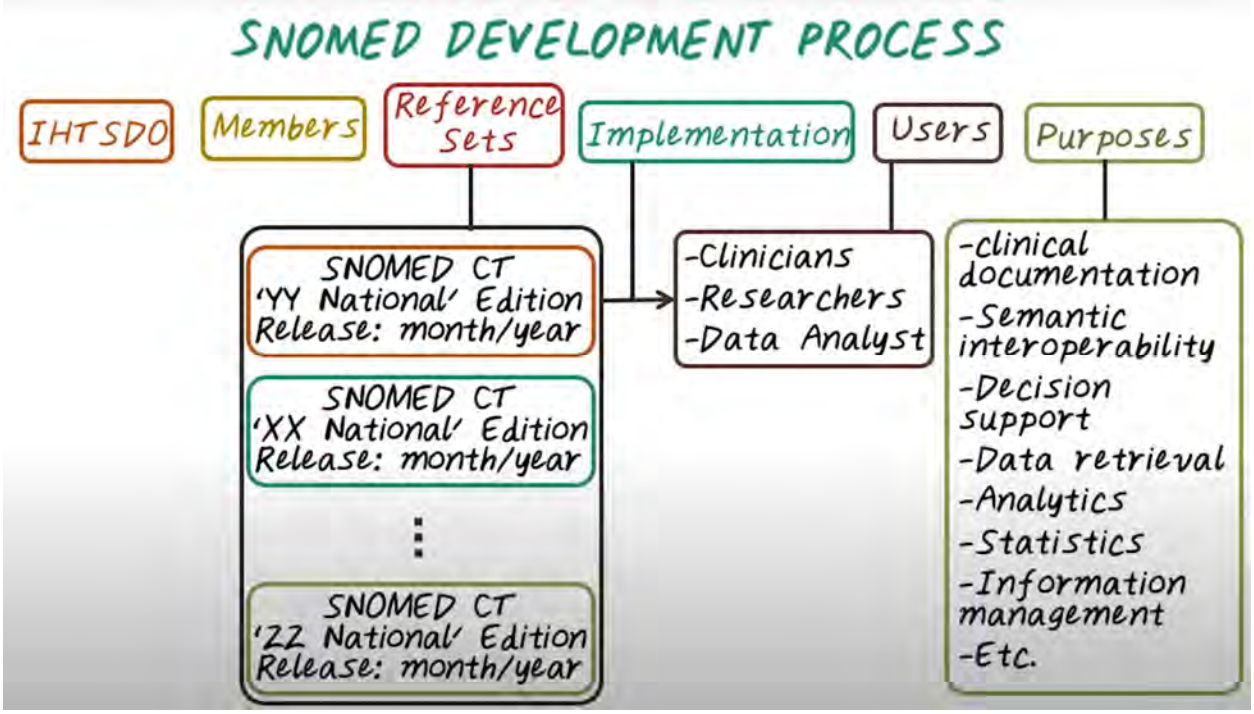
- -Development
- -Release
- -Distribution
- -Maintenance
- -Education

SNOMED CT international 'SNOMED CT Core' Release: month/year

Member Country YY National Release Center

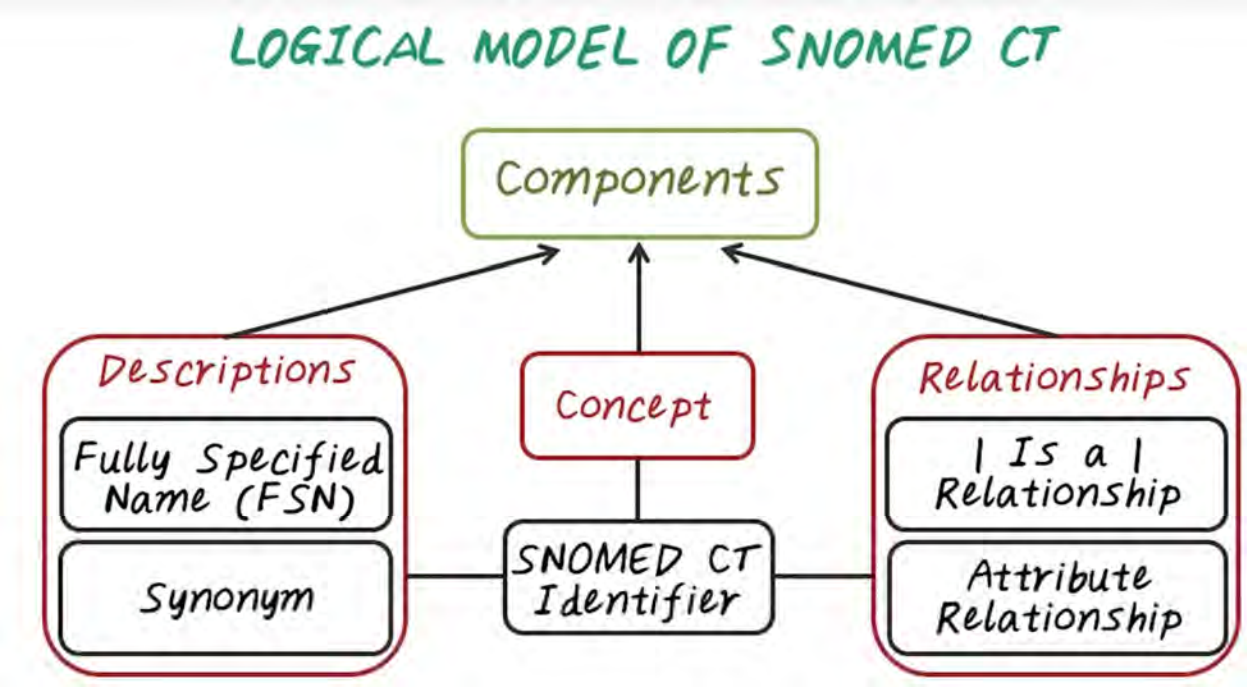
Member Country XX National Release Center

Member Country ZZ National Release Center



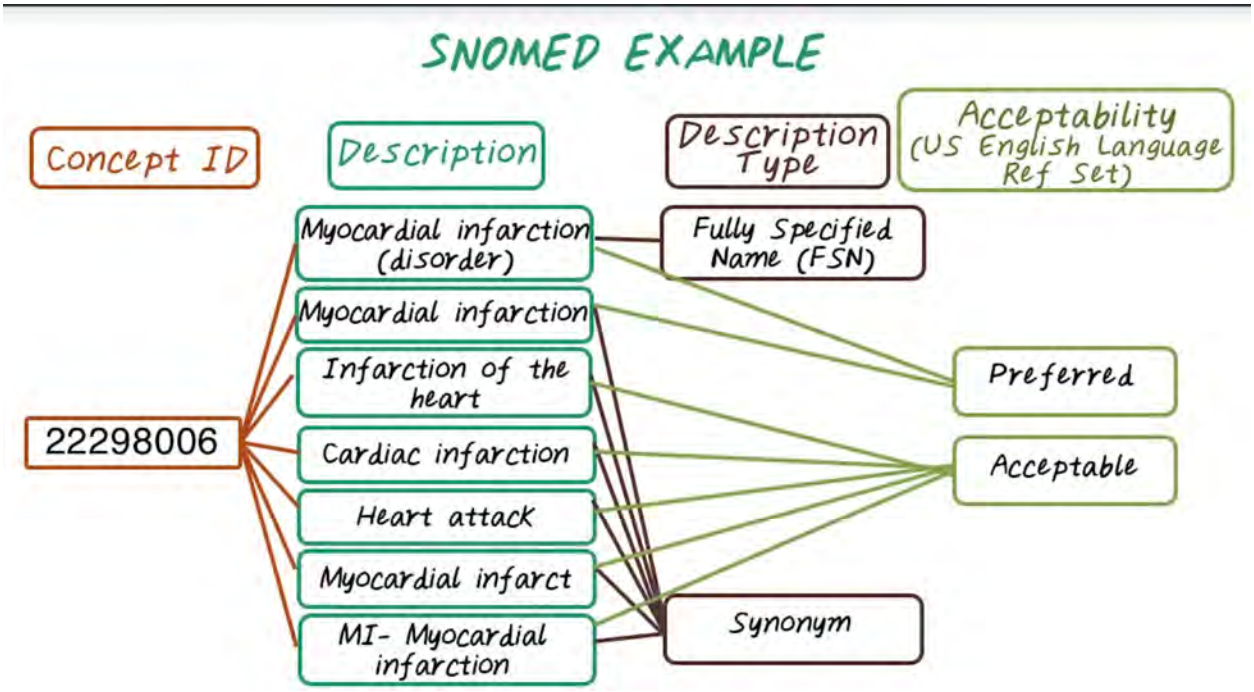
14. 14. Logical Model of SNOMED CT

Now lets 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 an 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.



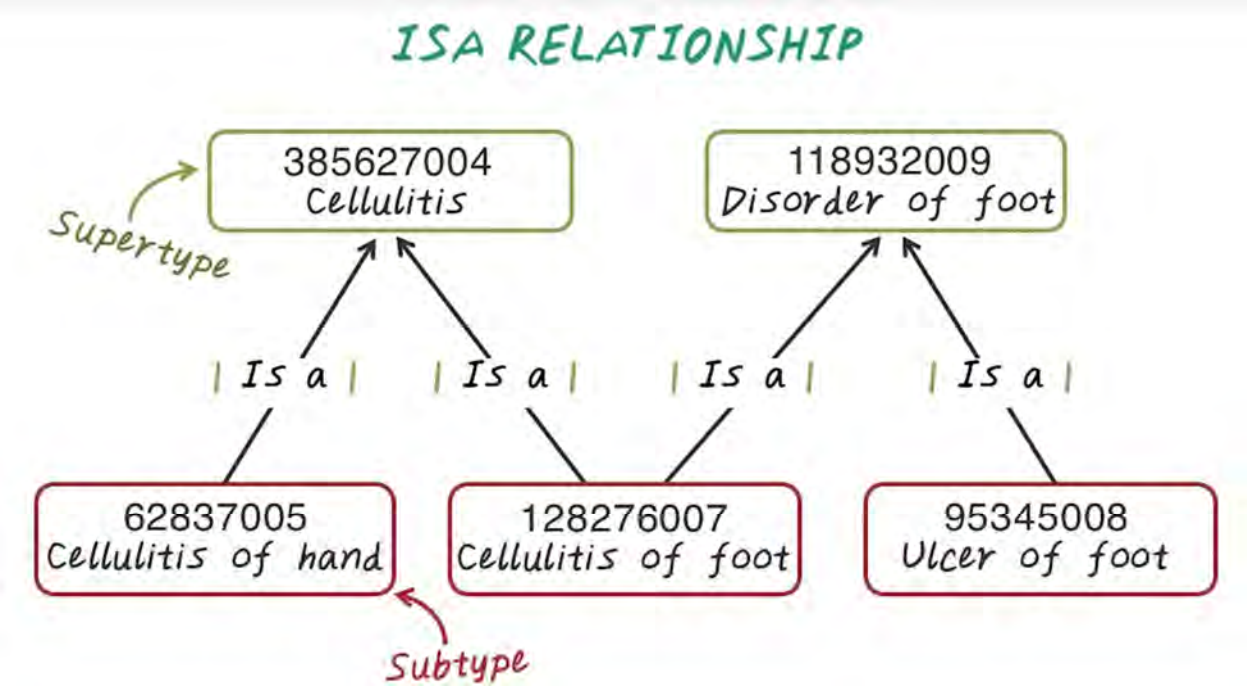
15. 15. SNOMED Example

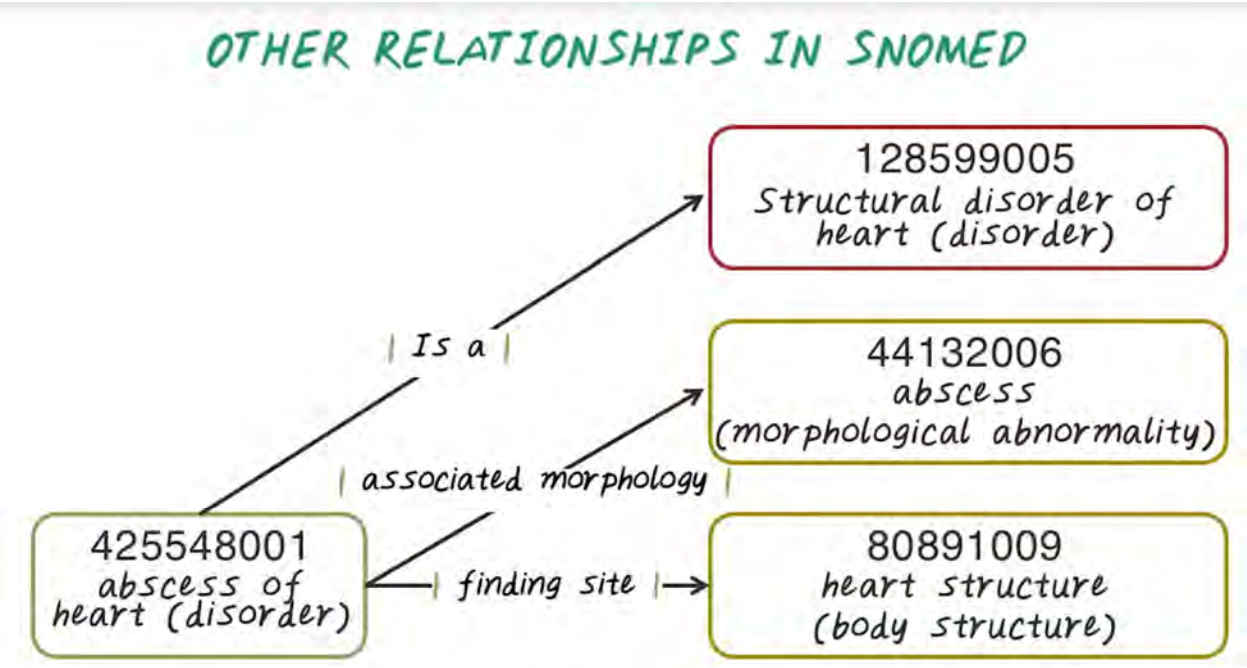
Now let's see an example of SNOMED concept. The concept ID is the following, 22298006. It is definitely in a machine readable form. This concept ID is associated with different descriptions, and among which myocardial infarction disorder is a fully specified name for this concept. Then there are many 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.



16. [16. SNOMED Relationships]

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.

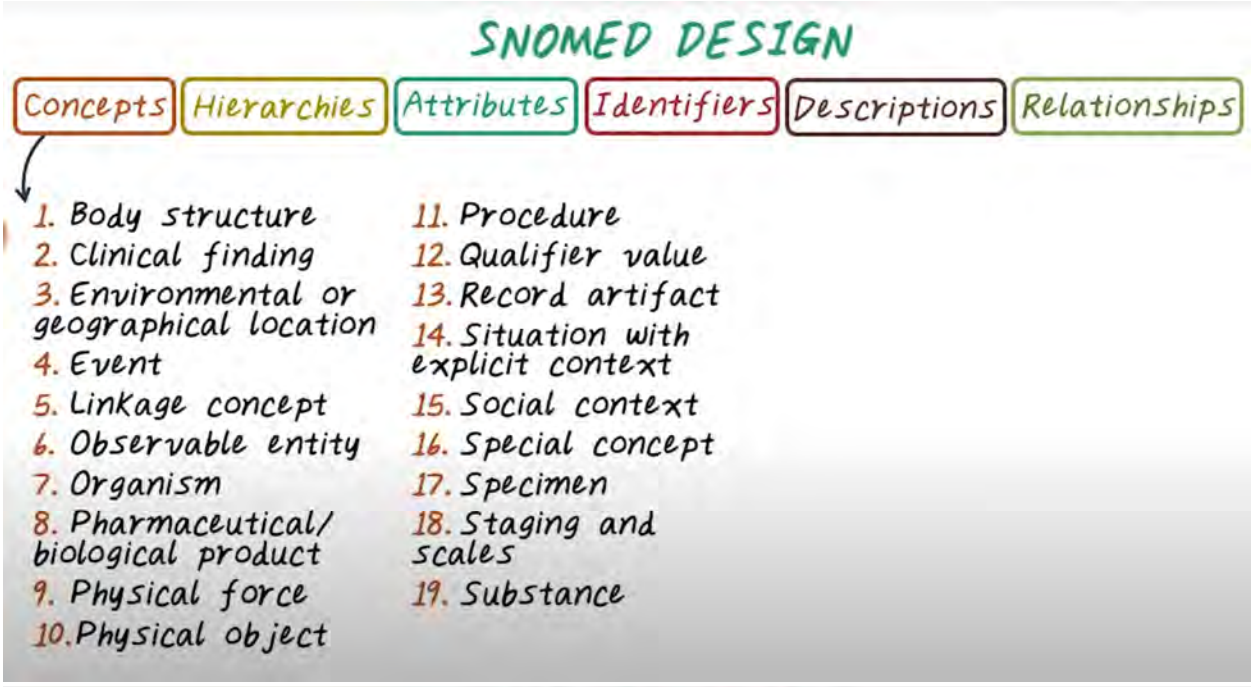


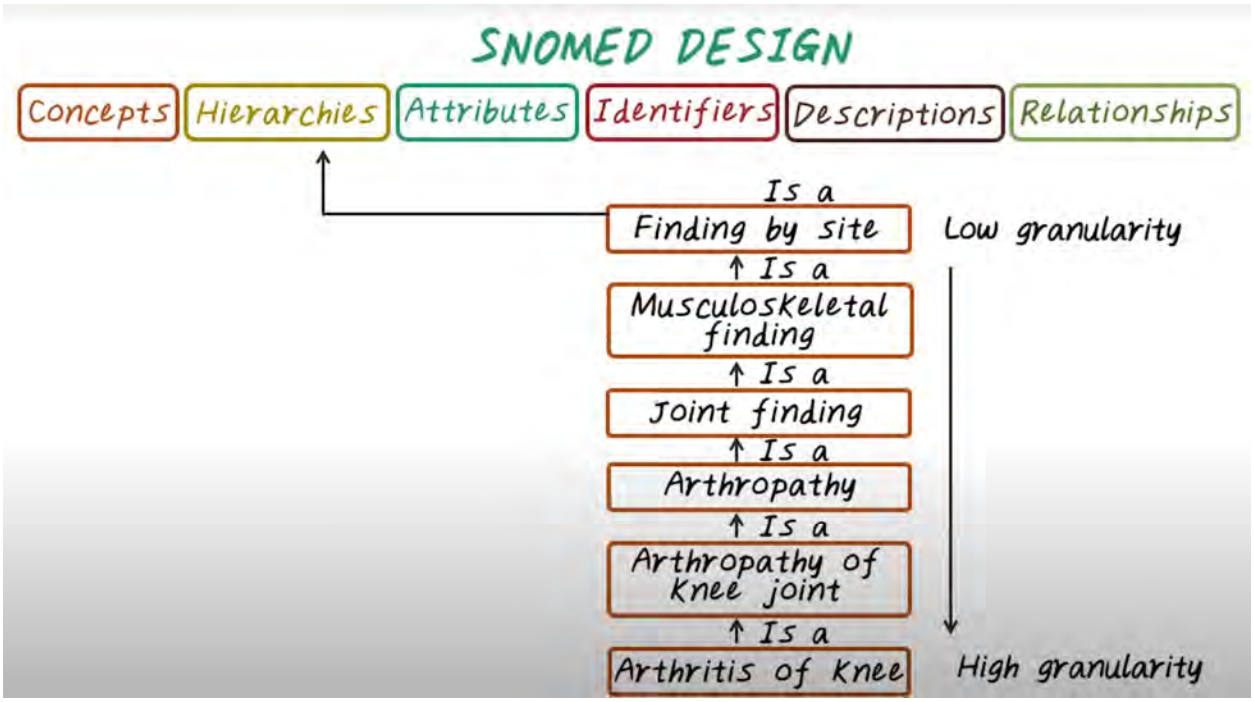


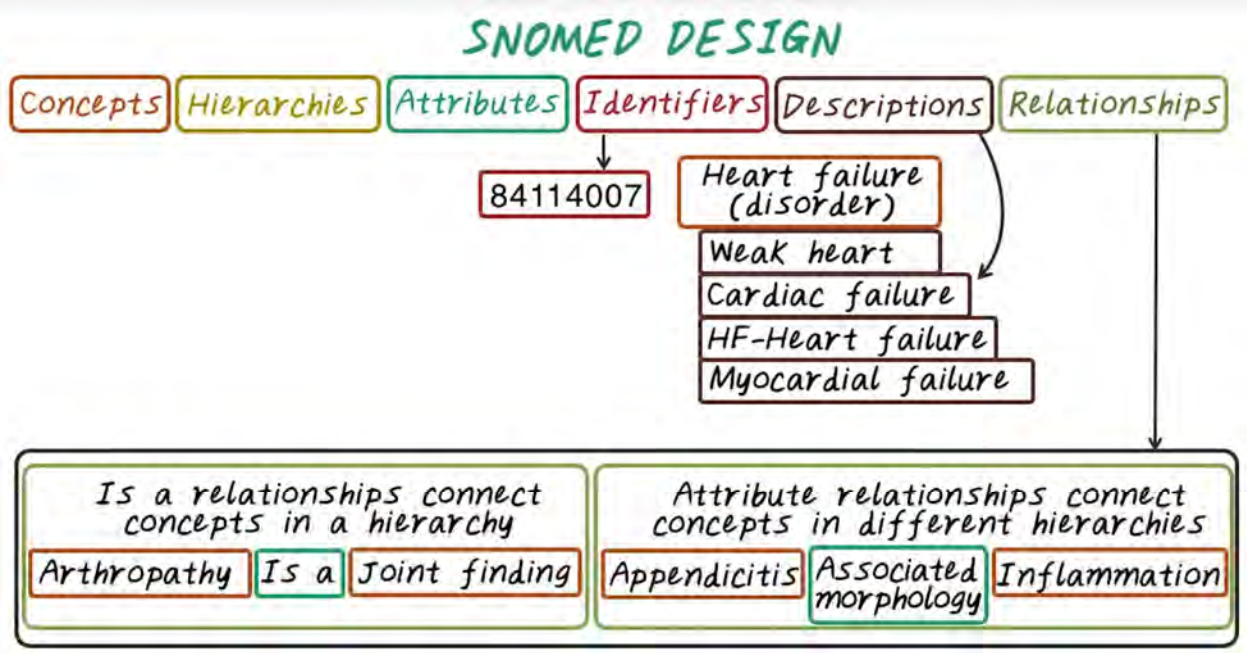
17. [17. SNOMED Design]

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 a 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.

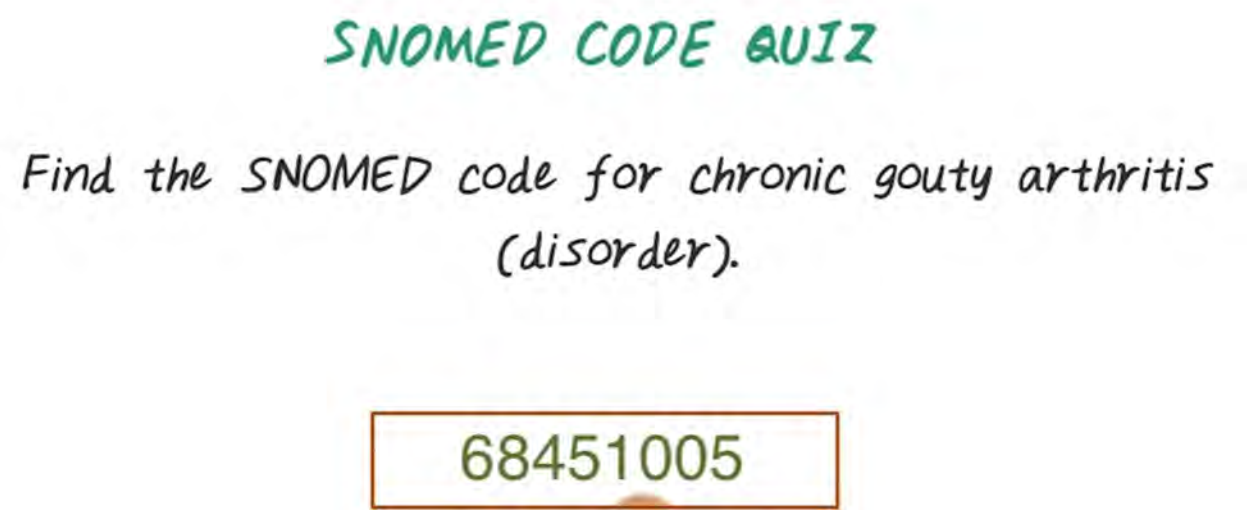






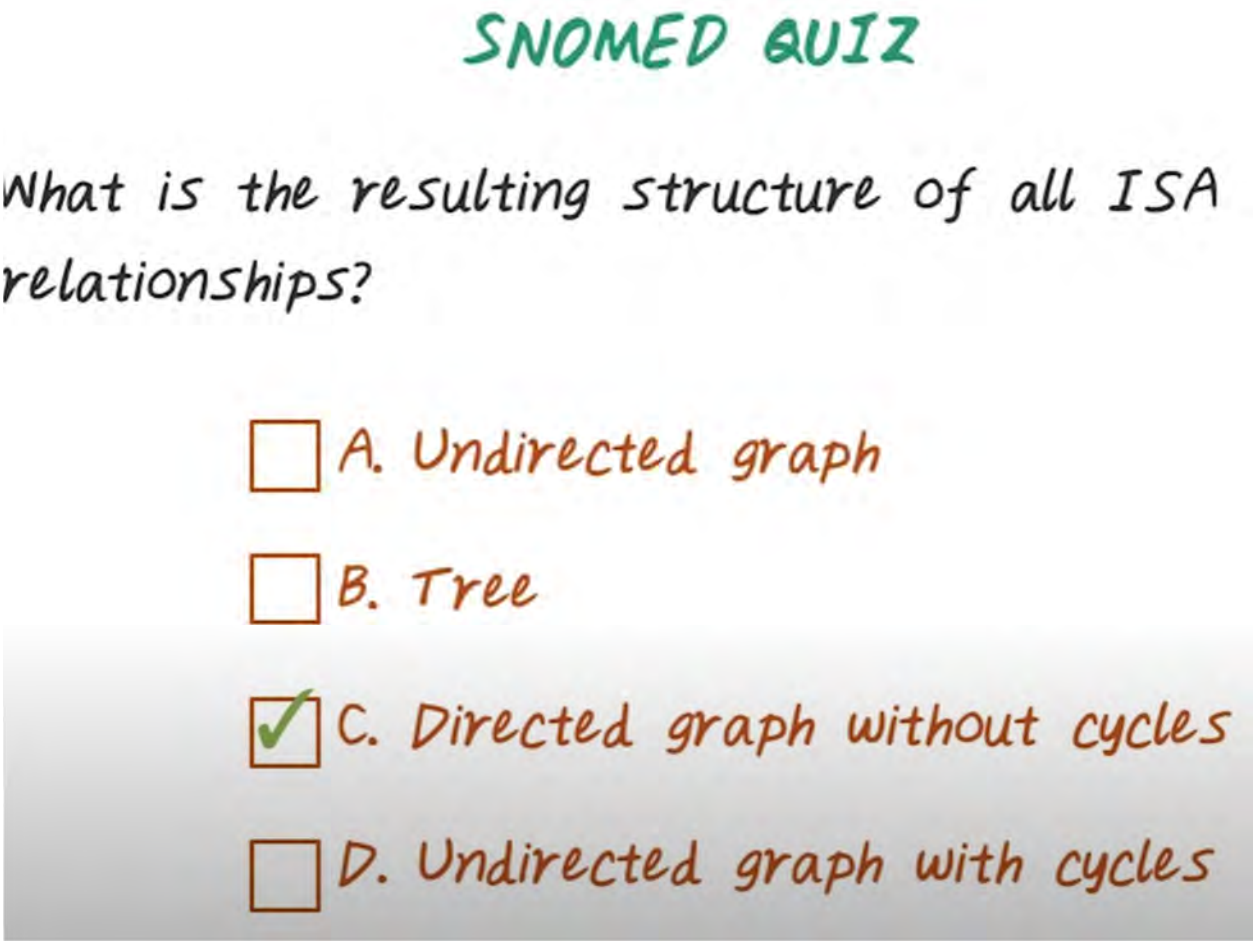
18. [18. SNOMED Code Quiz]

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



19. [19. SNOMED Quiz]

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?

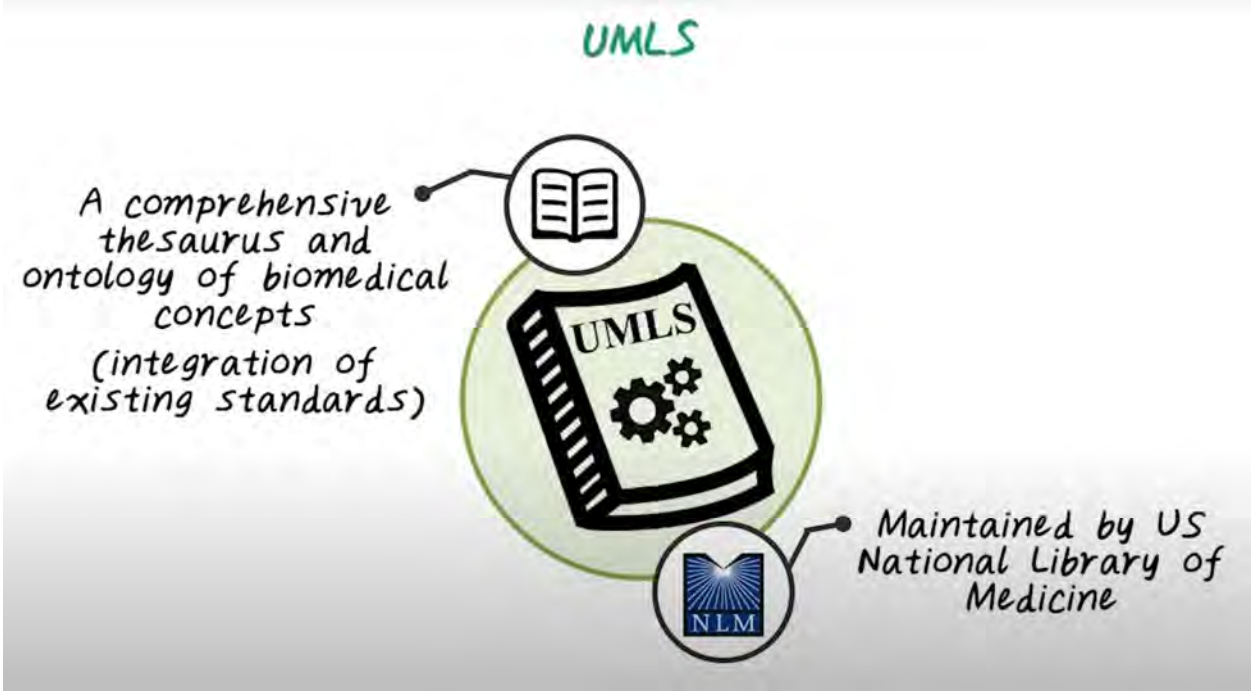


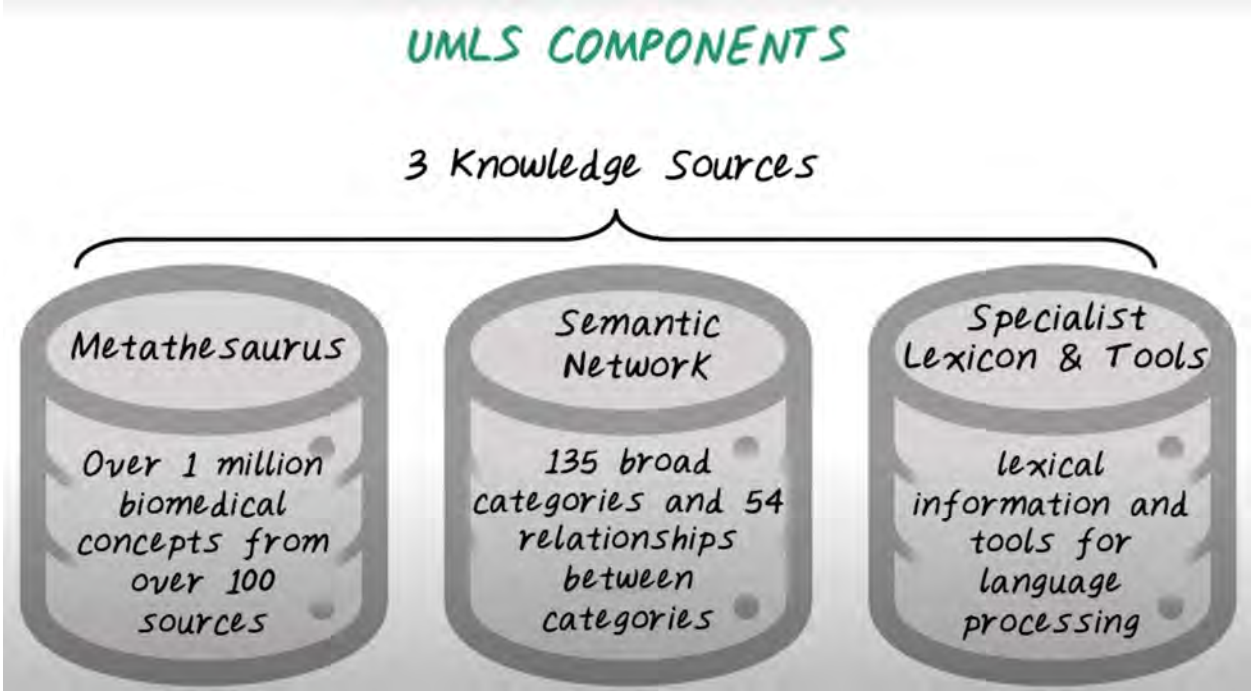
And the answer is C, it's a directed graph without cycles. It's a directed graph because all those ISA relationship has direction, so it's a directed graph. The reason it's a directed graph without cycle is because every relationship has direction, from the subtype to the super type. And given a concept, 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.

20. 20. UMLS

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.

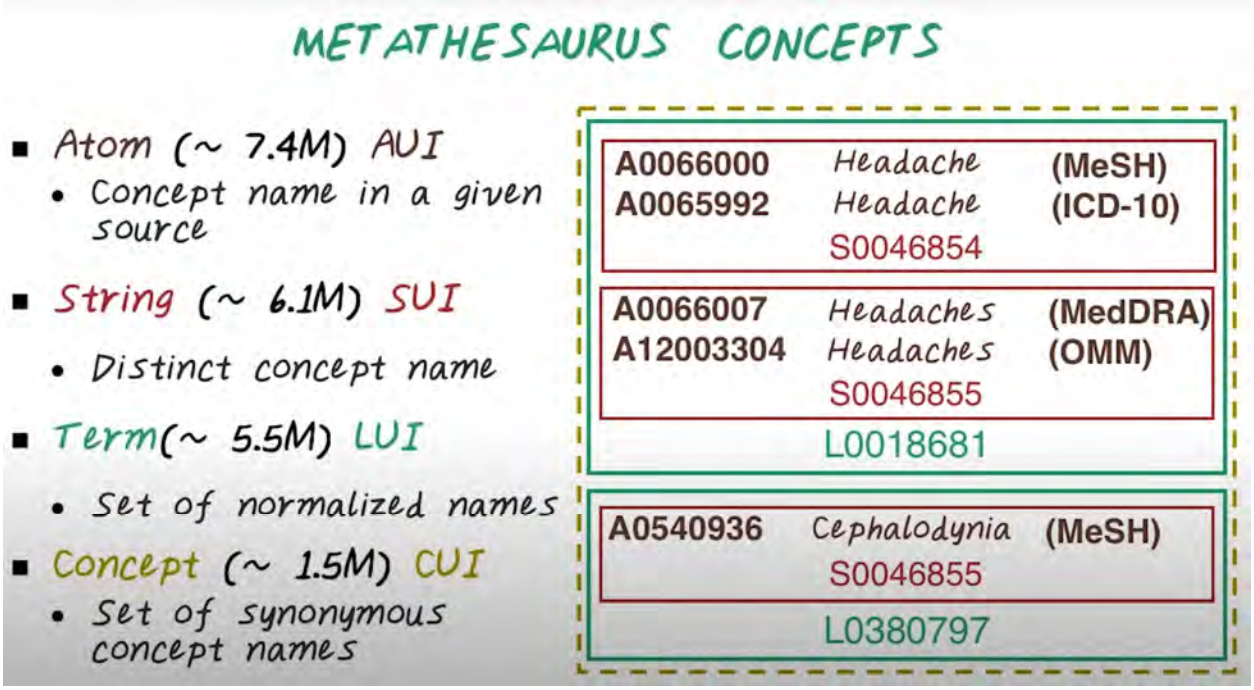




21. [21. Metathesaurus Concepts]

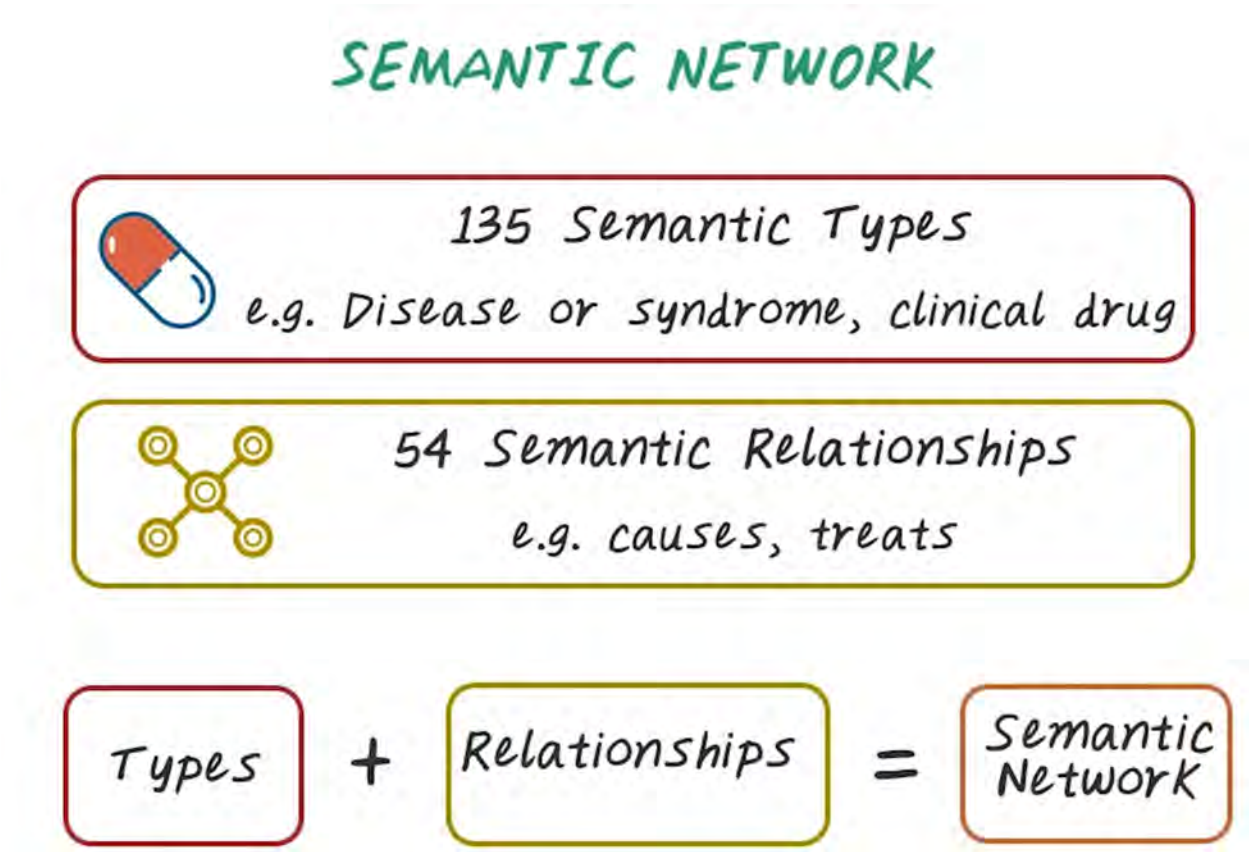
Metathesaurus Concepts. The idea is very similar to SNOMED. Each concept has a specific identifier and organized into a hierarchy. At the lowest level we have atom, at over 7.4 million atoms or AUI. Those are concept name in a specific source. For example, all this different AUIs mapped to something 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.



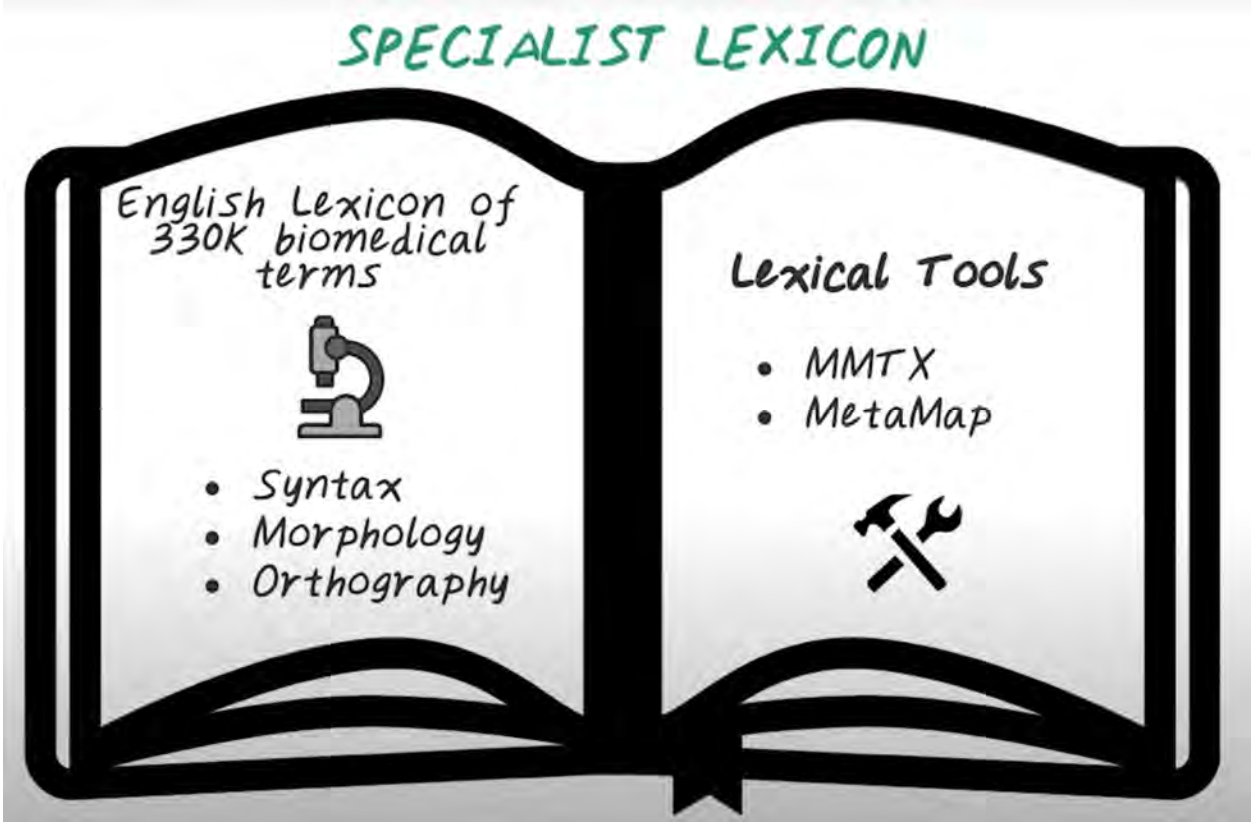
22. 22. Semantic Network

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 [INAUDIBLE] 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.



23. [23. Specialist Lexicon]

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



Graph Analysis

1. 1. Introduction to Graph Analysis

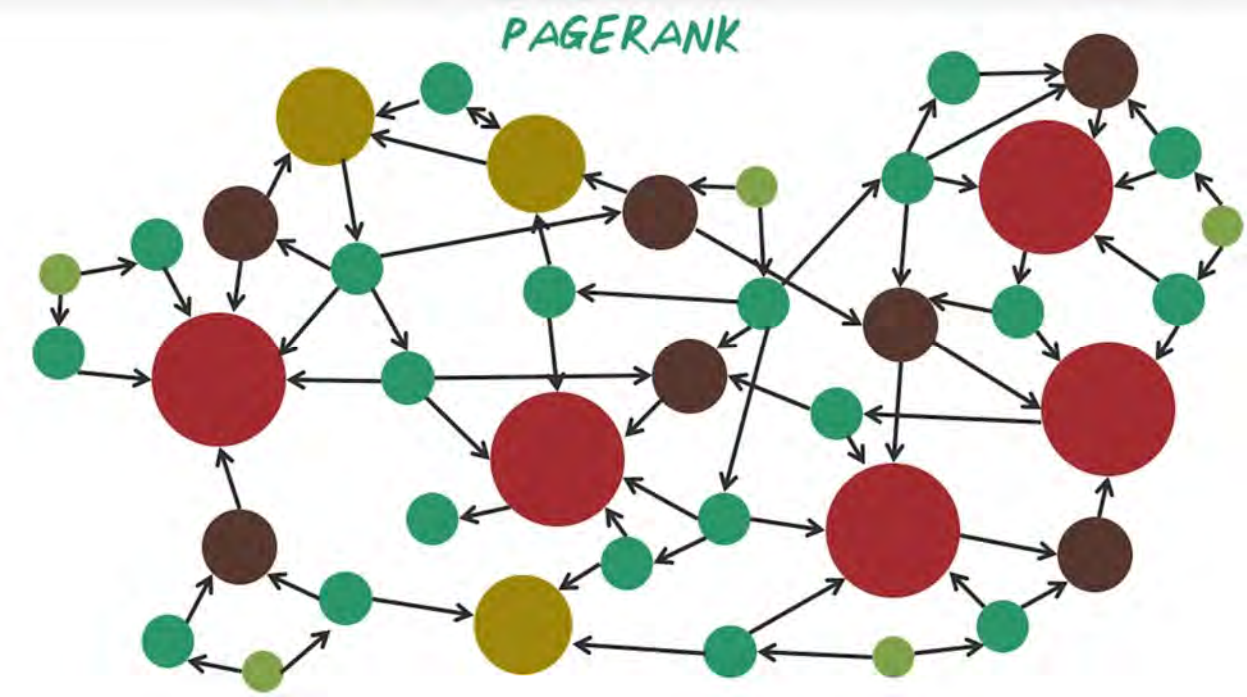
In this lesson we'll talk about graph analysis. Graph analysis, is a set of methods, that are commonly used in search engines, and social networks. And in fact, we'll start by describing graph analysis examples, in search engine tasks. We describe some core algorithms, used likely by search engines. 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.
\

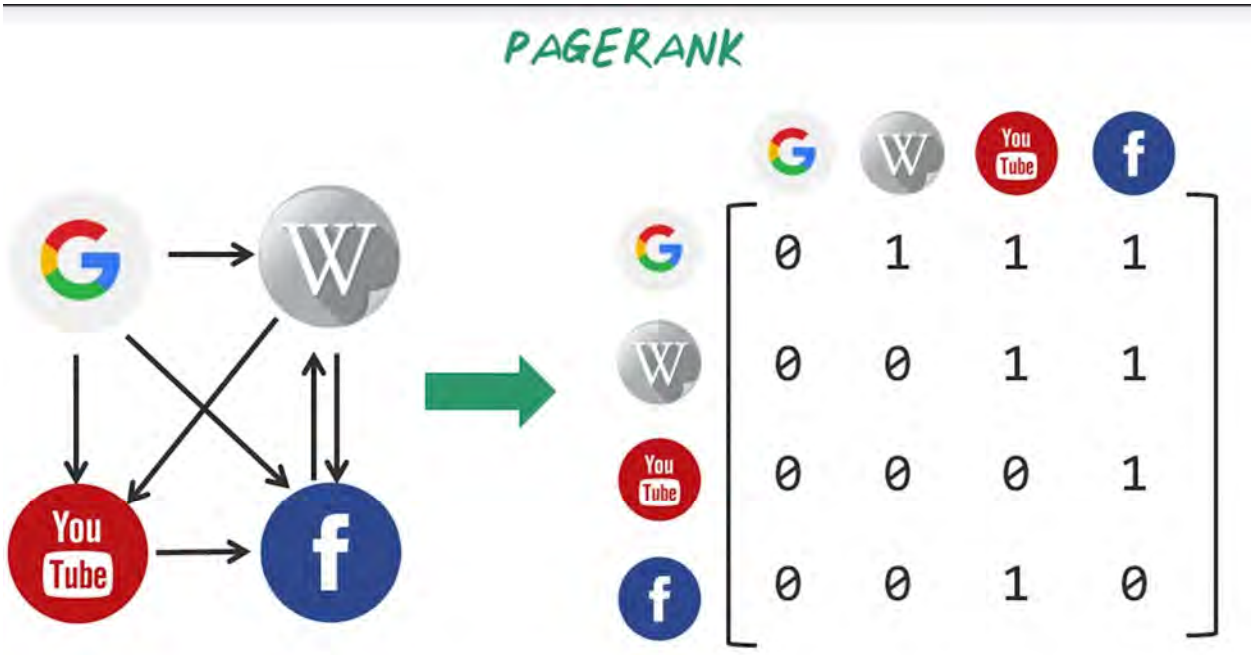
2. 2. Agenda

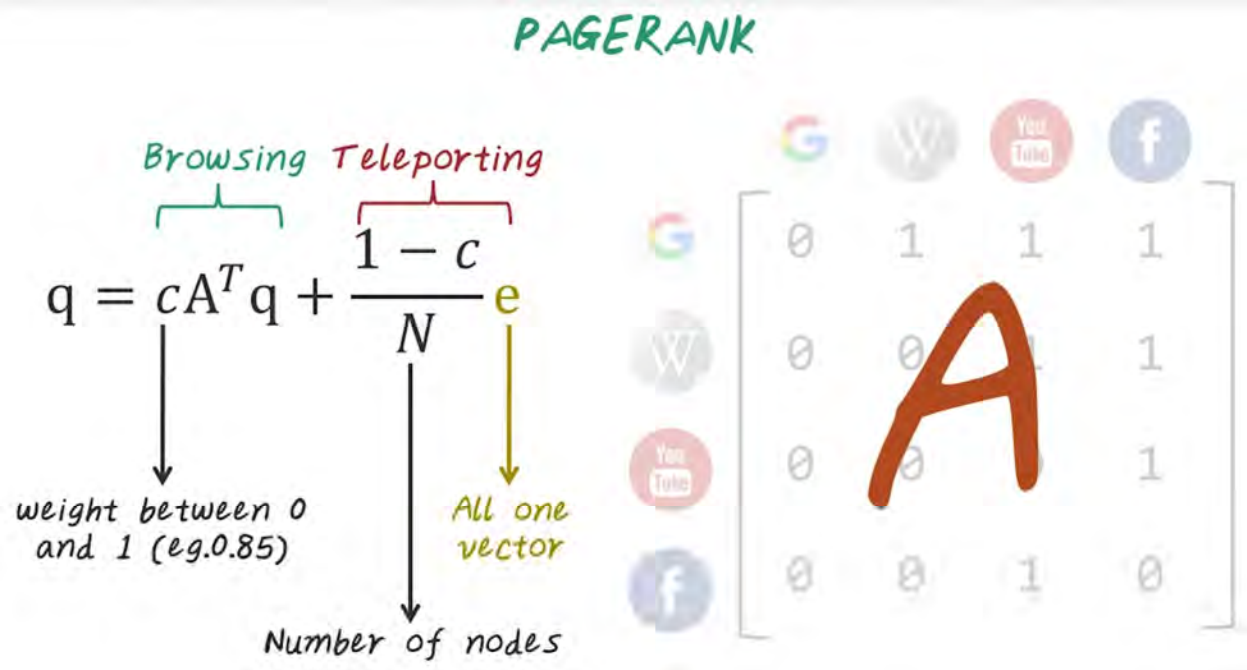
Today we'll talk about two important graph based algorithms. First, we'll talk about PageRank. Given a large directive graph, PageRank ranks all the node on this graph based on their importance. Second, we'll talk about spectral clustering, which is clustering algorithm based on graph partitioning. Let's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

3. 3. PageRank

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 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 PageRanks, 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 the 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.


PAGERANK

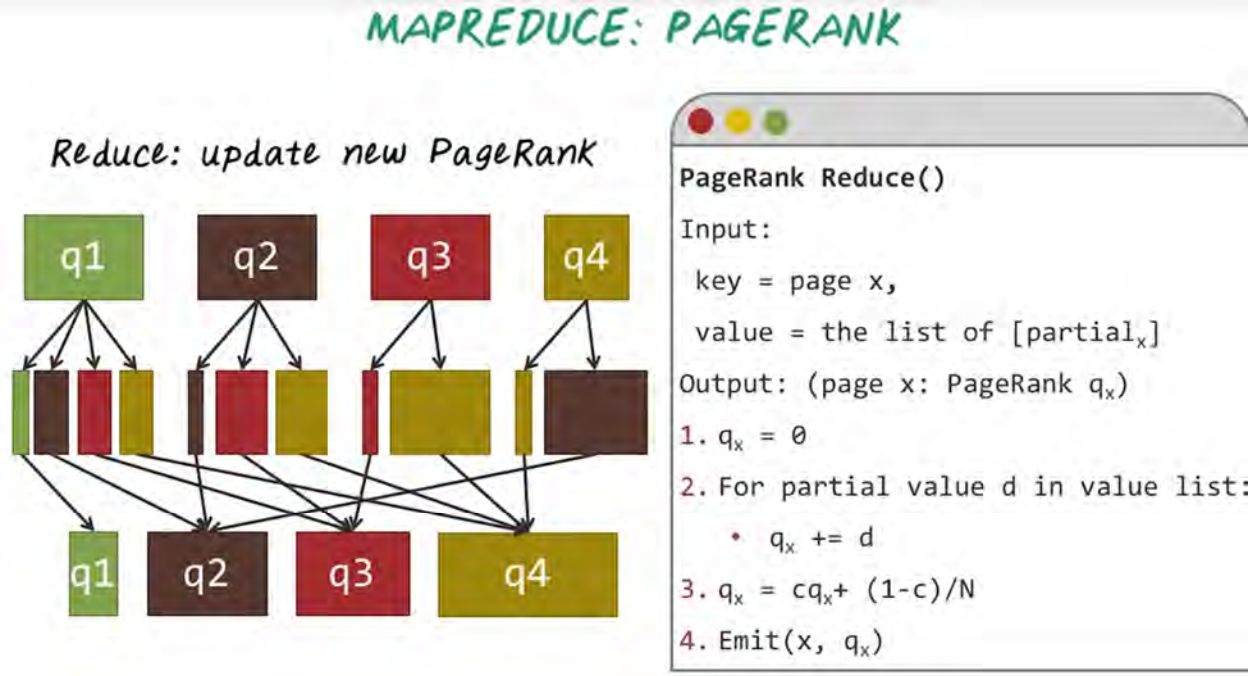

PAGERANK
You Tube
You Tube
0 1 1 1
0 0 1 1
0 0 0 1
0 0 1 0



4. 4. MapReduce PageRank

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 PageRanks. 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.



5. [5. PageRank Quiz]

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.

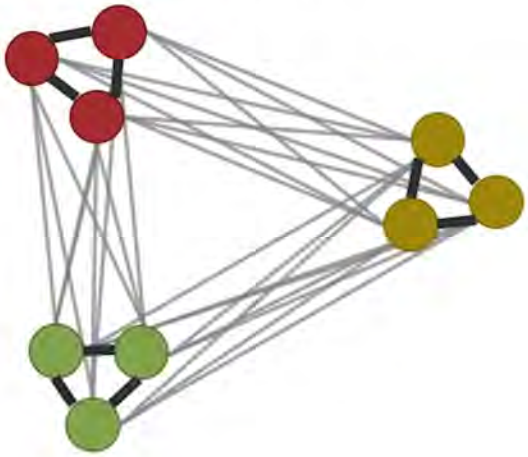
Let's figure out the answer together. The top ranked page is YouTube because it has three incoming links. Likewise, you can see Google ranked the last because there's no incoming links to this page, so it ranks number six. And similarly Twitter ranked the number fifth, second to the last because it only has one incoming link. So the rest of the pages, Wikipedia, Amazon and Facebook, they all have two incoming links. You my 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.

6. 6. Spectral Clustering

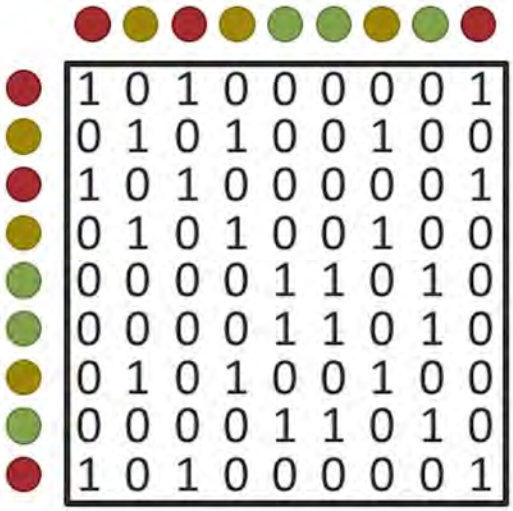
Next, let's talk about spectral clustering. In traditional clustering algorithms, given the input data and matrix, for example this disease by patient matrix. Every row corresponding to a patient, every column corresponding to a disease. We want to learn a function, f, that partition this matrix into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

SPECTRAL CLUSTERING

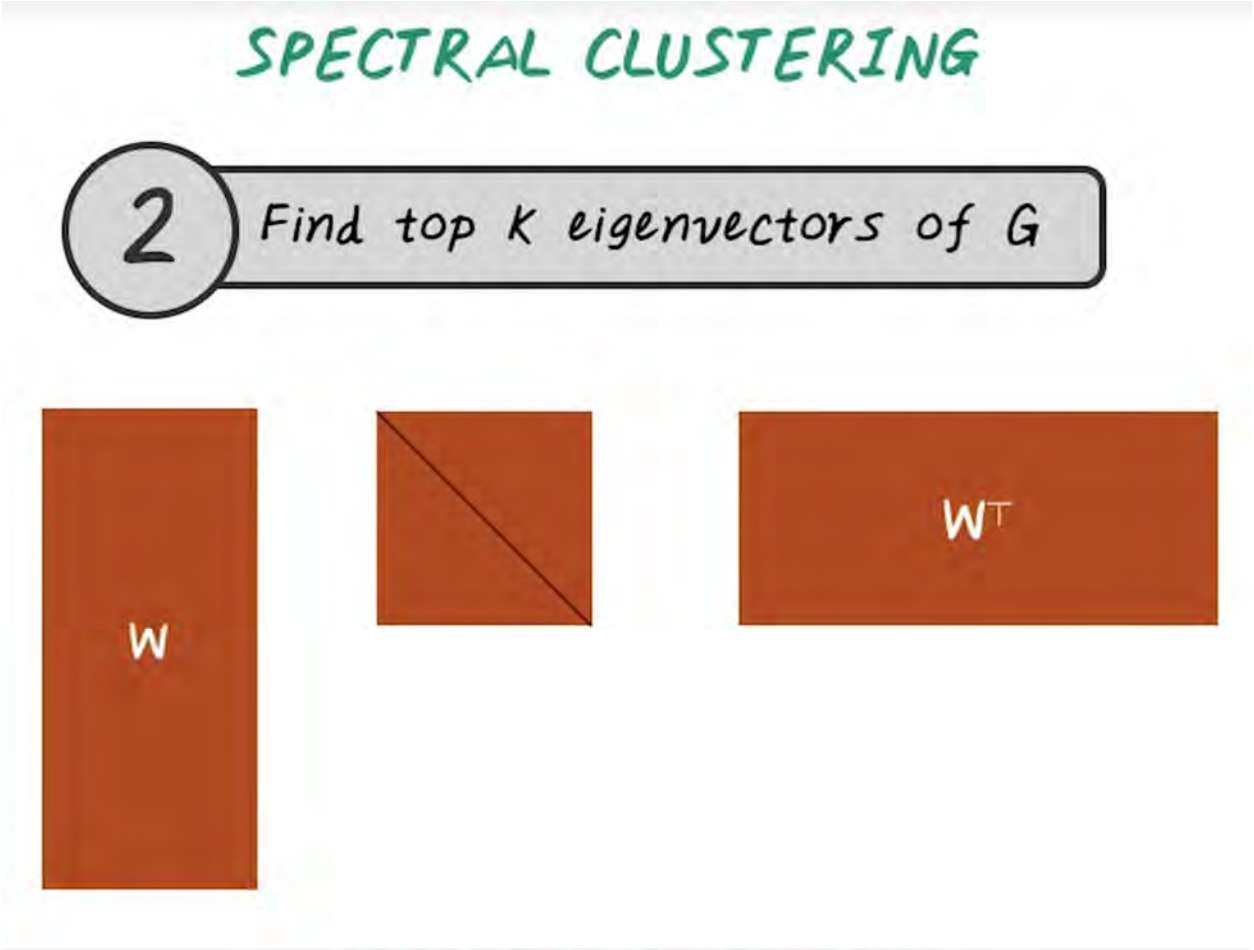
1 Construct the graph

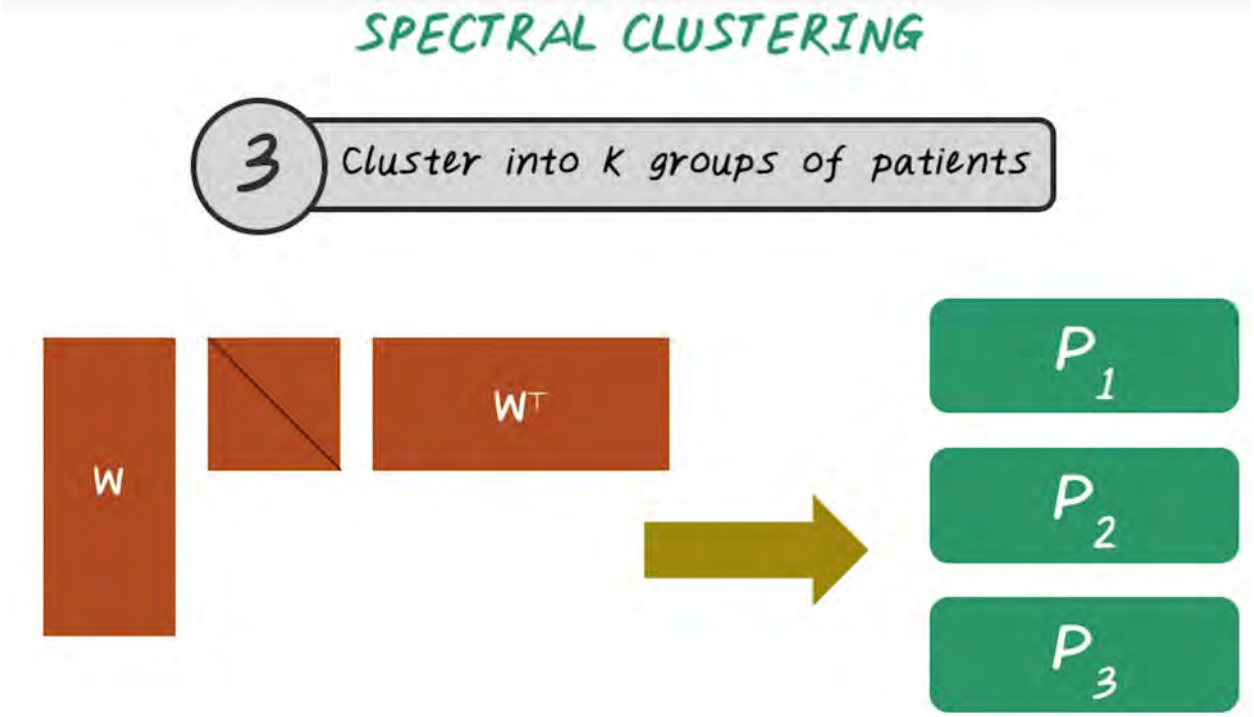


Similarity Graph



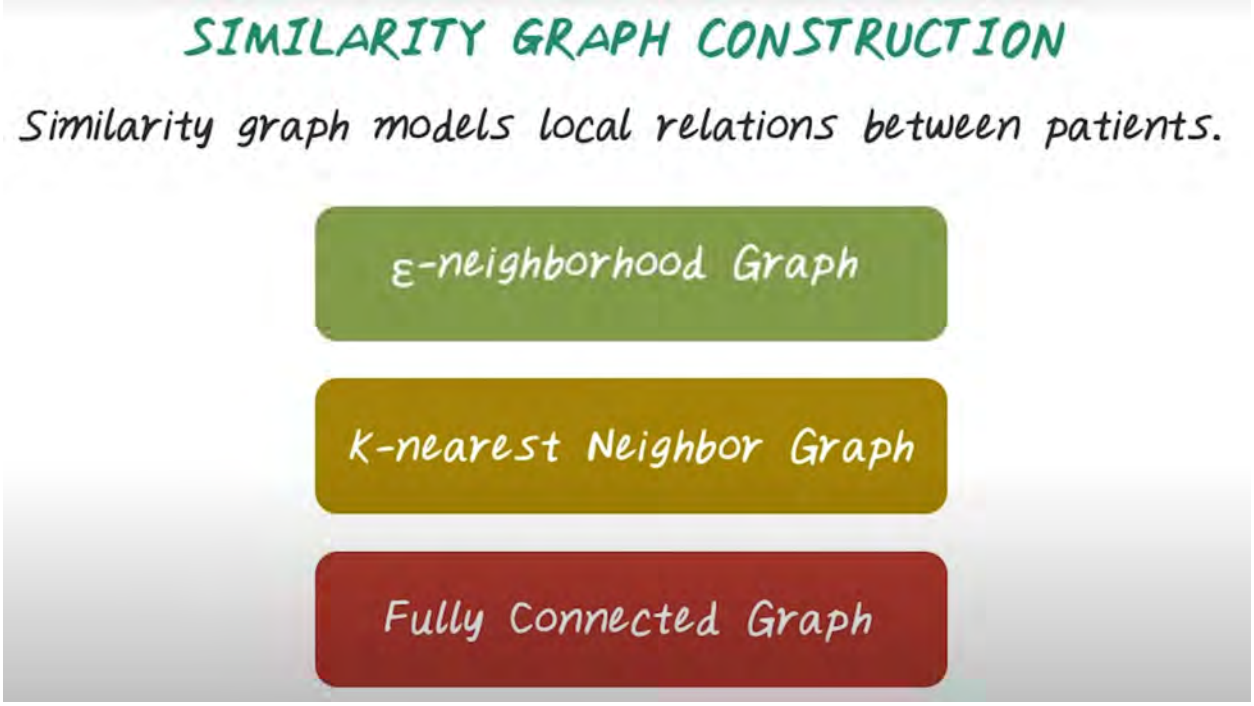
Matrix Representation





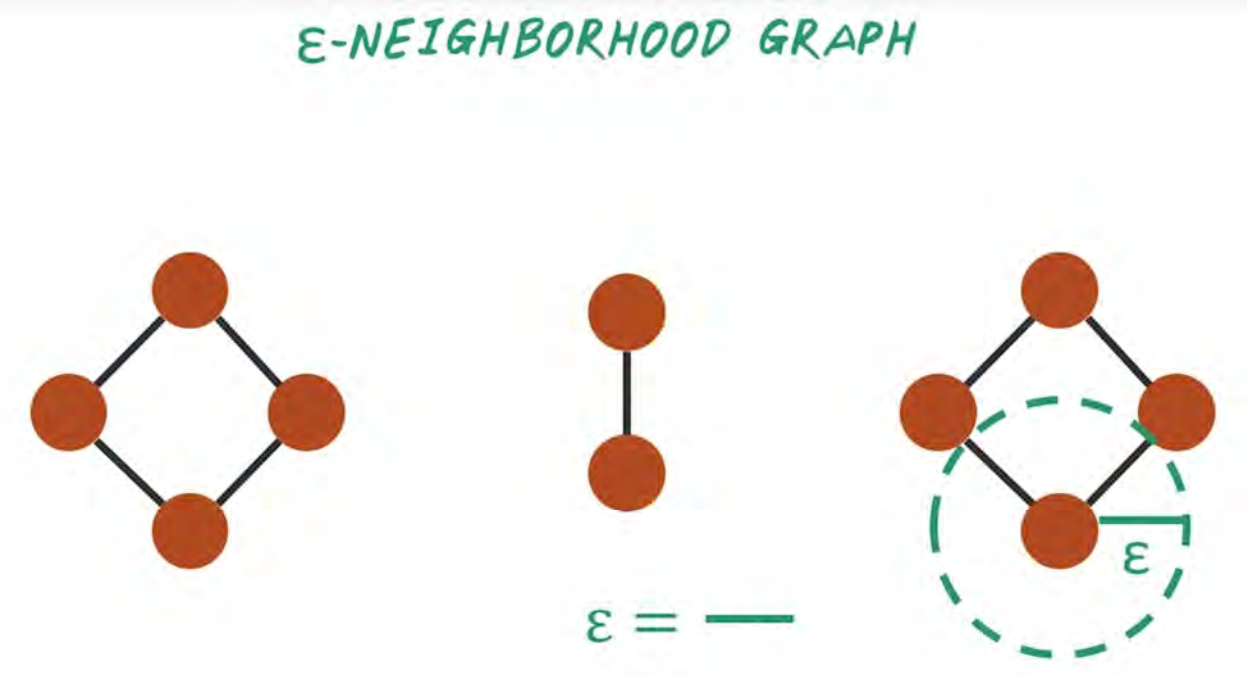
7. [7. Similarity Graph Construction]

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.



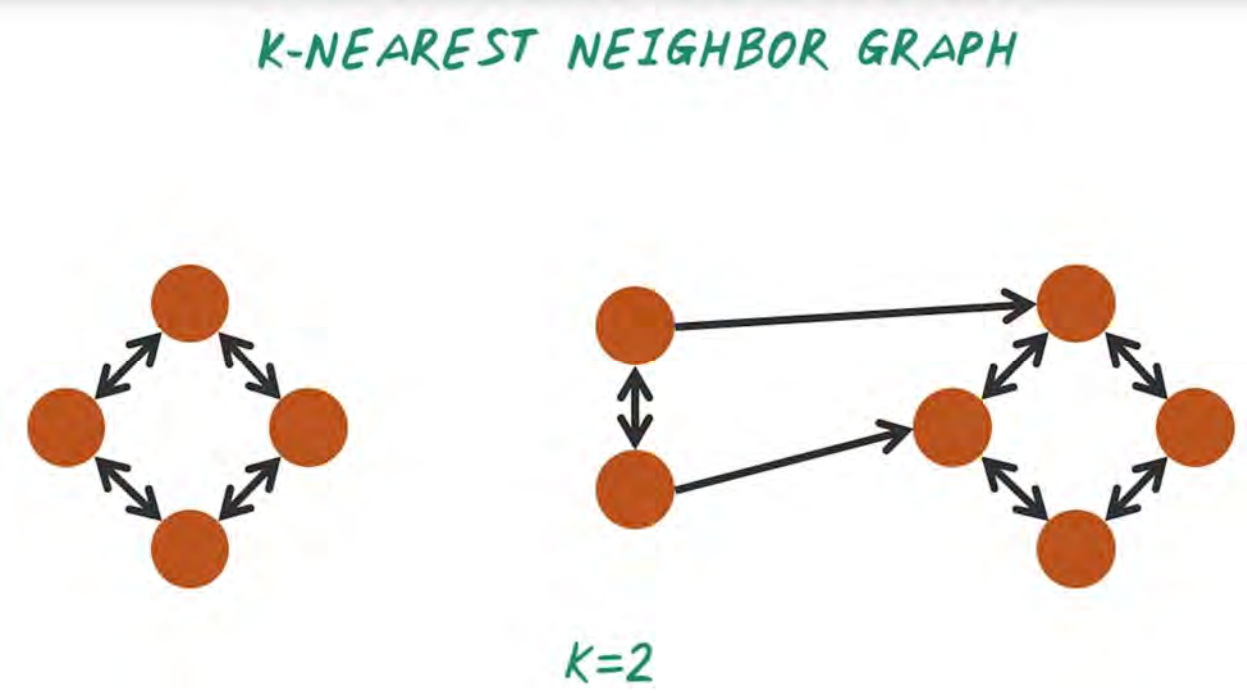
8. 8. E Neighborhood Graph

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.



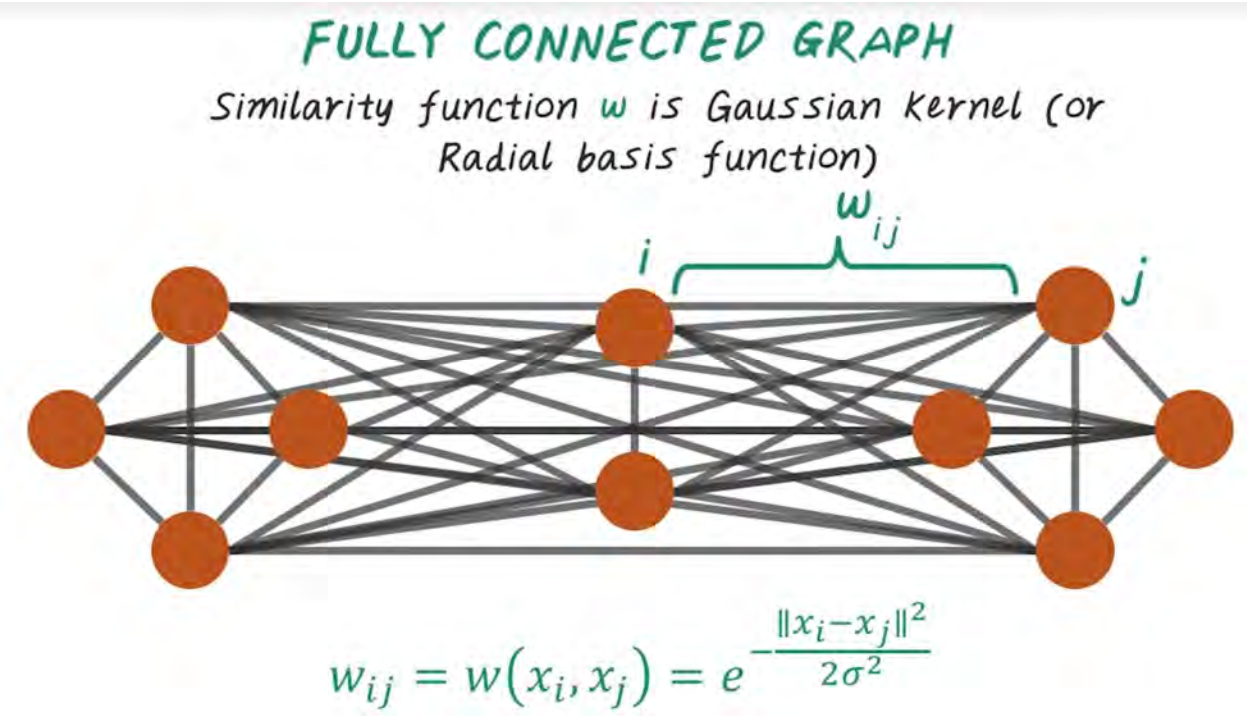
9. [9. K Nearest Neighbor Graph]

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.



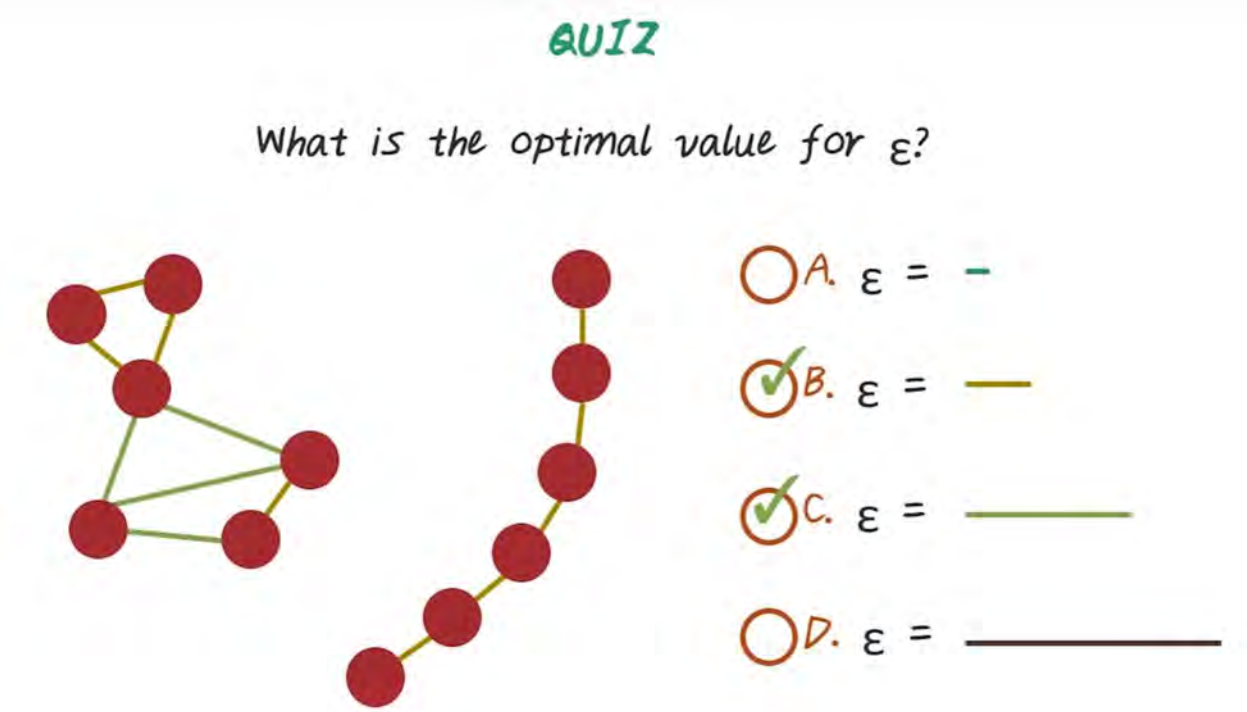
10. [10. Fully Connected Graph]

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.



11. 11. E Neighborhood Graph Quiz

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?



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

12. 12. Spectral Clustering Algorithm

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

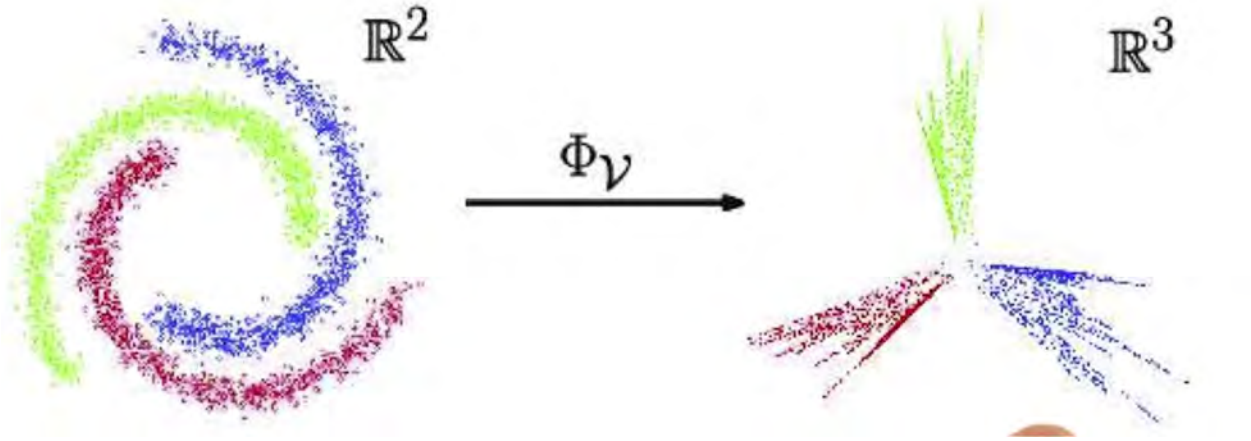
Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.



So far, we explained the high level ideas of spectral clusterings. Depending on how you implement each steps, there are many different variations. If you want to learn more about different variation of spectral clusterings, please refer to this tutorial, and to learn about all those different variations. For example, we can have this very simple unnormalized spectral clusterings, 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 clusterings published in 2000 and another different normalized spectral clusterings 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.

Dimensionality Reduction

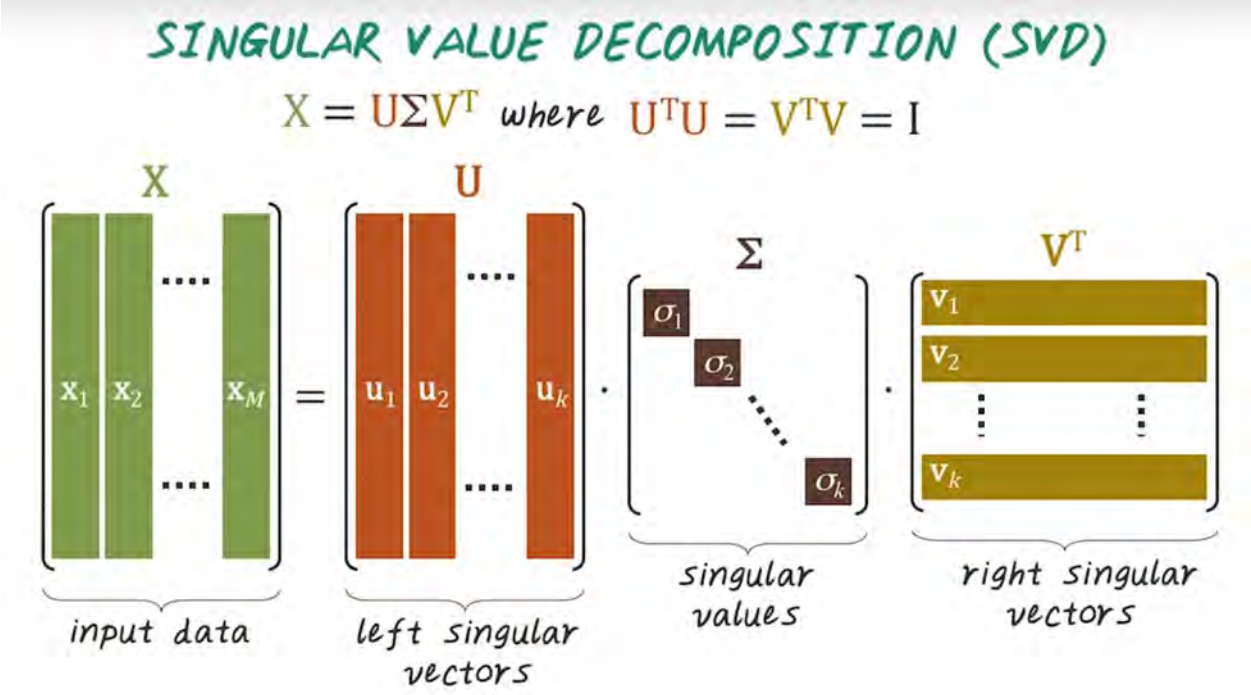
1. 1. Introduction to Dimensionality Reduction

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.

2. 2. Dimensionality Reduction

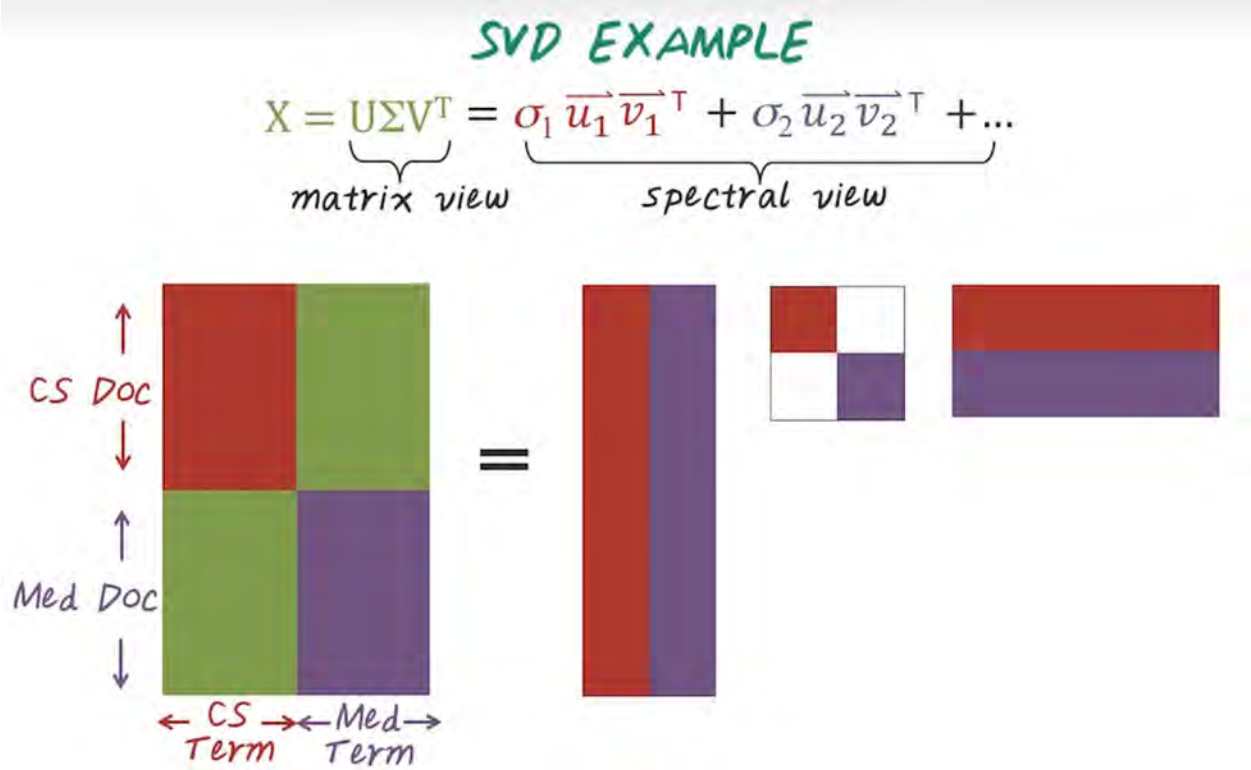
First, a recap of clustering algorithms. In previous lectures, we talked about hard clusterings, such as k-means, hierarchical clustering. We talked about soft clustering algorithms such as Gaussian mixture model. We also talked about scalable clustering algorithms. Such as, mini batch k-means and DBScan. 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.

3. 3. Singular Value Decomposition



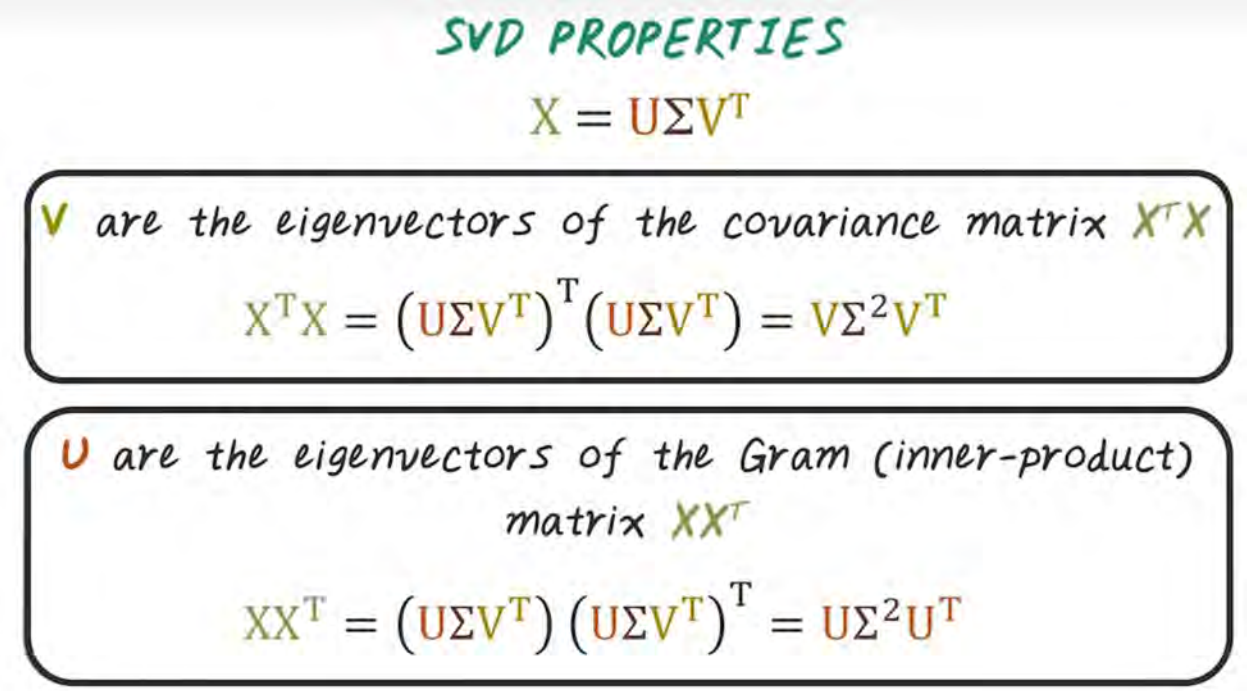
First, let's talk about singular value decomposition, or SVD. SVD vectorized the input matrix X as product of three matrices. U, sigma, V transpose. Where the U and V matrices are which means U transpose times U equals to V transposed times V equal to identity matrix. Visually, given a large matrix X. 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.

4. 4. SVD Example



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 traspose. 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 [INAUDIBLE] of this huge matrix Into this two dimensional space. One for CS document, one for medical document.

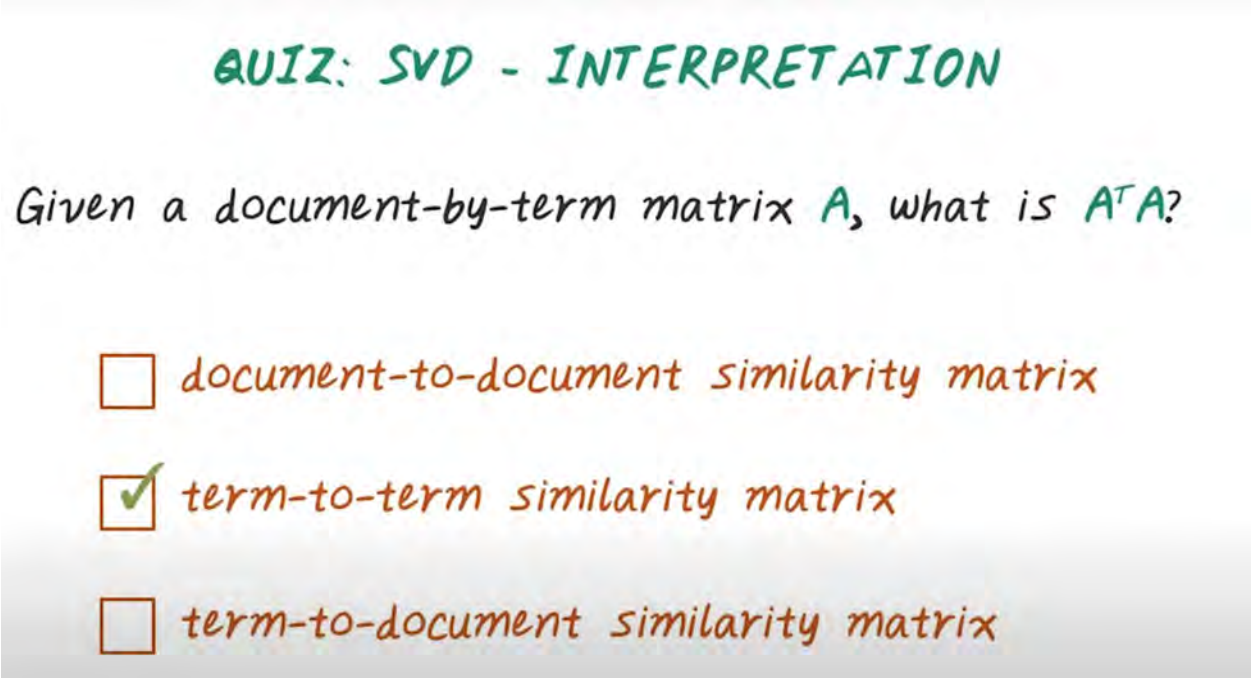
5. 5. SVD Properties



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.

6. 6. Quiz SVD Interpretation

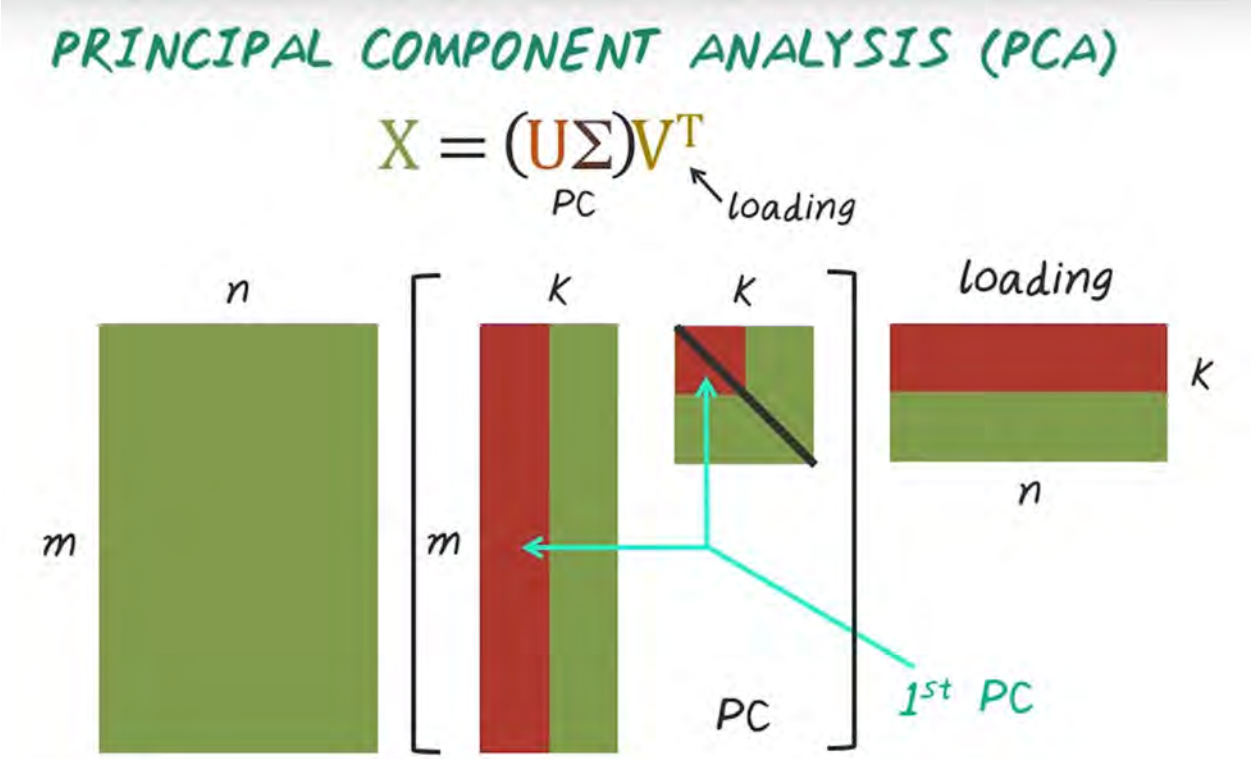
Let's do a quiz to understand SVD better. Given a document-by-term matrix like the one we have shown you in the previous slide, what is A transpose A? Is it a document-to-document similarity matrix? Or is it term-to-term similarity matrix? Or is term-to-document similarity matrix?



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.

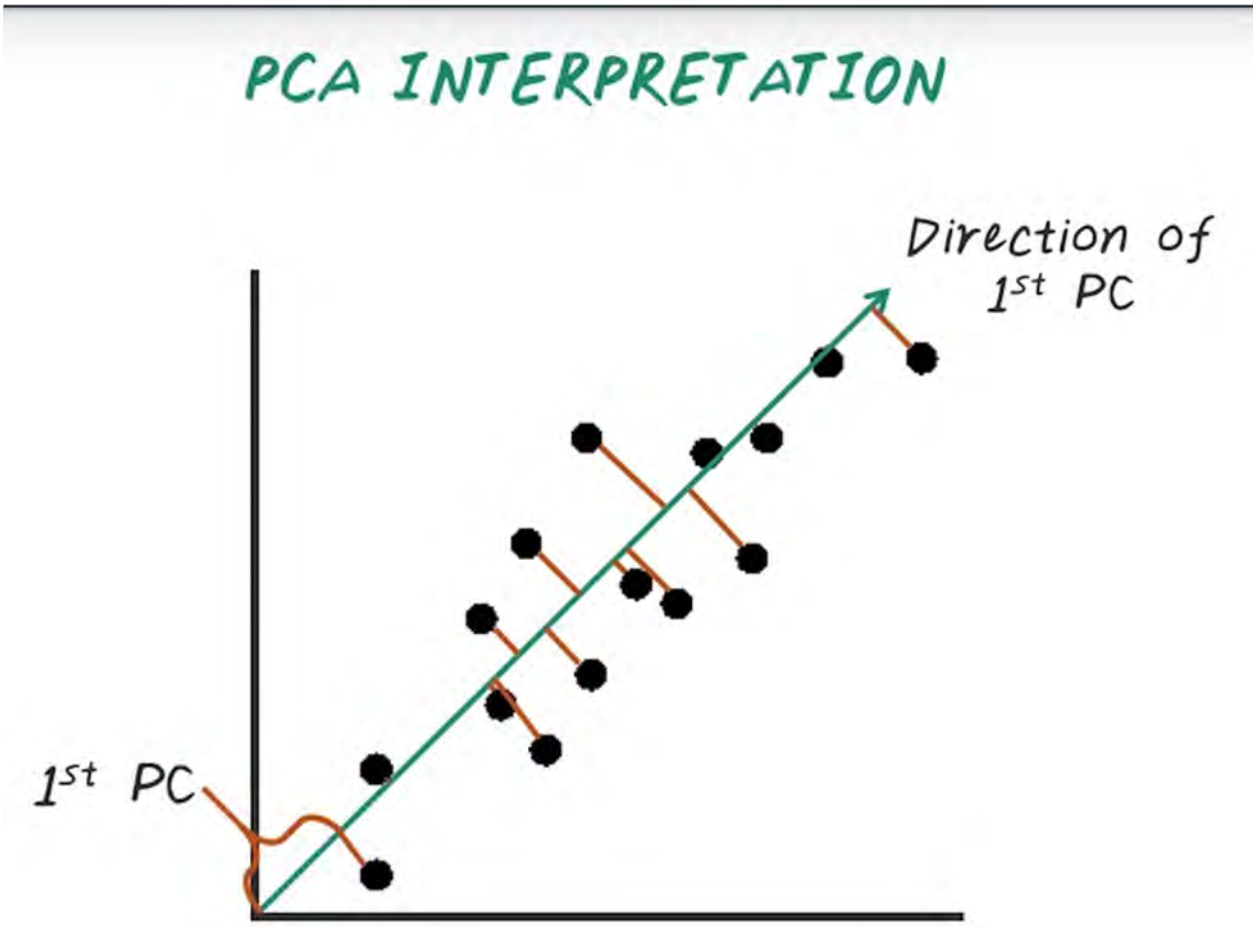
7. 7. Principal Component Analysis

Next lets 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.



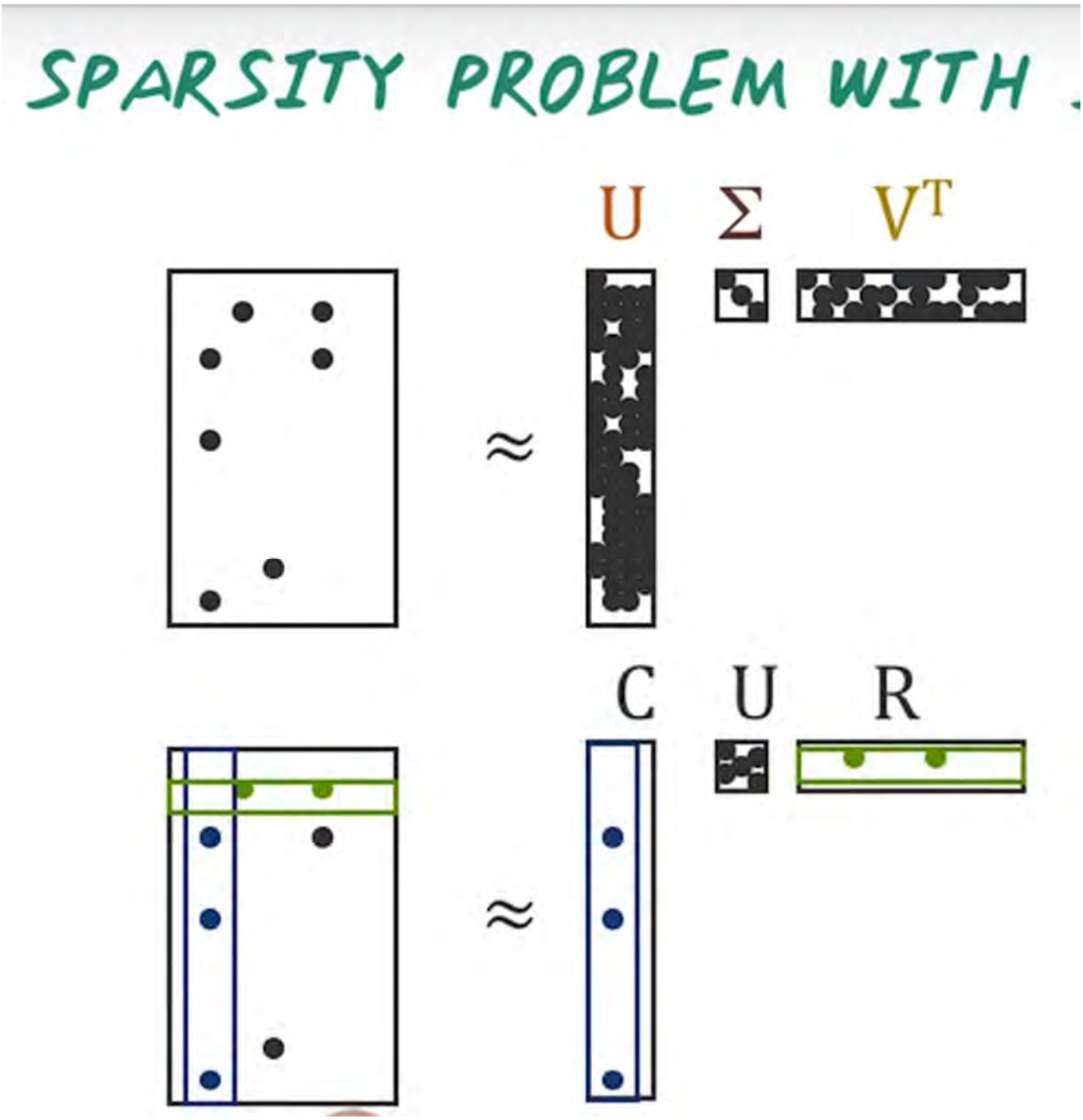
8. [8. PCA Interpretation]

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 represent 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.



9. 9. Sparsity Problem with SVD

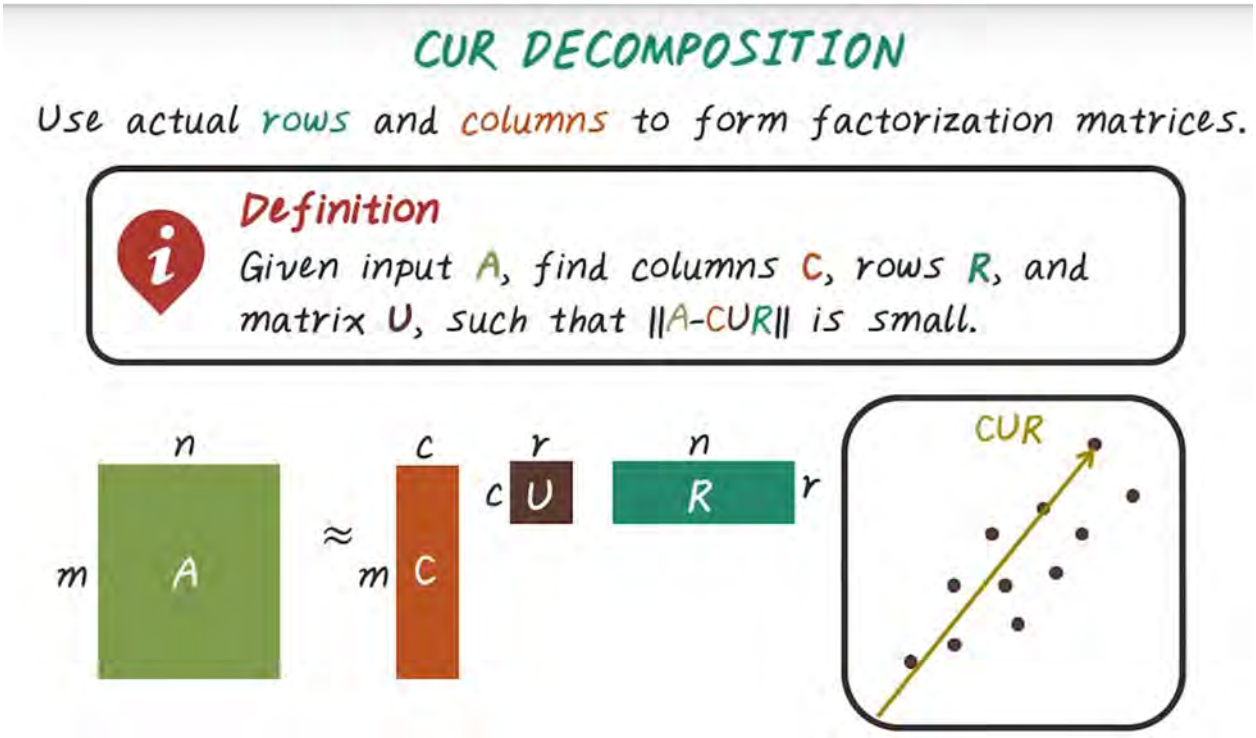
So what's the problem with SVD or PCA? It has a sparsity problem. For example most of the input dataset for analytics are often sparse, meaning that only a small number of elements in the large matrix are non zeros, majority of these elements are zeros. For example, every row represent a patient, every 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.



10. [10. CUR Decomposition]

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.



11. [11. CUR Algorithm]

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.

CUR ALGORITHM

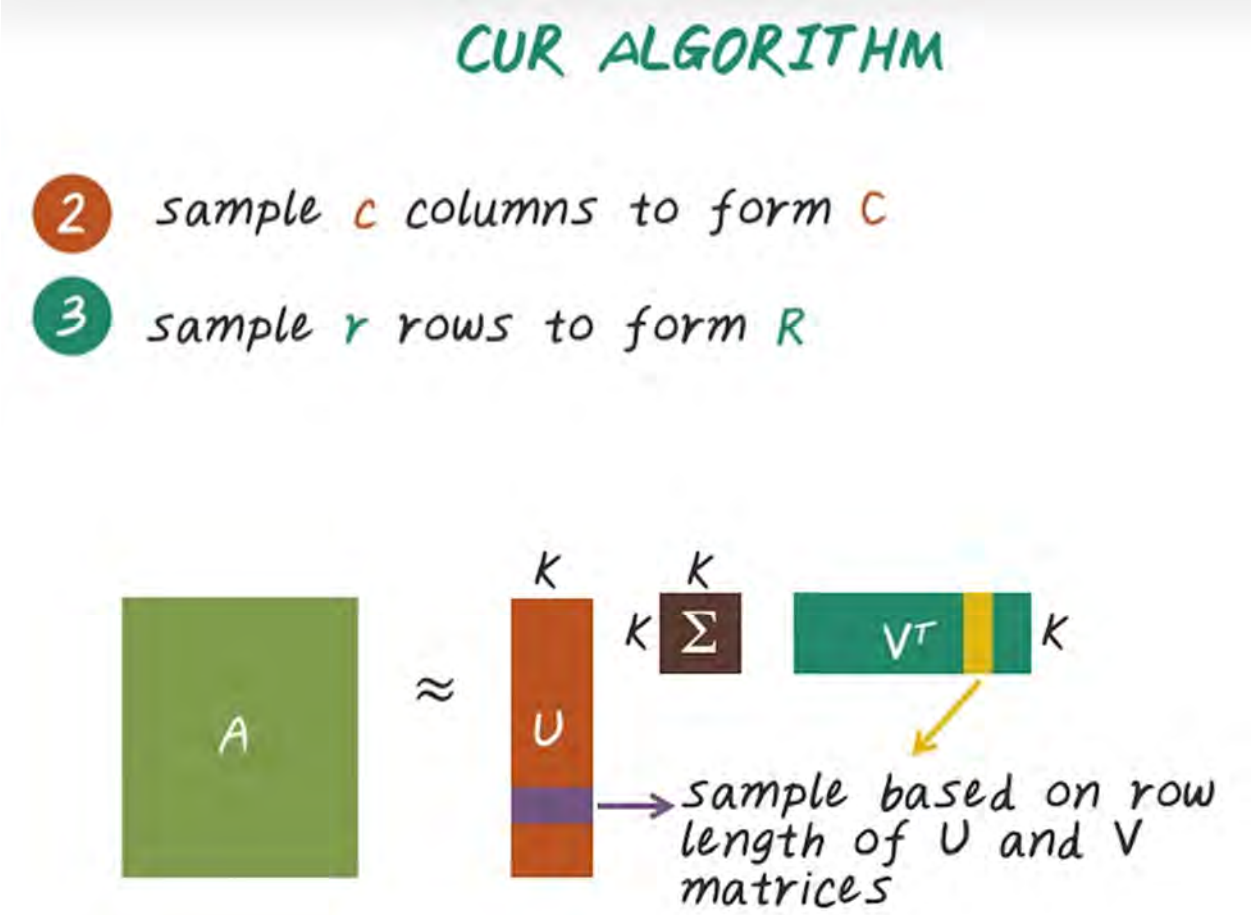
INPUT A, # columns c, # rows r

OUTPUT C, U, R

1. $A = U\Sigma V^T$ // find rank-k approx. with SVD
2. sample c columns to form C
3. sample r rows to form R
4. $U = C^{+}AR^{+}$

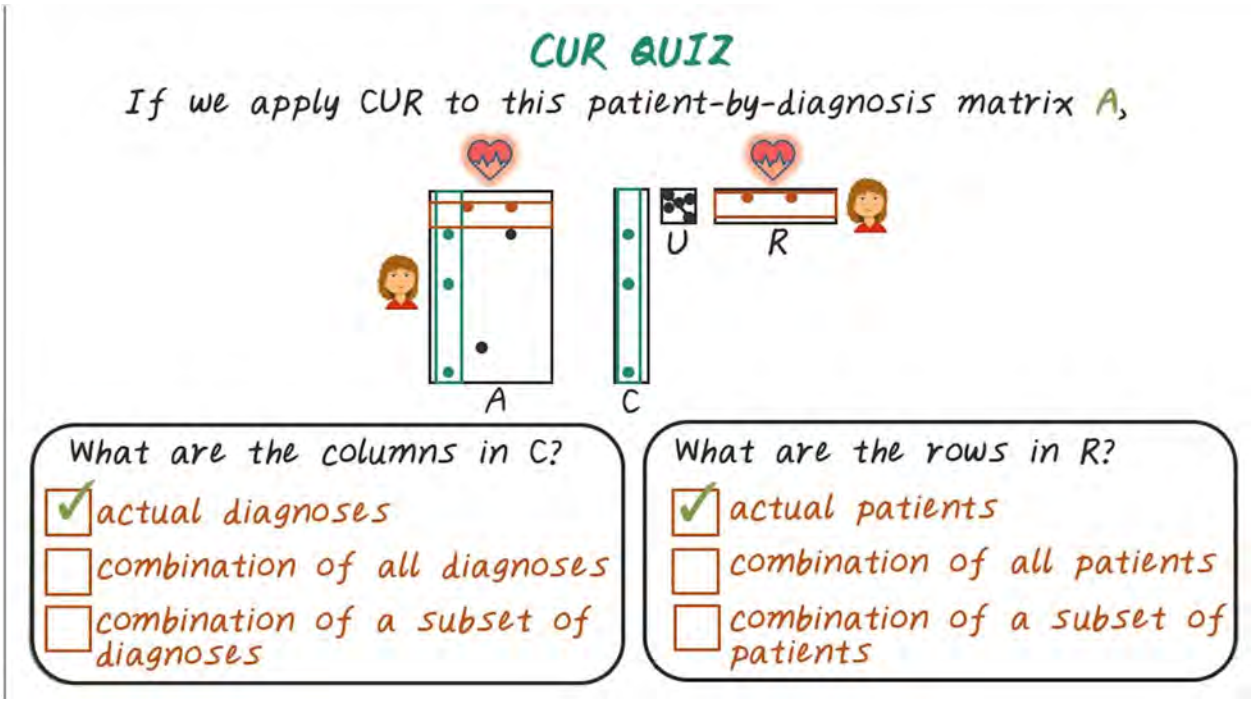
pseudo-inverse

$X = U\Sigma V^T \rightarrow X^{+} = V\Sigma^{-1}U^T$



12. [12. CUR Quiz]

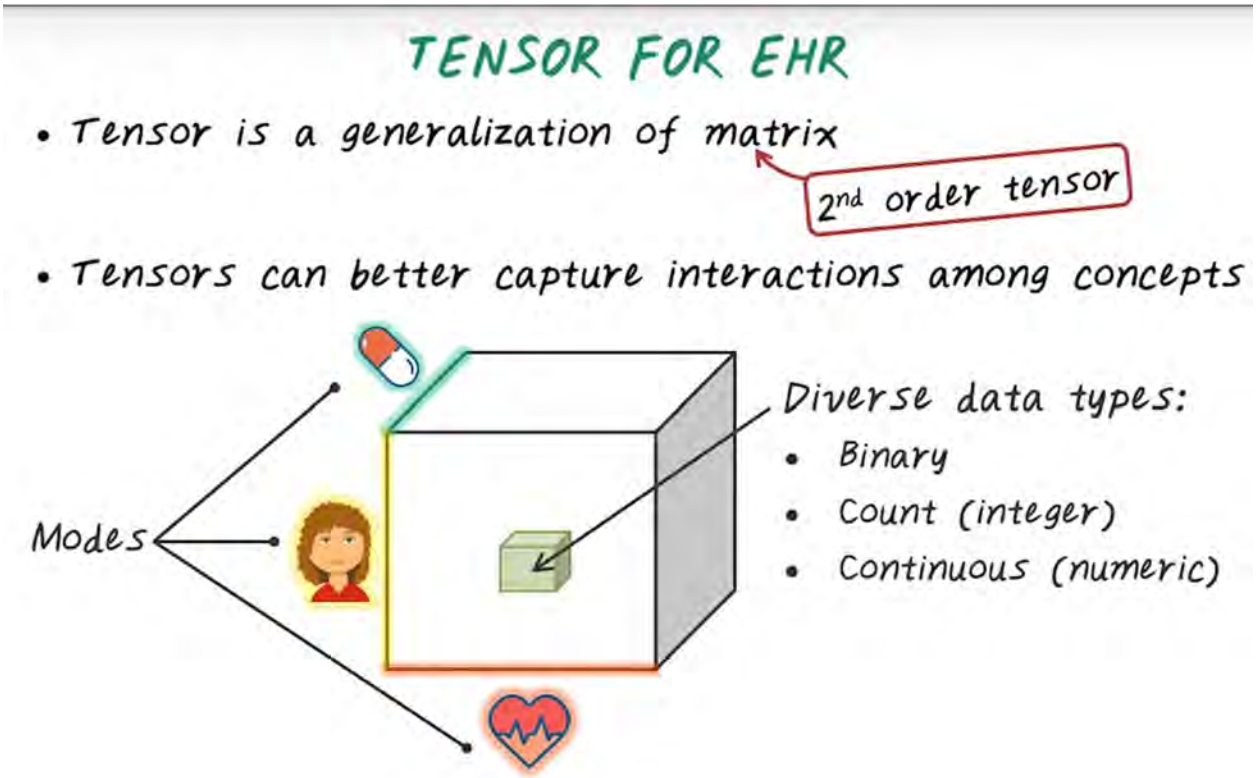
Here's a quiz on CUR decomposition. If we apply CUR on this patient-by-diagnosis matrix A, to get the CUR decomposition from this patient-by-diagnosis matrix, then what are the columns in C? Are they actual diagnoses, combination of all diagnoses, or combination of a subset of diagnoses? Similarly, what are the rows in R? Are they actual patients, combination of all patients, or a combination of a subset of patients?



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.

13. 13. Tensor for EHR

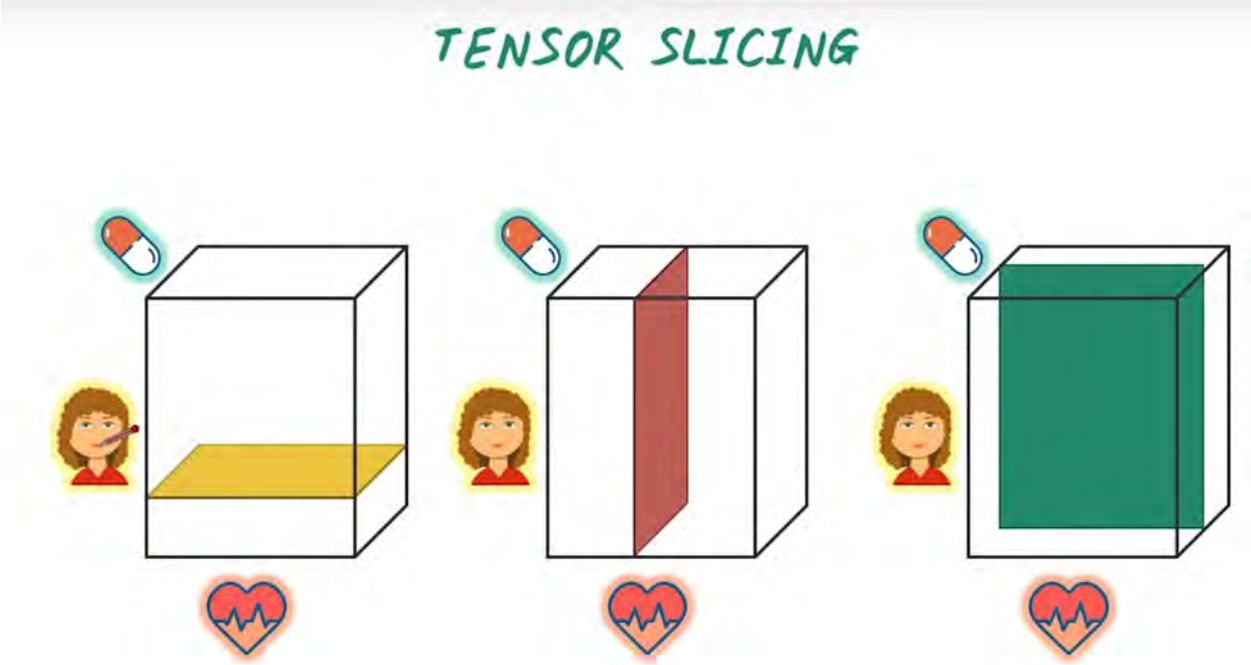
Next, we want to talk about tensor factorization as another dimensionality reduction method, but first, what is a tensor? Tensor is a generalization of a matrix. A matrix is actually a special case of tensor of the second order, so we call matrix a second order tensor. Tensor can better capture interactions 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.



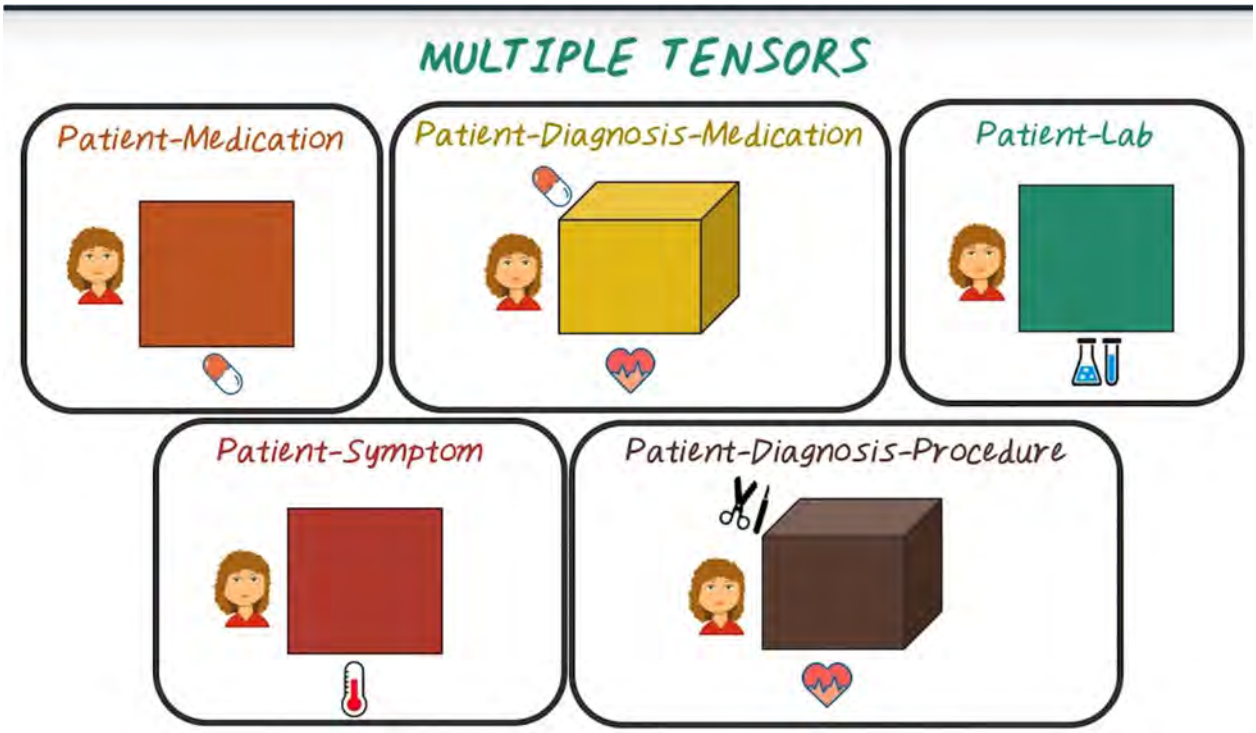
Now let's talk about some basic operation on tensor. Tensor slicing is an operation to extract a subtensor, in this case, a matrix. So the idea is to set all the mode except two modes the same. By doing so, we'll get a matrix. For example, given this third alter tensor, patient, diagnosis, medications, 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.

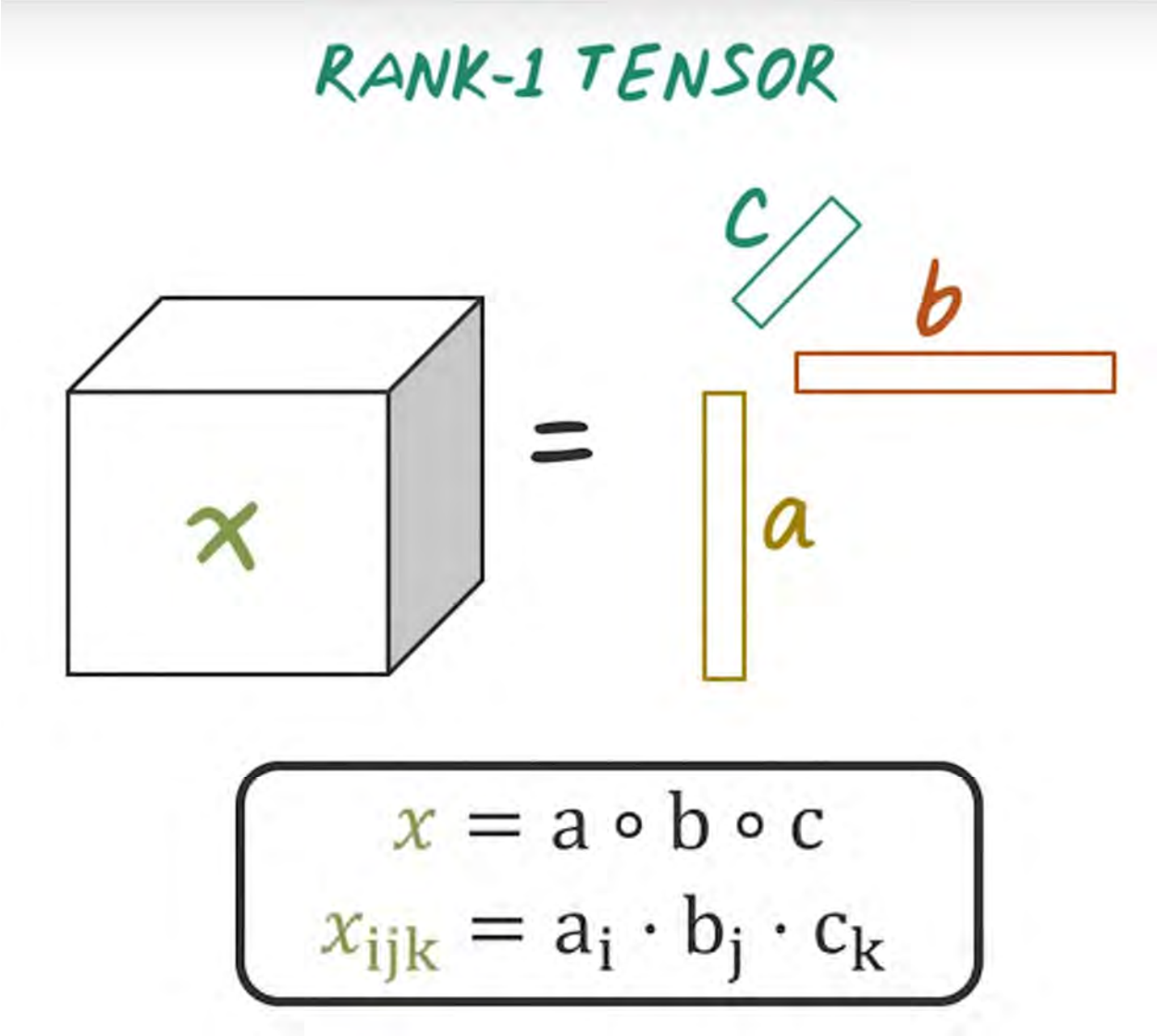
14. 14. Tensor Slicing

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 alter 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.



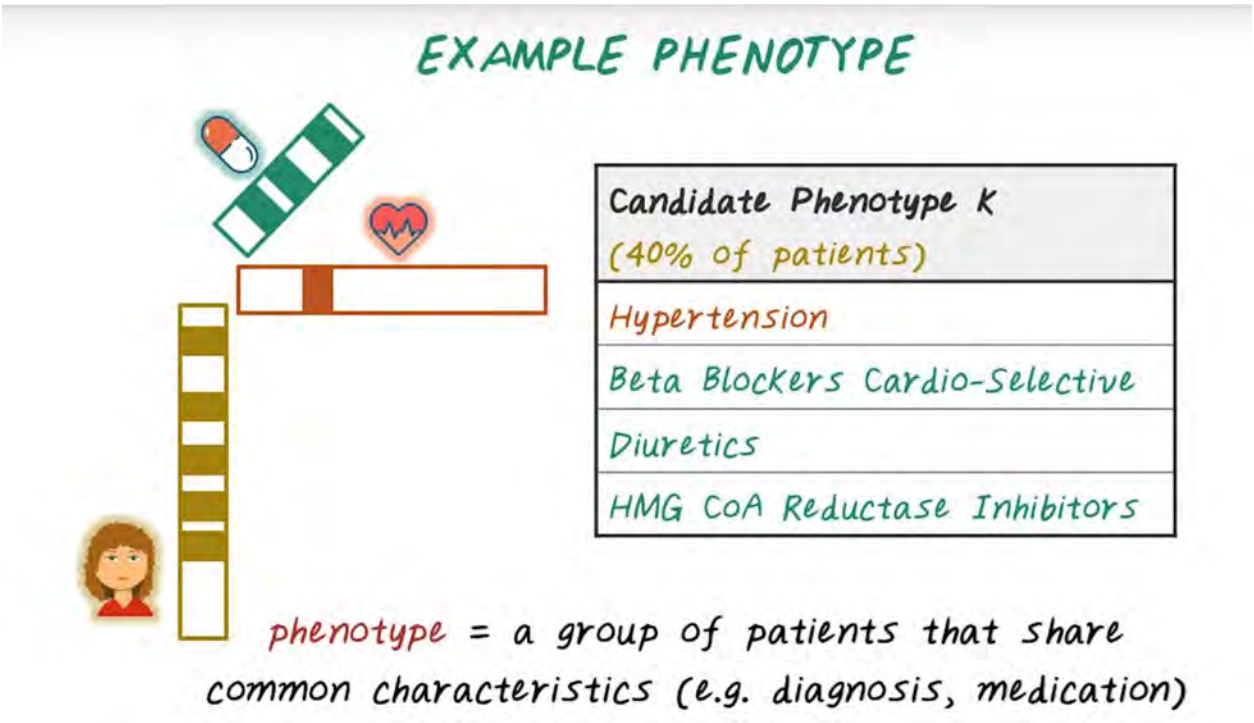
15. [15. Rank 1 Tensor]





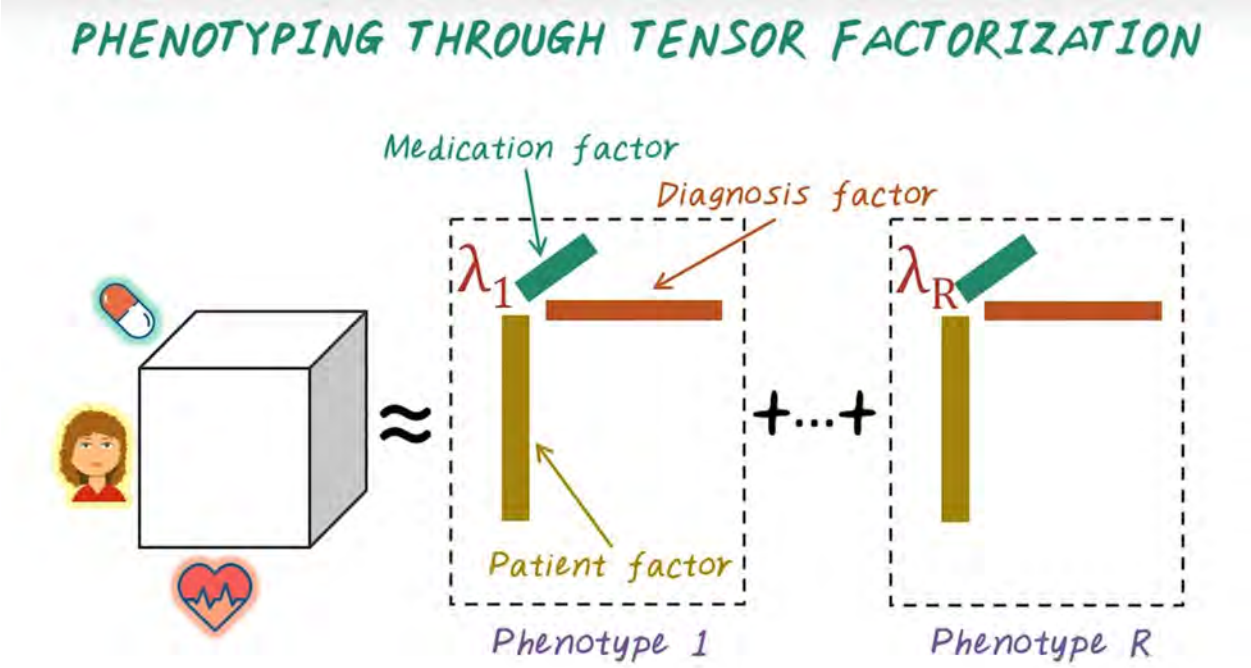
16. 16. Example Phenotype

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.



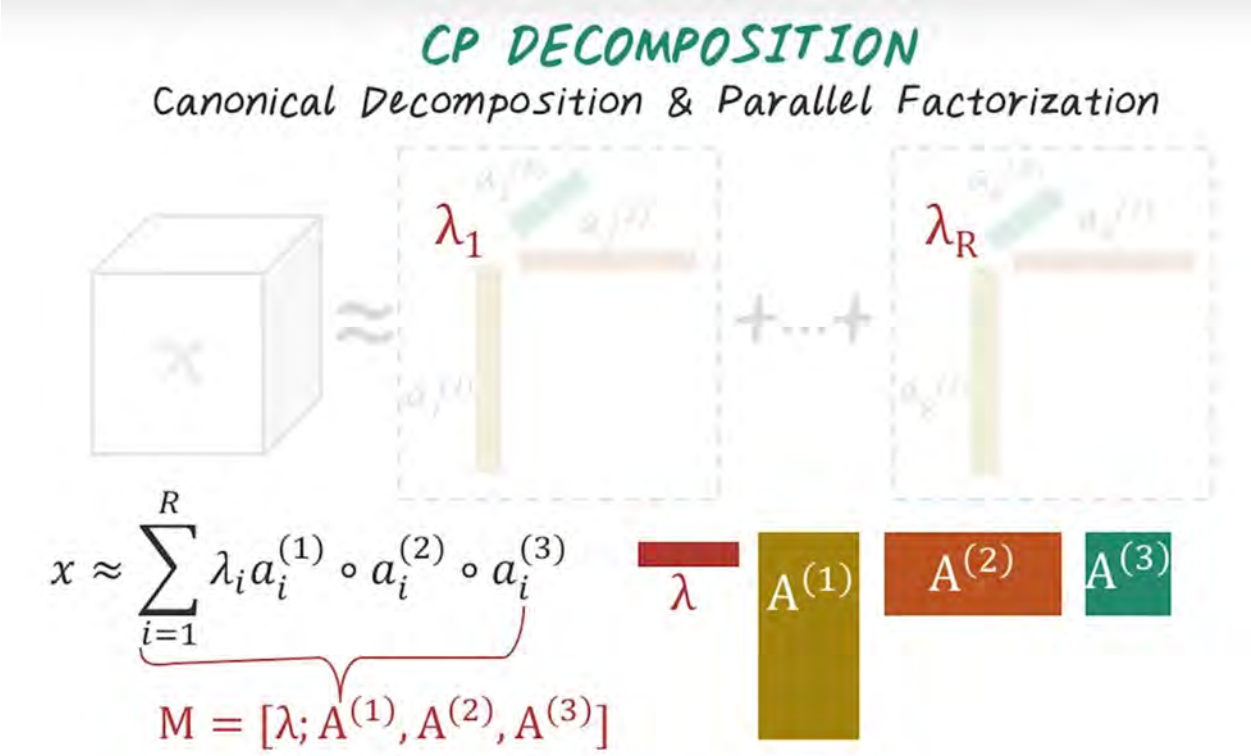
17. 17. Phenotyping Through Tensor Factorization

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.



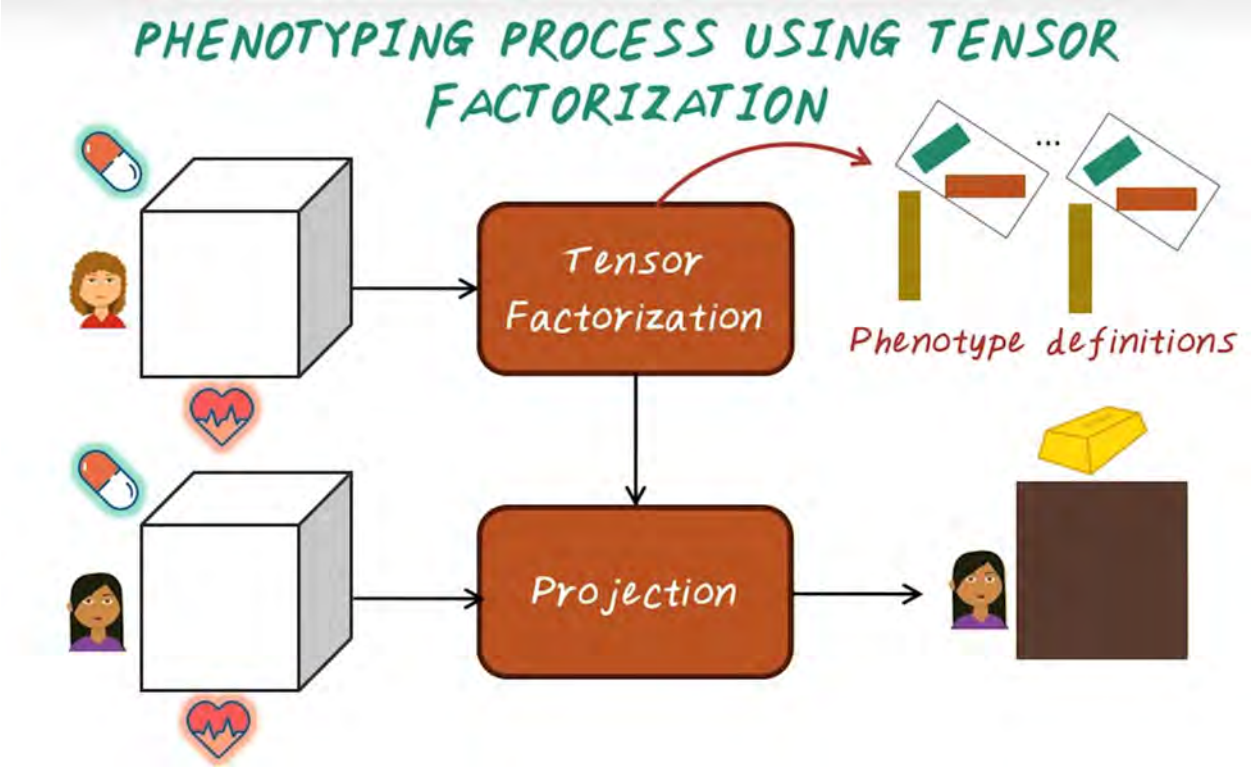
18. 18. CP Decomposition

Simply the composition factorized in [INAUDIBLE] 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.



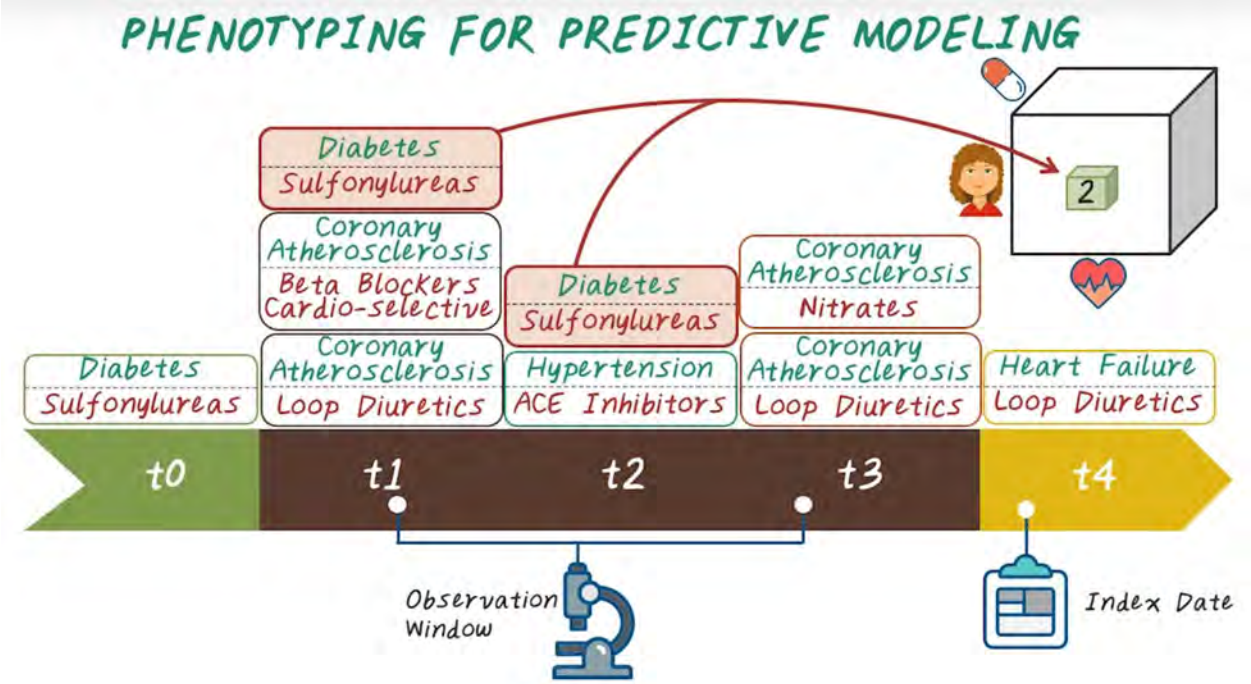
19. [19. Phenotyping Process Using Tensor Factorization]

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 tells us, for a given patient, which phenotypes she has.



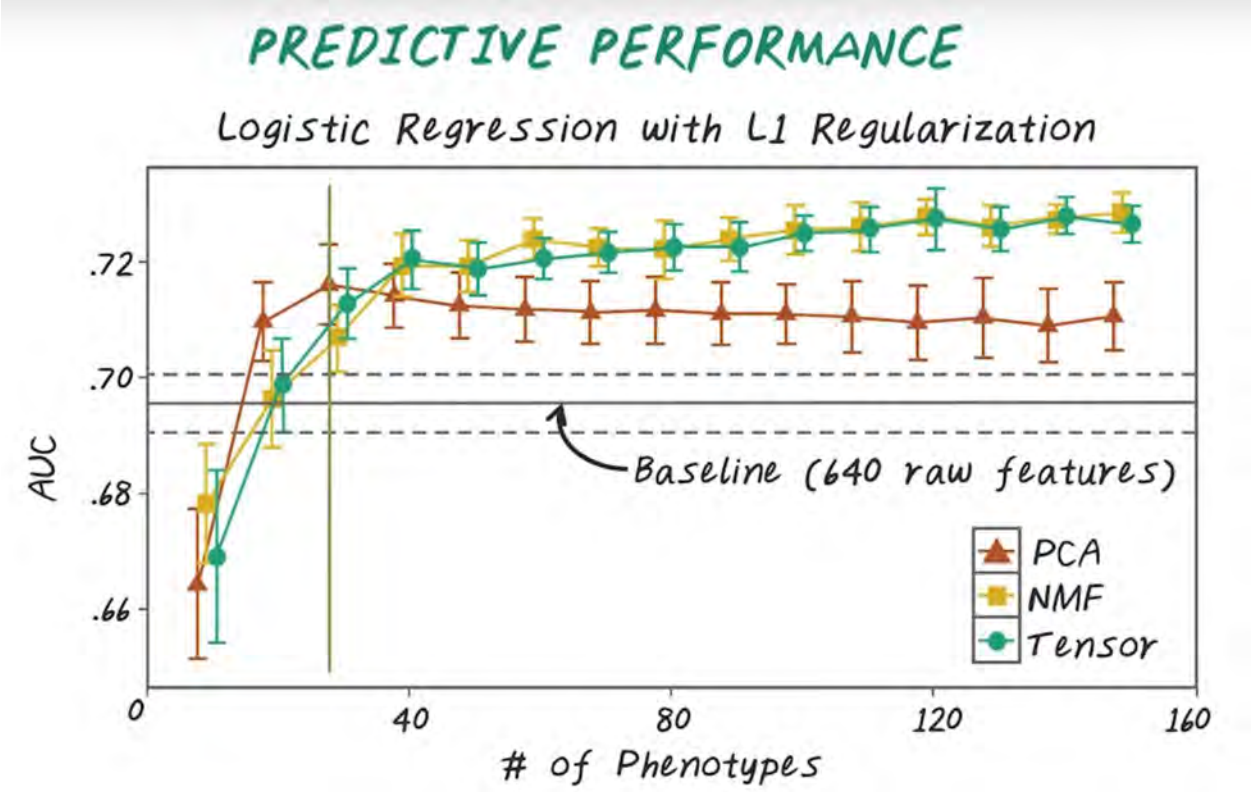
20. 20. Phenotyping for Predictive Modeling

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 tensure 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.



21. [21. Predictive Performance]

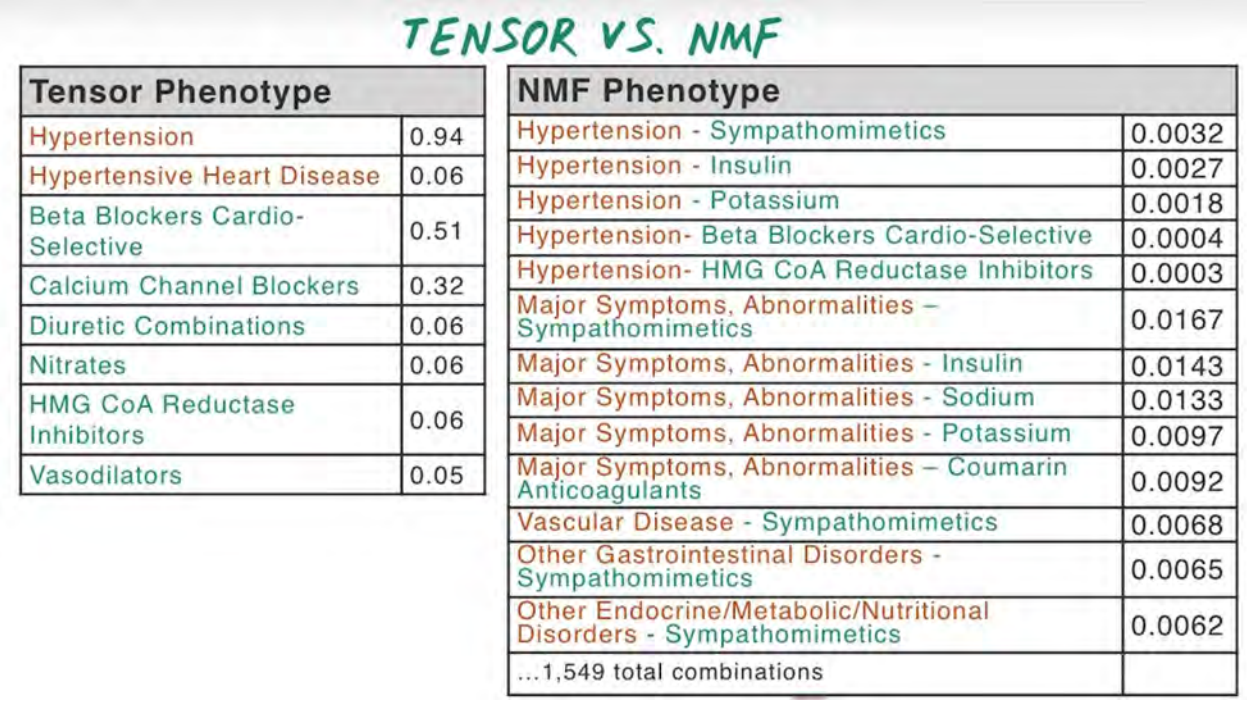
So here's a predictive performance. Here, we're trying to compare different dimensionality reduction method. And using those low dimensional representation as features. To predict whether the patient will have heart failure or not. In this case, the baseline performance is indicated by this line. Is 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.



22. 22. Major Disease Phenotypes

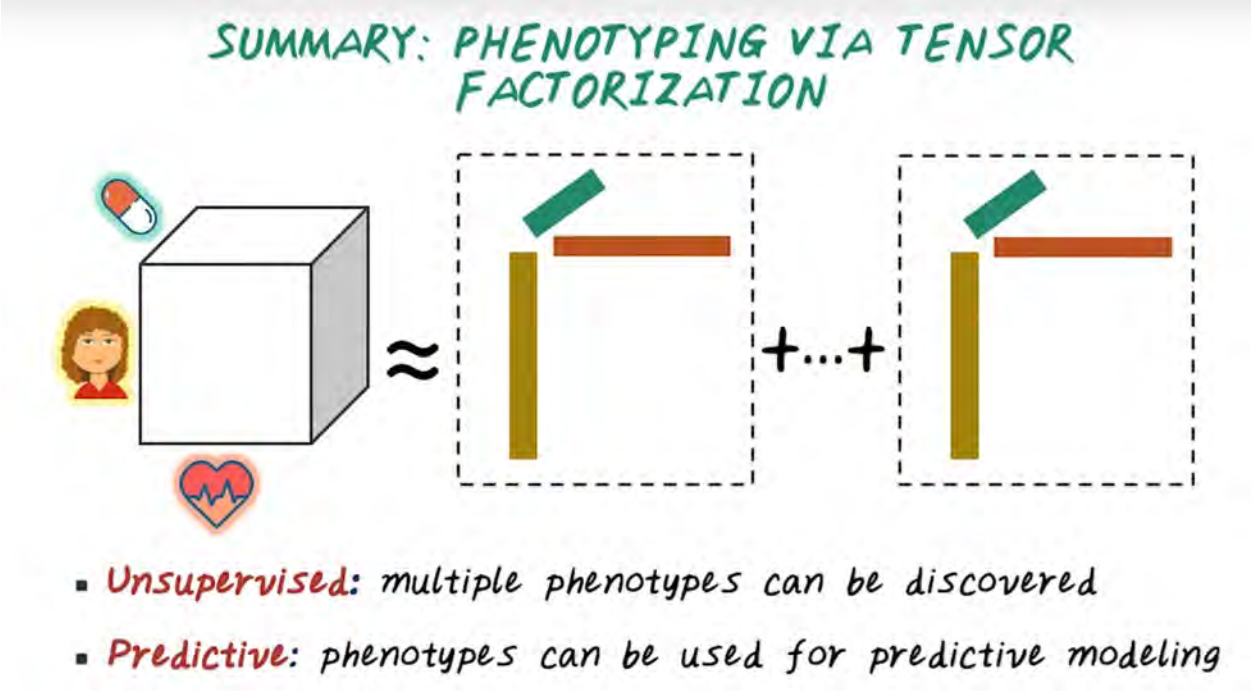
Next, I want to show you intuitively how those phenotypes look like. Using tensor factorization, we're able to extract different disease phenotypes, some corresponding to major disease such as diabetes without complications, so we call this uncomplicated diabetes phenotypes that cover 17.6% of of the 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.

23. 23. Tensor vs NMF



So you may wonder how does tensor factorization compare to this non negative 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.

24. [24. Summary Phenoty]



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.

Patient Similarity

1. 1. Introduction to Patient Similarity

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.

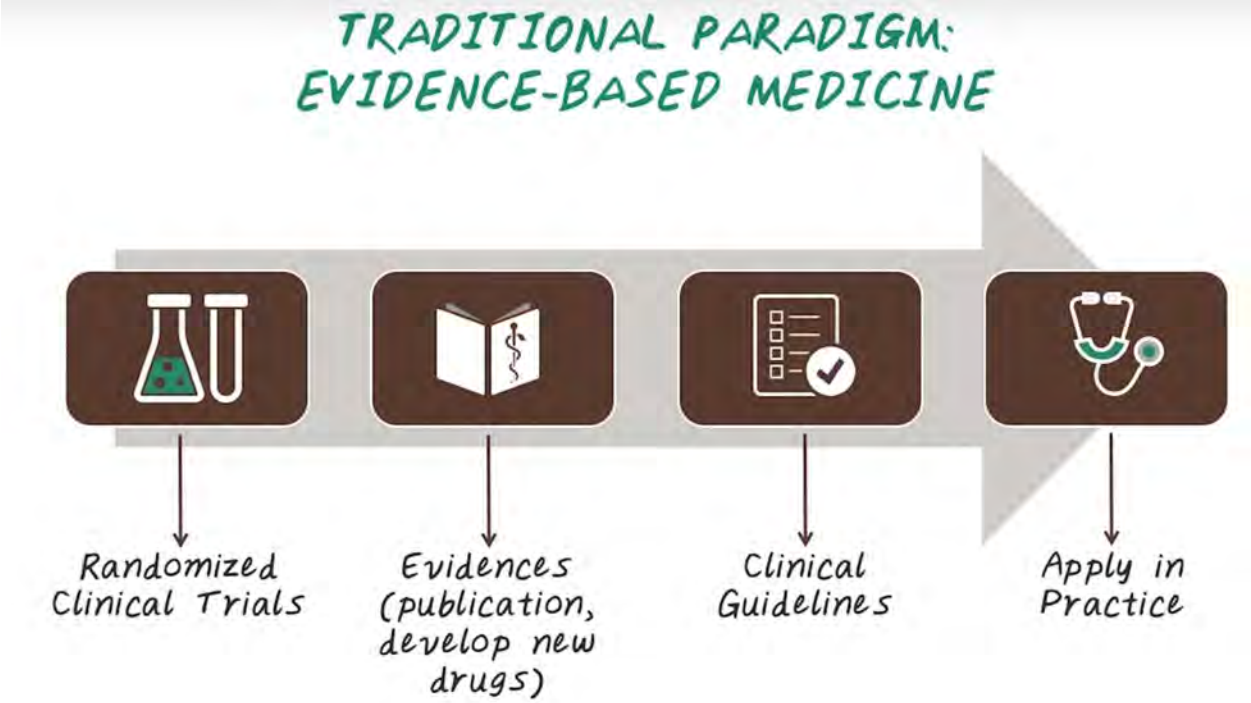
2. 2. Motivation

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.

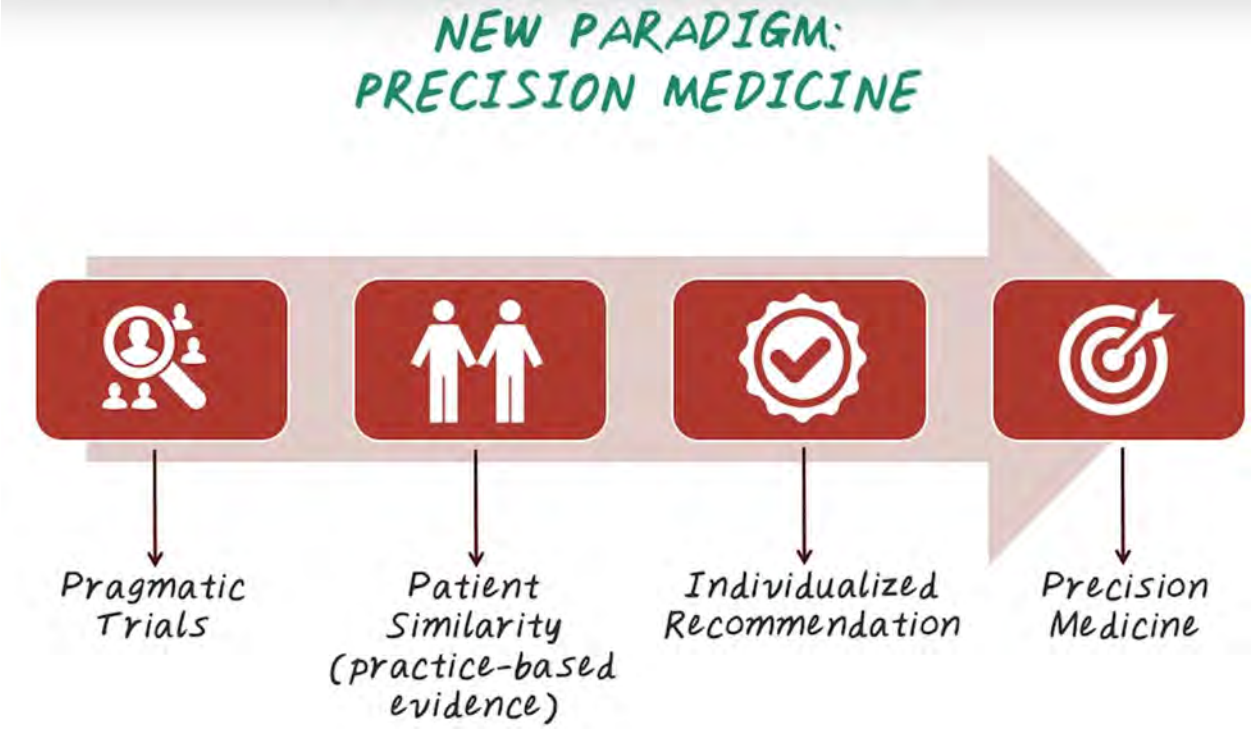
3. 3. Traditional Paradigm

The traditional paradigm is sometime called evidence-based me dicine. 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.



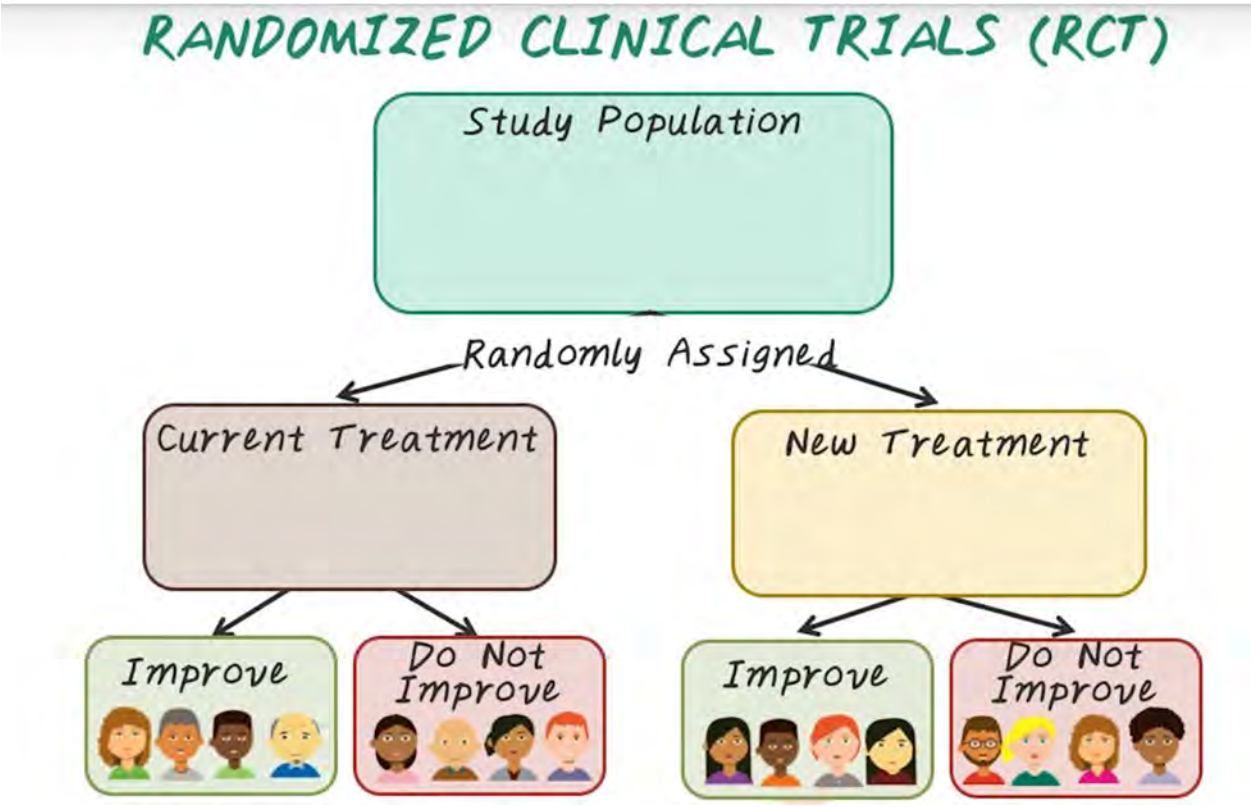
4. 4. New Paradigm

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.



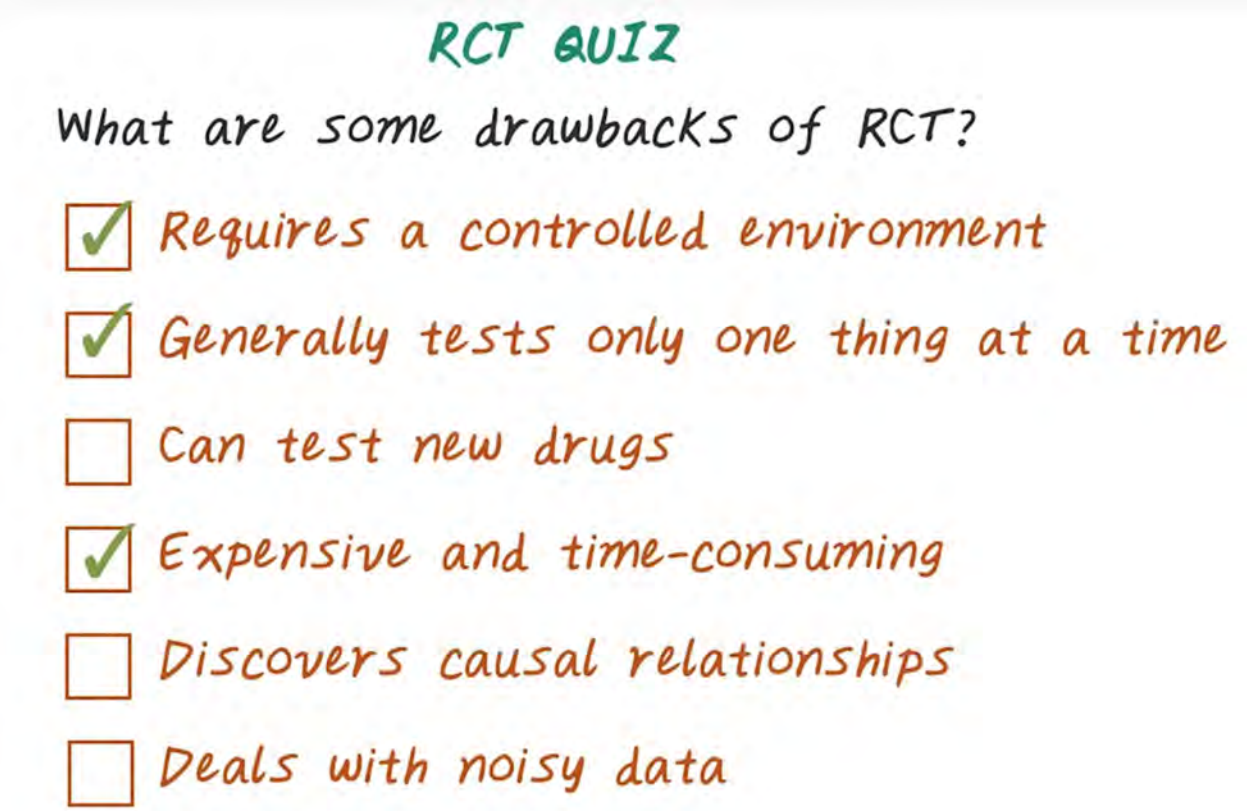
5. 5. Randomized Clinical Trials

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.



6. 6. RCT Quiz

Now lets 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.

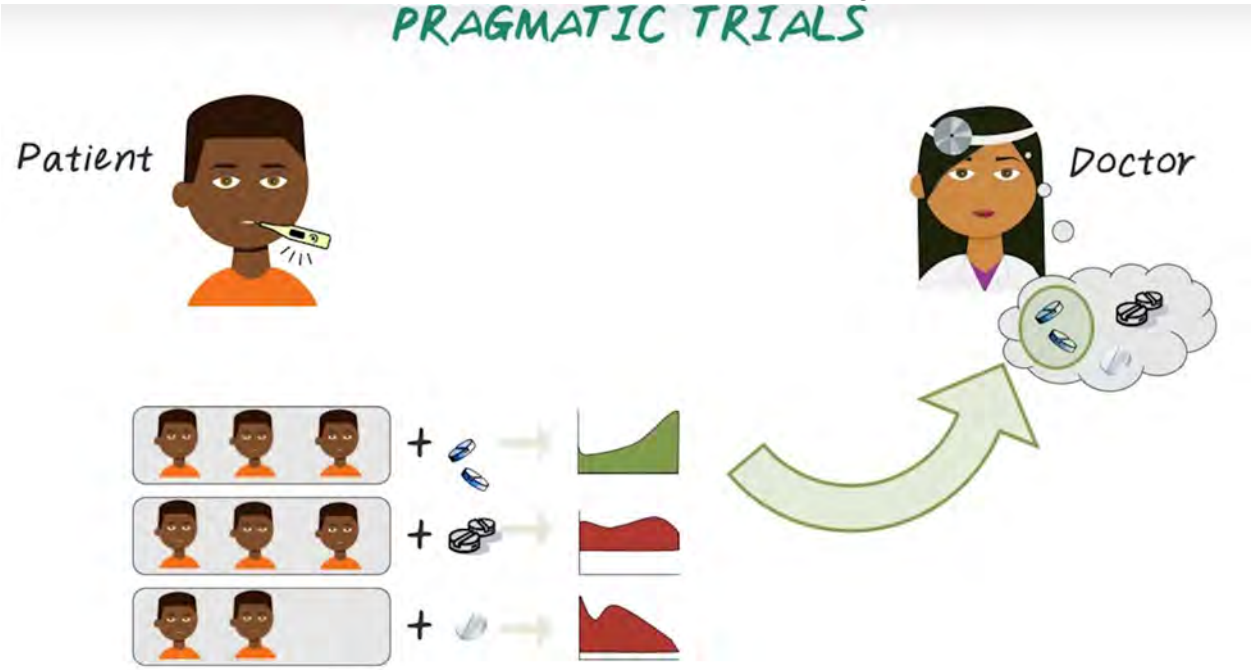


Here's the answers. RCT requires a controlled environment. The studied population in RCT are often times carefully selected with very strict inclusion and exclusion criteria. So the studied population often do not reflect the general population in the real world. Another drawback of RCT is it generally 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

7. 7. Pragmatic Trials

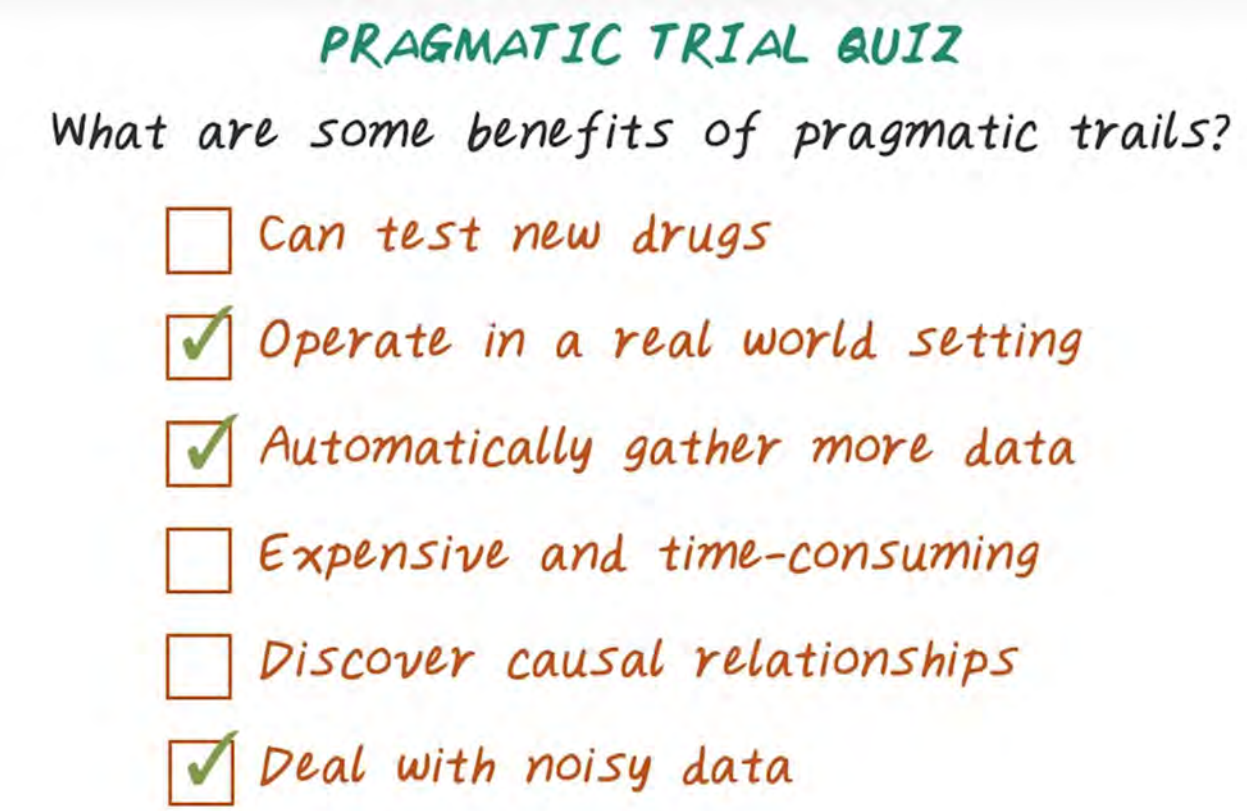
Next let's talk about pragmatic trials. So in traditional RCT, we generally measure the efficacy of a treatment that produces under ideal conditions. Often use carefully designed patient population in a research clinic. Pragmatic trials on the other hand, try to measure the effectiveness of a treatment 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.



8. 8. Pragmatic Trial Quiz

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?

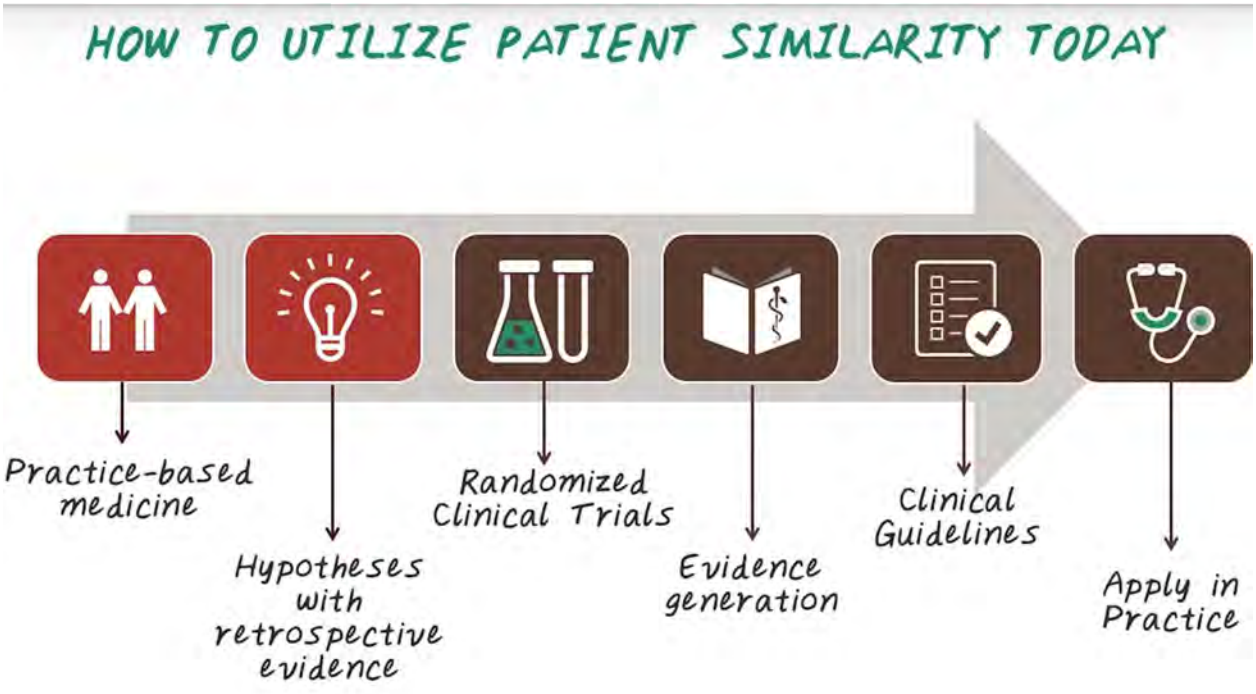


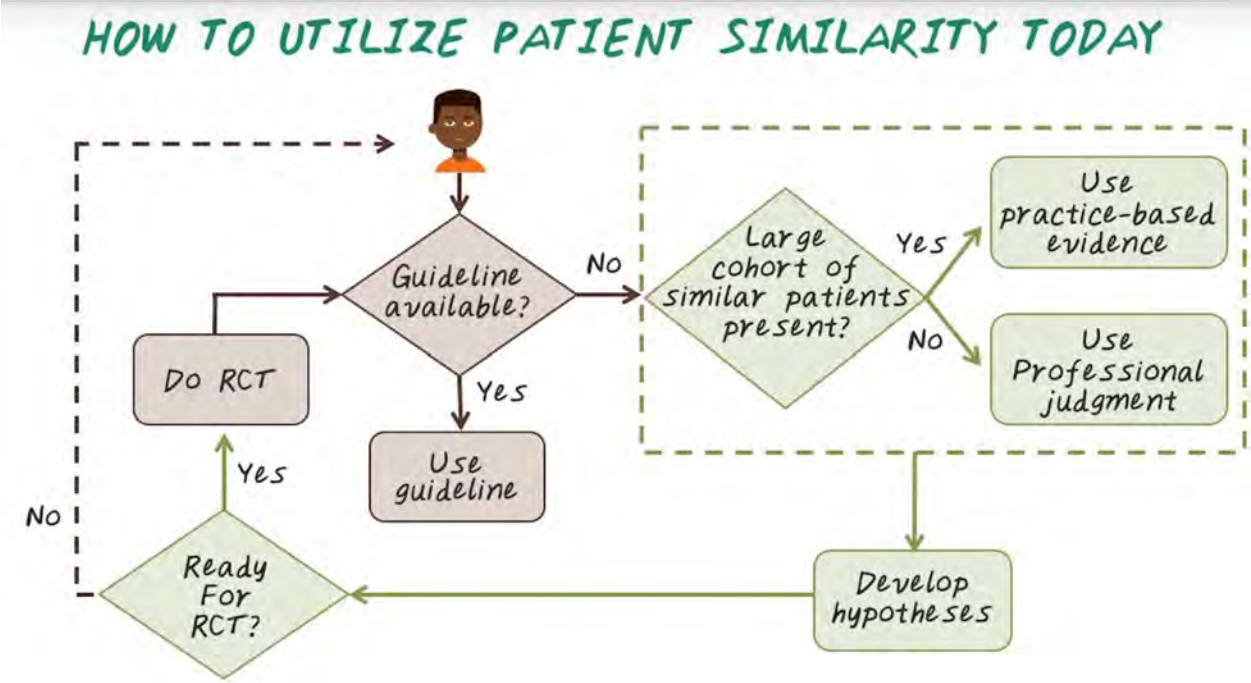
So let's go through this list, one by one. First, pragmatic trials cannot test new drugs. Because pragmatic trials depends on the historical data and all the treatments already happened in the past. In that case, we won't have any information about effectiveness of a new drug. And pragmatic trials do 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.

9. 9. Utilize Patient Similarity

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

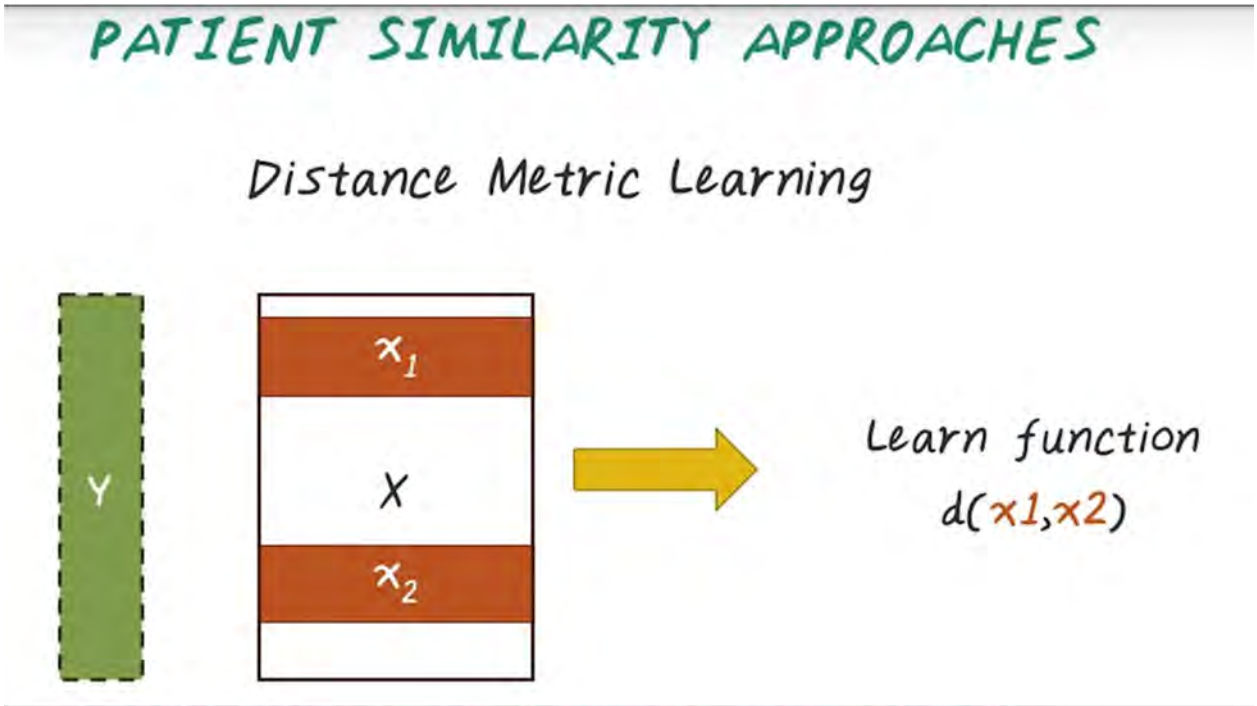
thing we can do is, try to see whether appropriate guideline available to apply to the given individual. If there are a proper guideline, then we can directly use the clinical guideline to treat this patient. If we dont have a guideline, that is suitable for this individual, we could go ahead look for 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.

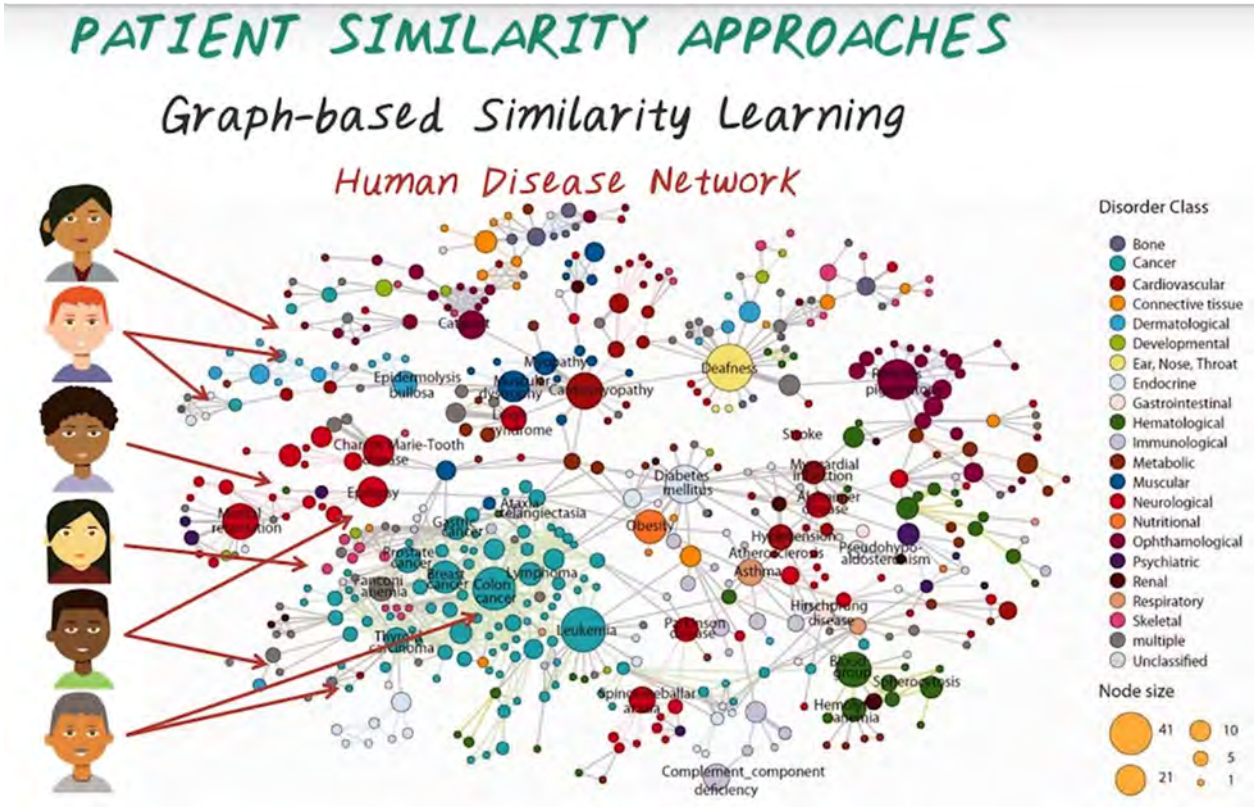




10. [10. Patient Similarity Approaches]

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 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 ontology, 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?





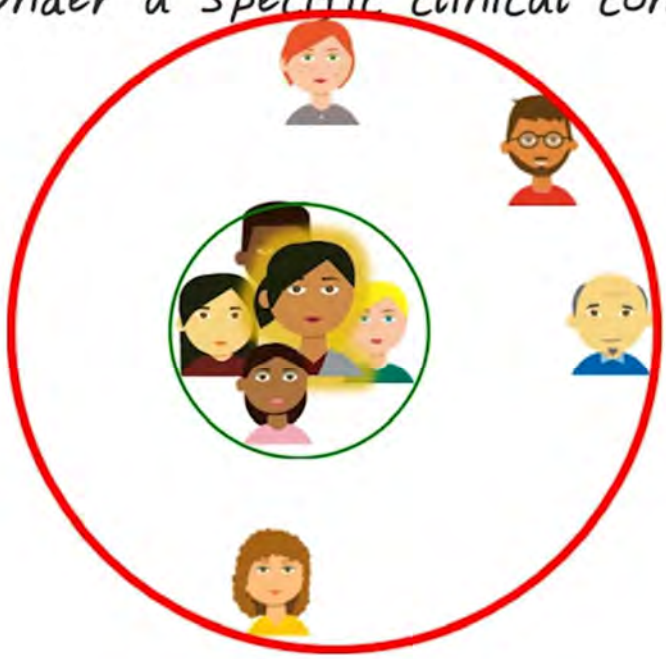
11. [11. Patient Similarity Through LS]

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 fracturize 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 fracturized 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.

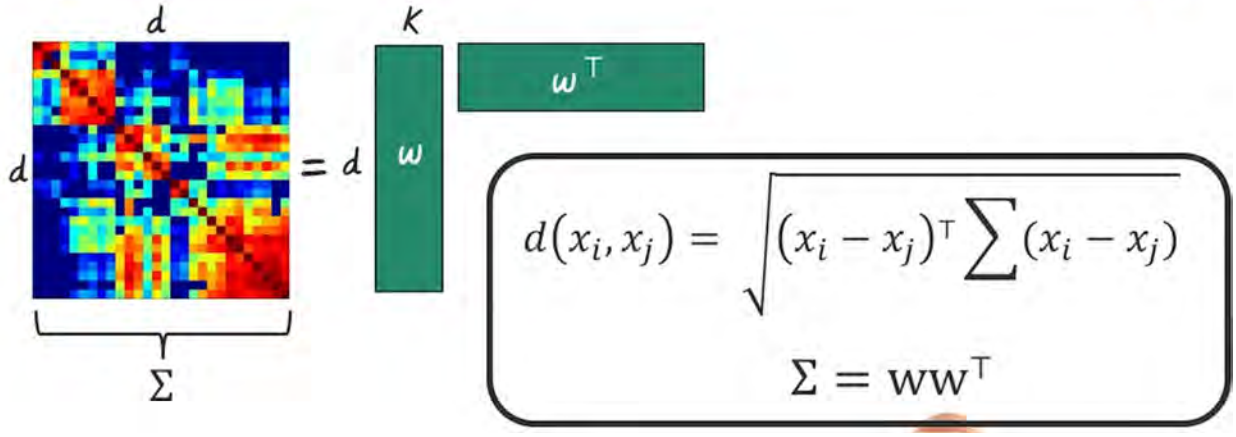
PATIENT SIMILARITY THROUGH LOCALLY SUPERVISED METRIC LEARNING

Under a specific clinical context



PATIENT SIMILARITY THROUGH LOCALLY SUPERVISED METRIC LEARNING

Goal: Learn a generalized Mahalanobis distance for a specific clinical context (target label)



$$d(x_i, x_j) = \sqrt{(x_i - x_j)^\top \sum (x_i - x_j)}$$

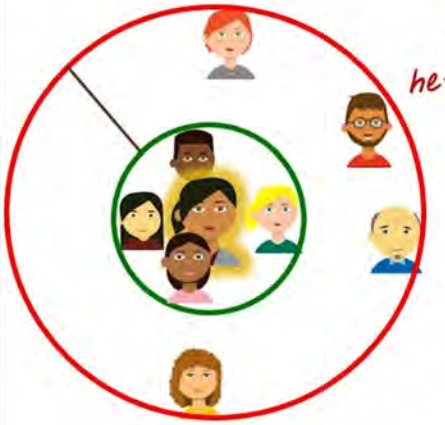
$$\Sigma = ww^\top$$

LOCALLY SUPERVISED METRIC LEARNING (LSML)

Margin for x_i $\quad \gamma_i = \sum_{k:x_k \in \mathcal{N}_i^e} \| w^T x_i - w^T x_k \|^2 - \sum_{j:x_j \in \mathcal{N}_i^o} \| w^T x_i - w^T x_j \|^2$

Total distance to heterogeneous neighbors

Total distance to homogeneous neighbors



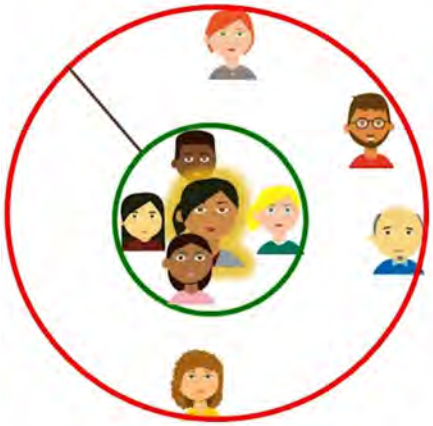
$\mathcal{N}_i^e$ heterogeneous neighbors

$\mathcal{N}_i^o$ homogeneous neighbors

Maximize the total margin

$$\gamma = \Sigma_i \Sigma_{k:x_k \in \mathcal{N}_i^e} w^T (x_i - x_k)(x_i - x_k)^T w - \Sigma_i \Sigma_{j:x_j \in \mathcal{N}_i^o} w^T (x_i - x_j)(x_i - x_j)^T w$$

LOCALLY SUPERVISED METRIC LEARNING (LSML)



OPTIMIZATION PROBLEM

$$\max_{W:\, W^T W = I} \operatorname{tr}(W^T H W)$$

where

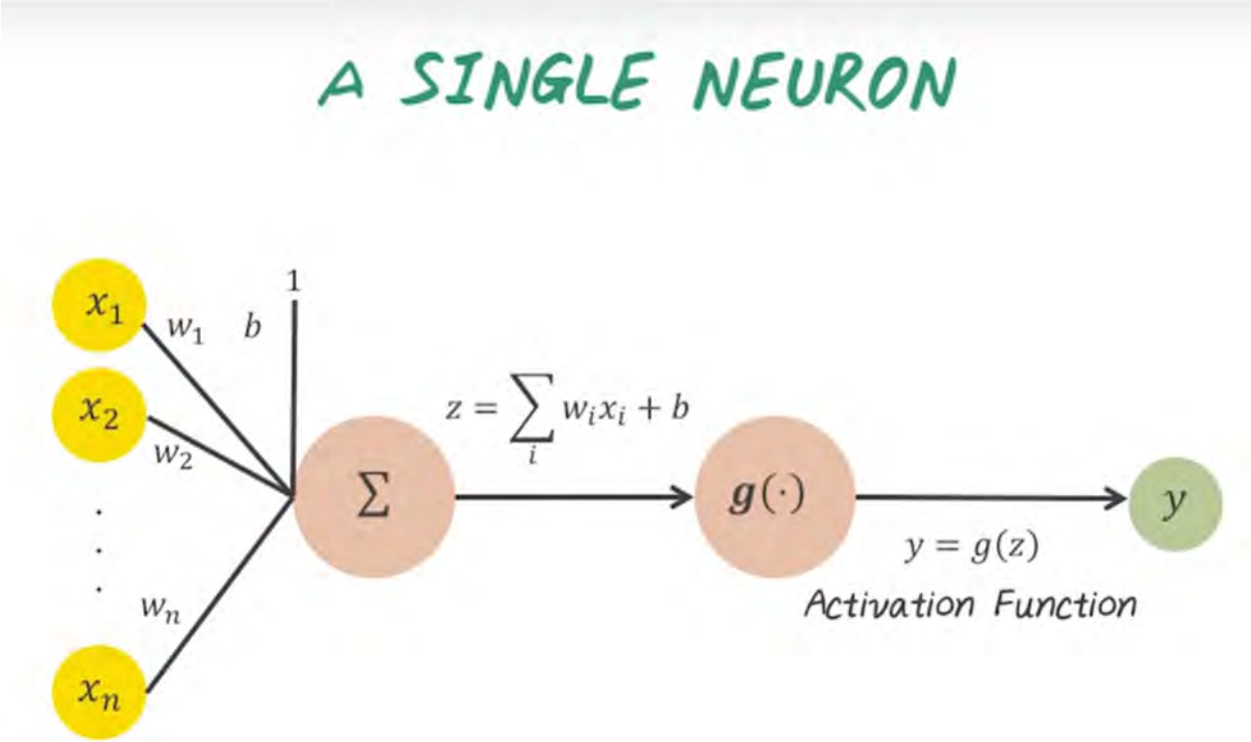
$$L^o = \Sigma_i \Sigma_{j:x_j \in \mathcal{N}_i^o} (x_i - x_j)(x_i - x_j)^T$$

$$L^e = \Sigma_i \Sigma_{k:x_k \in \mathcal{N}_i^e} (x_i - x_k)(x_i - x_k)^T$$

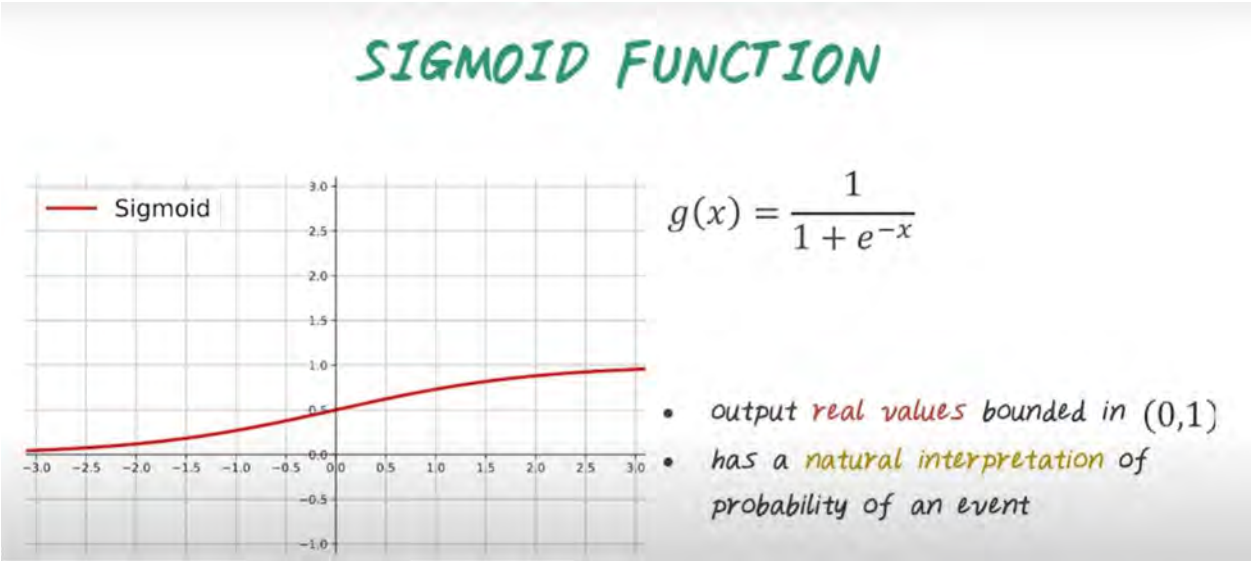
$$H = L^e - L^o$$

Deep Neural Network (DNN)

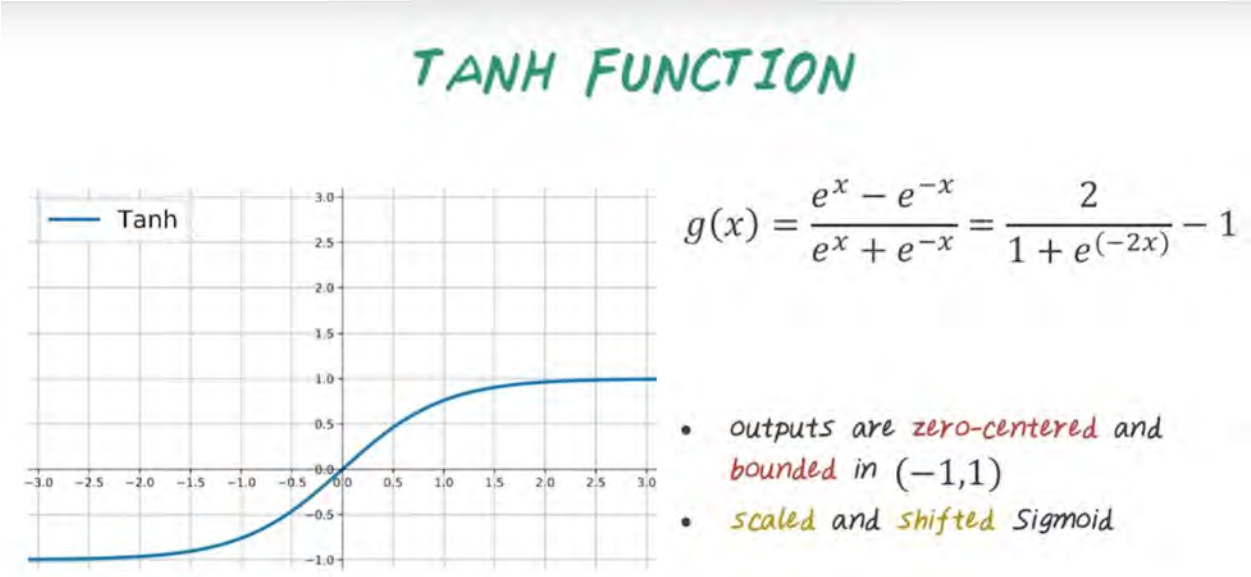
1. 1. Backward Computation for a Neuron



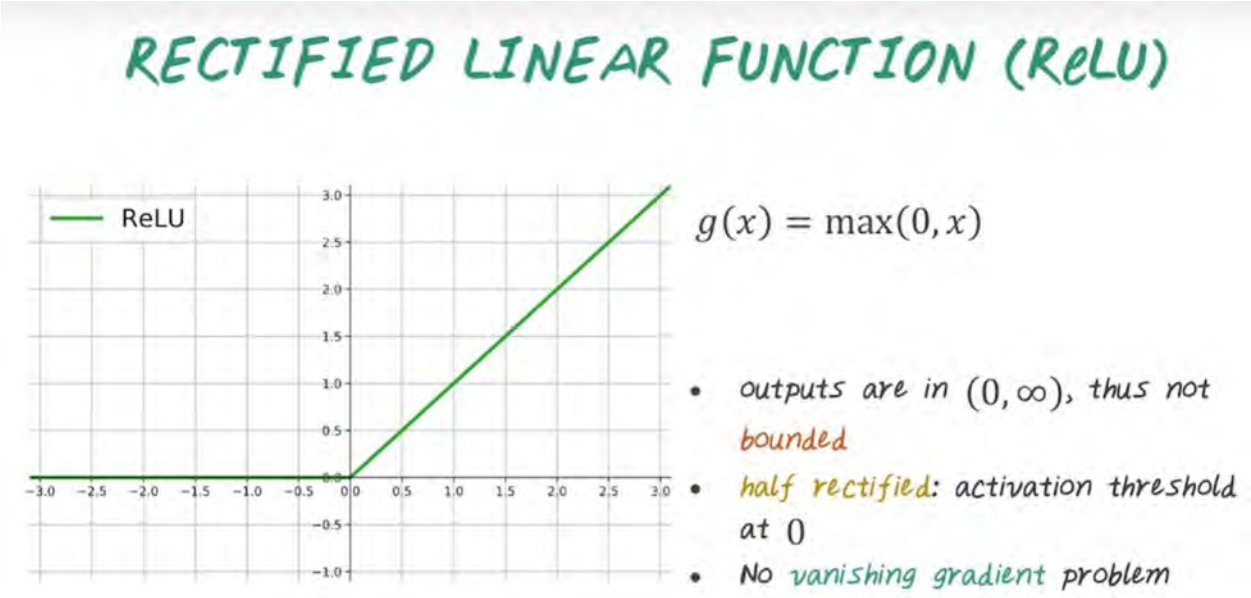
2. 2. Sigmoid Function



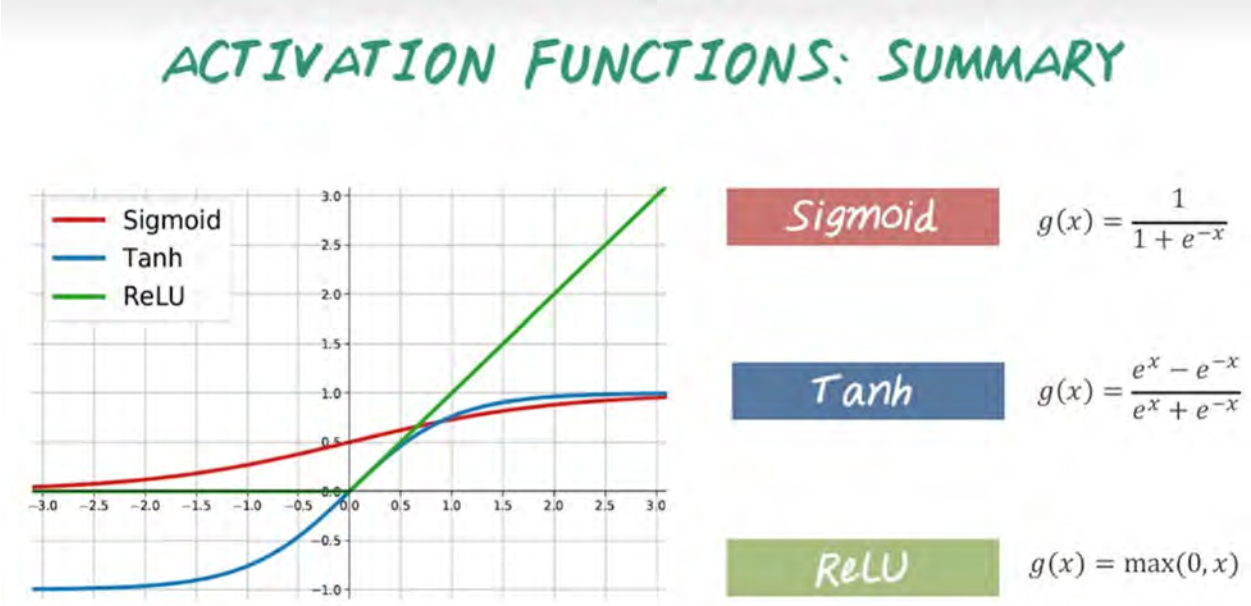
3. 3. TANH Function



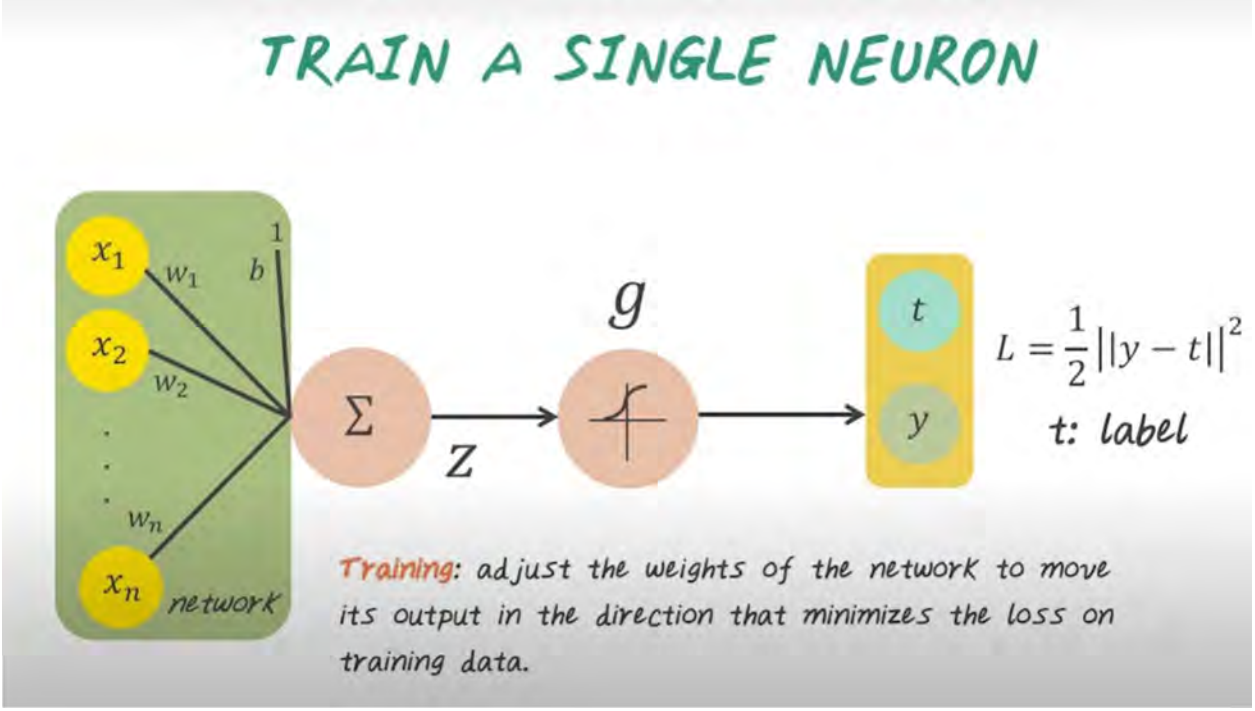
4. 4. Rectified Linear Function



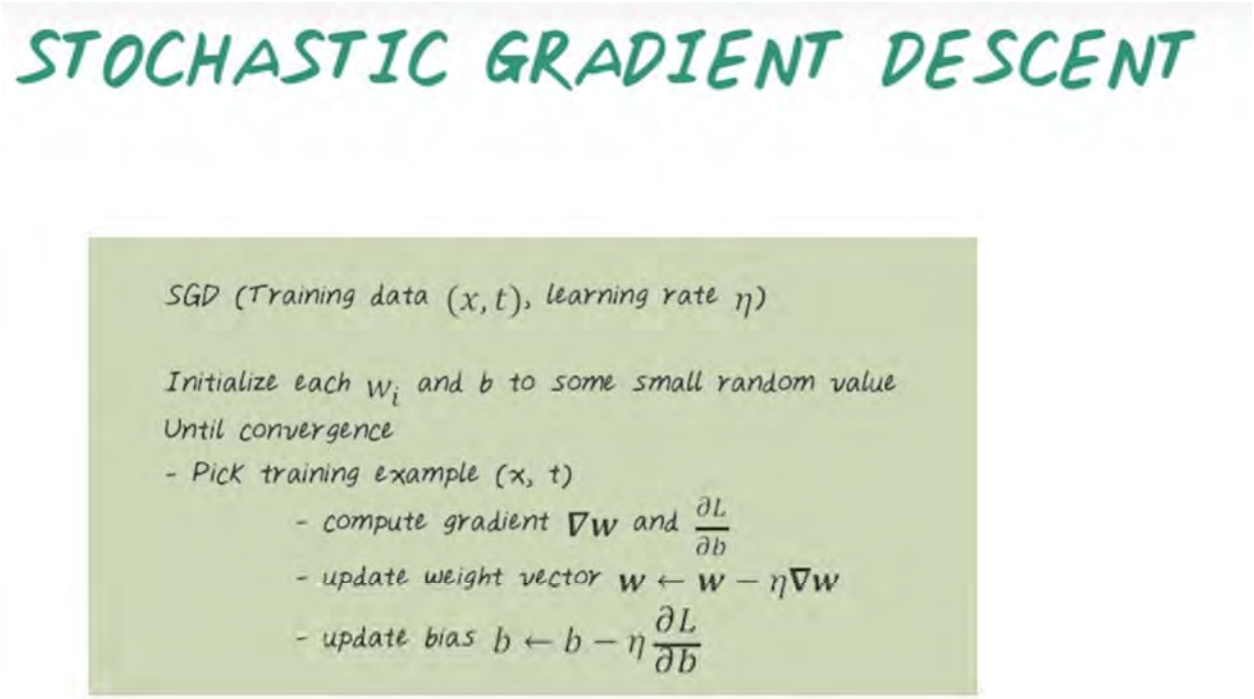
5. 5. Activation Functions: Summary



6. 6. Train a Single Neuron

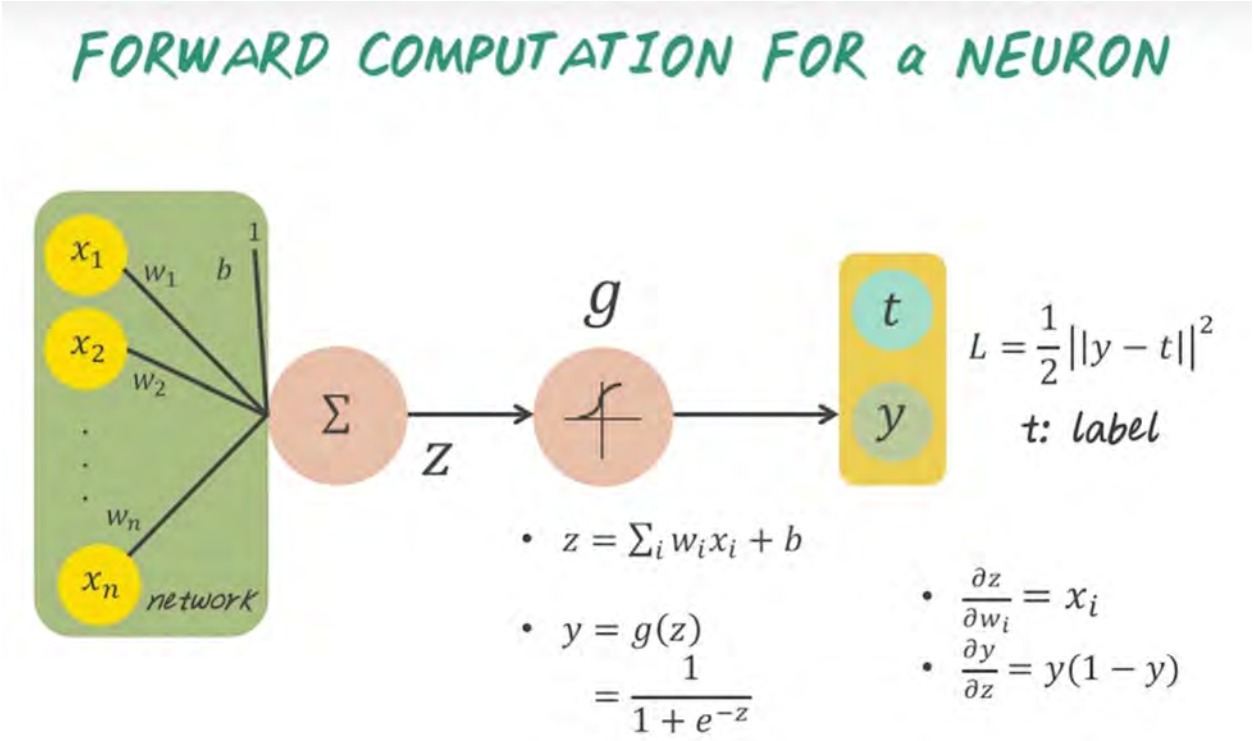


7. 7. Stochastic Gradient Descent

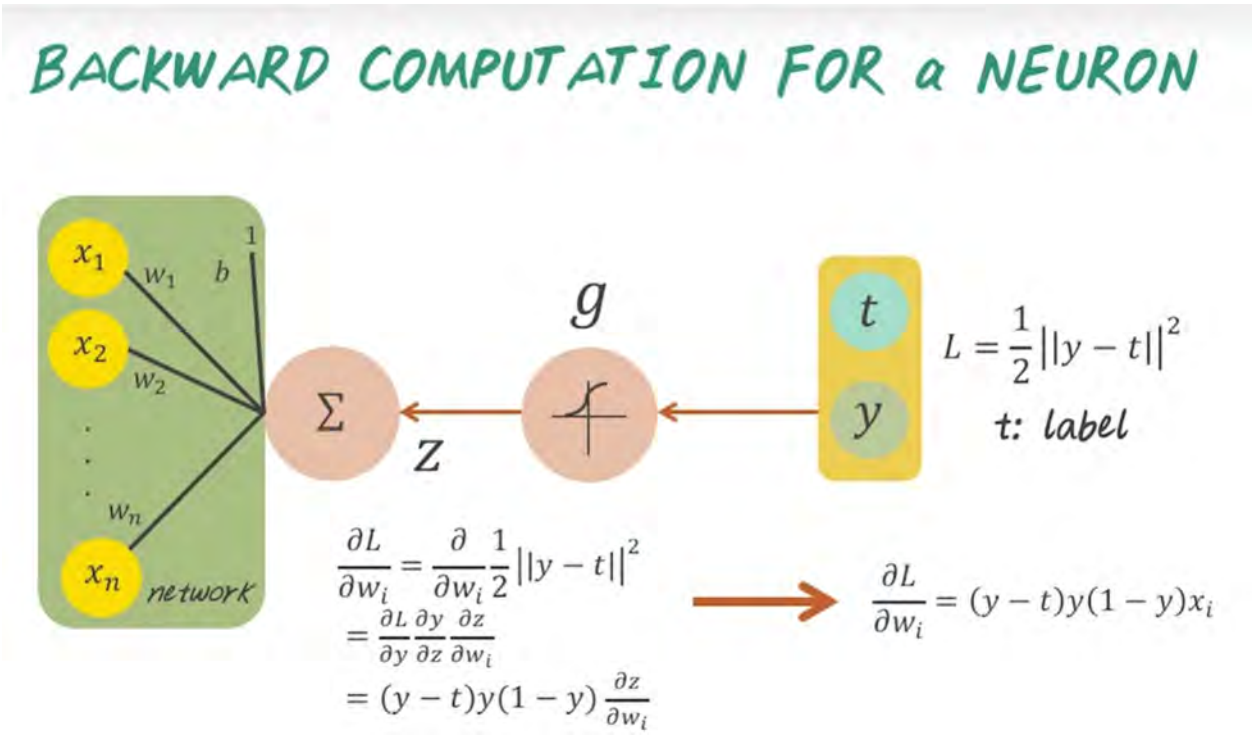


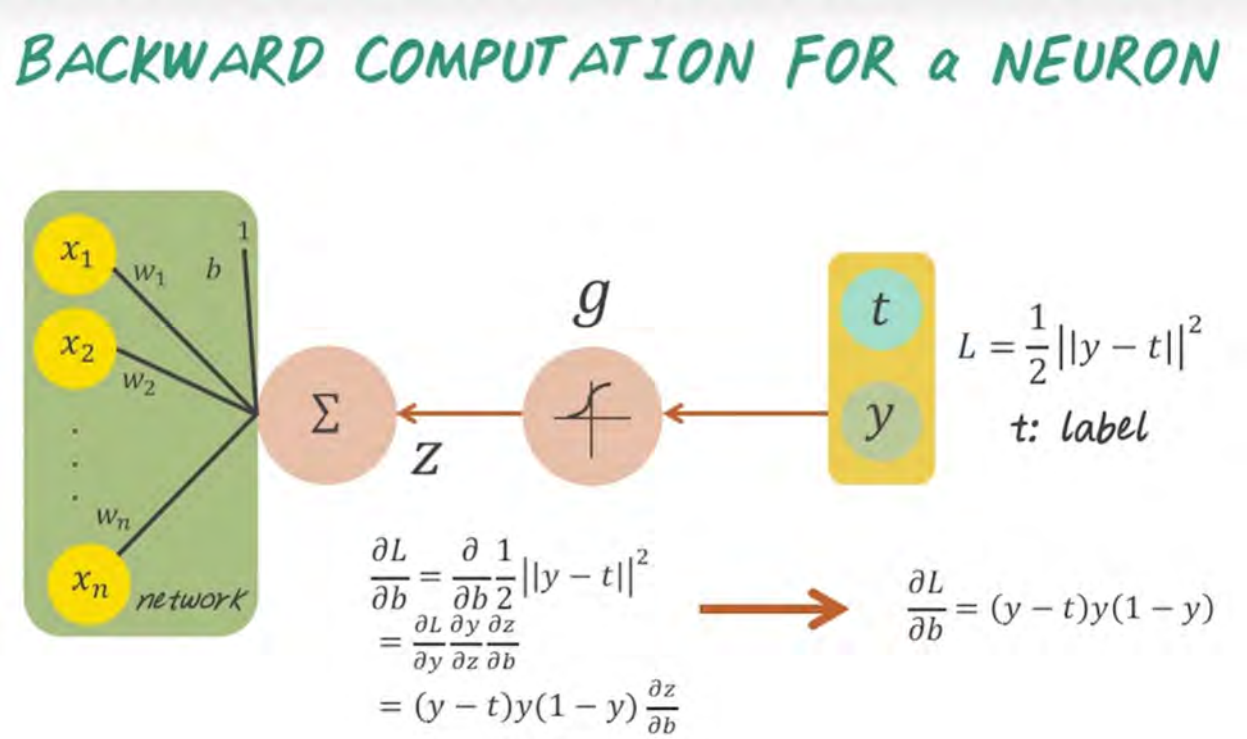
$$\nabla w = \left[\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \cdots, \frac{\partial L}{\partial w_n}\right]$$

8. 8. Forward Computation for a Neuron

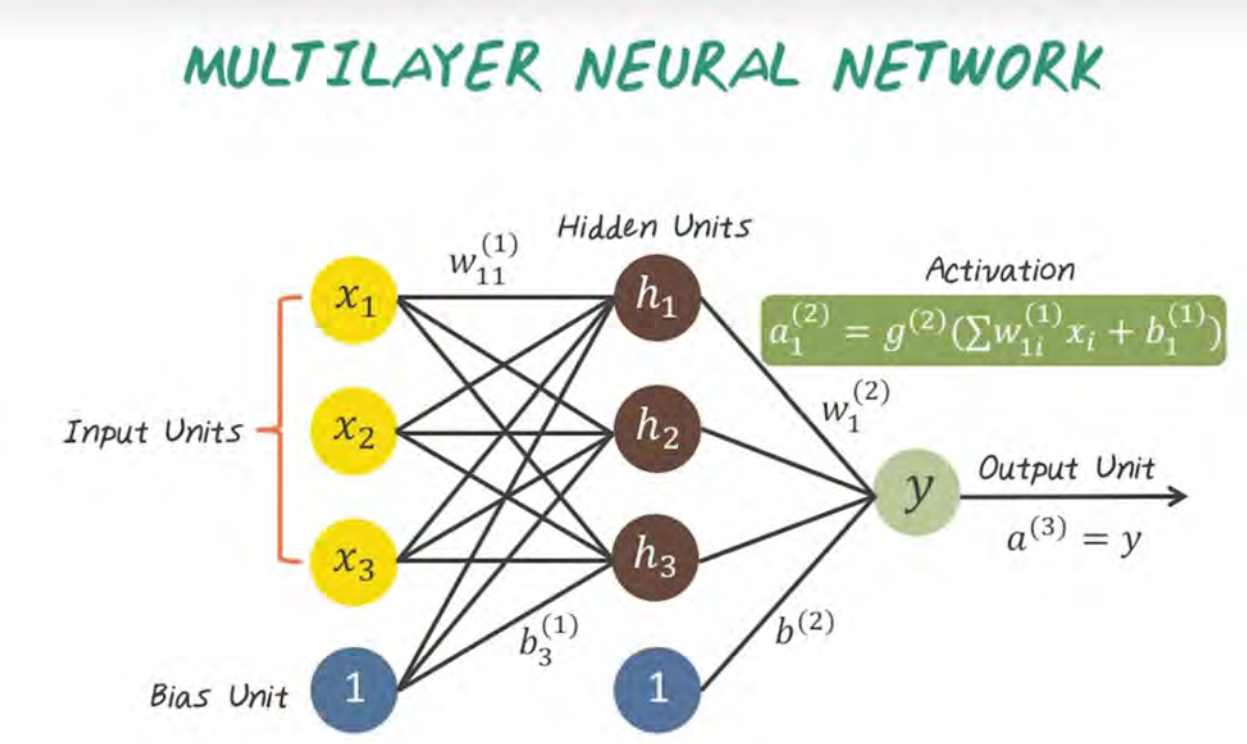


9. [9. Backward Computation for a Neuron]





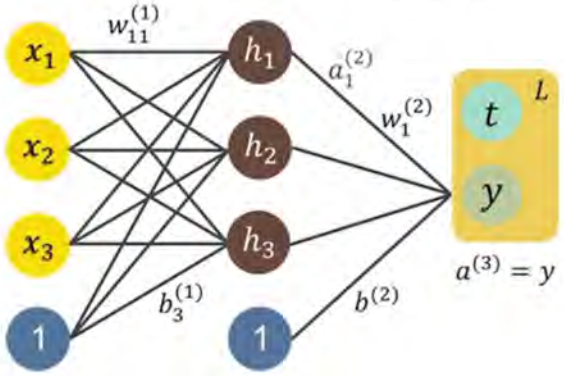
10. [10. Multilayer Neural Network]



11. [11. Train a Multilayer Neural Network]

TRAIN A MULTILAYER NEURAL NETWORK

Given training sample (x, t)



$$L(\boldsymbol{W}, \boldsymbol{b}) = \tfrac{1}{2}\|y - t\|^2$$

SGD (Training data (x, t), learning rate η)

Initialize each $w_{ji}^{(l)}$ and $b_j^{(l)}$ to small random value

Until convergence

DO

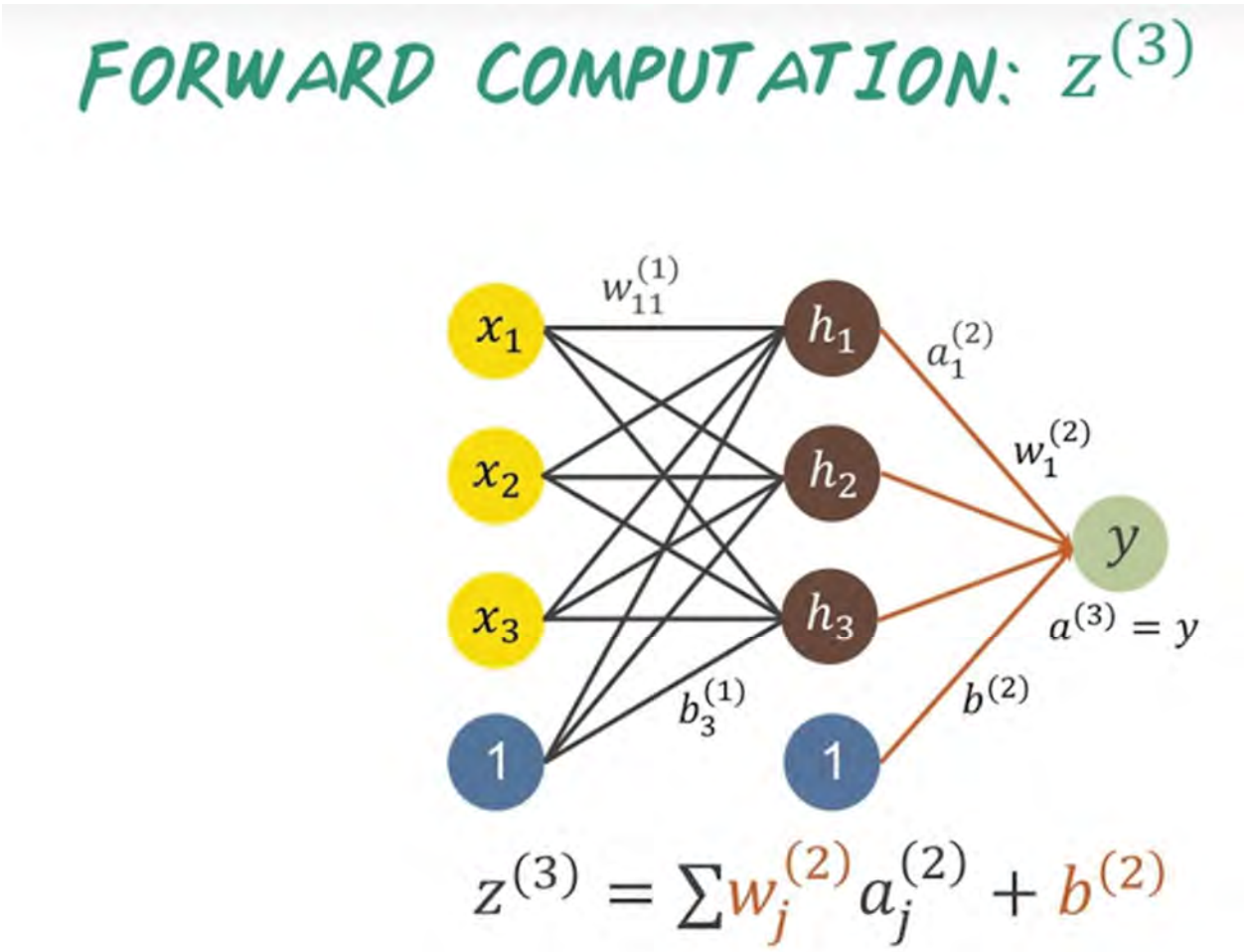
For each (x, t) in training data, DO

- compute $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial w_{ji}^{(l)}}$ and $\frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial b_j^{(l)}}$
- For each $w_{ji}^{(l)}$ and $b_j^{(l)}$:

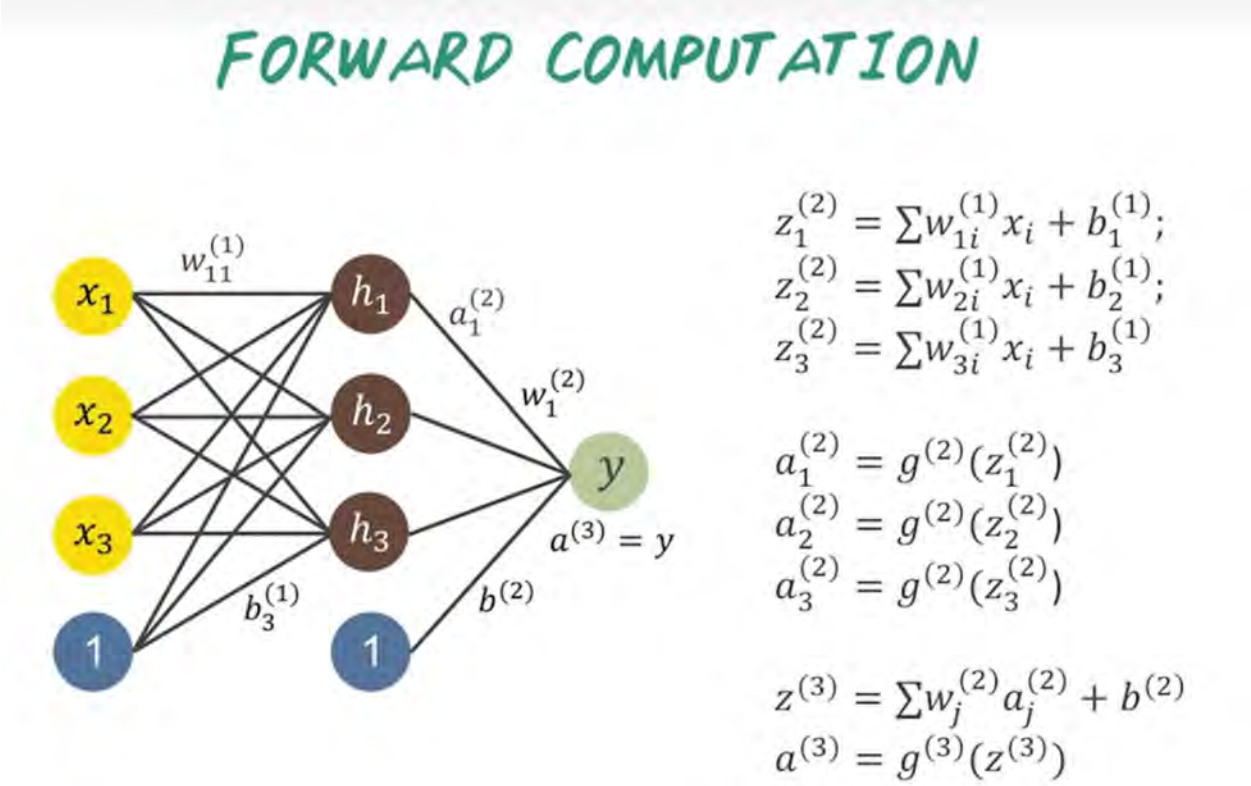
$$w_{ji}^{(l)} = w_{ji}^{(l)} - \eta \frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial w_{ji}^{(l)}}$$

$$b_j^{(l)} = b_j^{(l)} - \eta \frac{\partial L(\boldsymbol{W},\boldsymbol{b})}{\partial b_j^{(l)}}$$

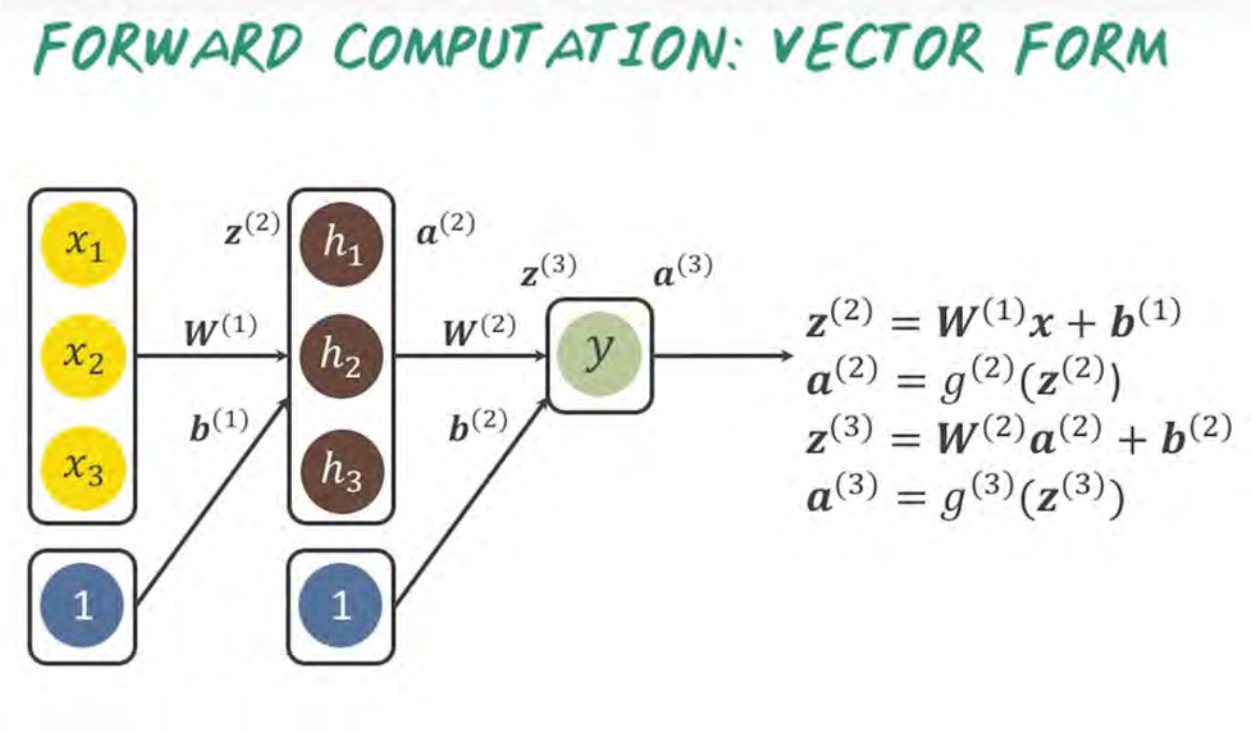
12. 12. Forward Computation, part 1



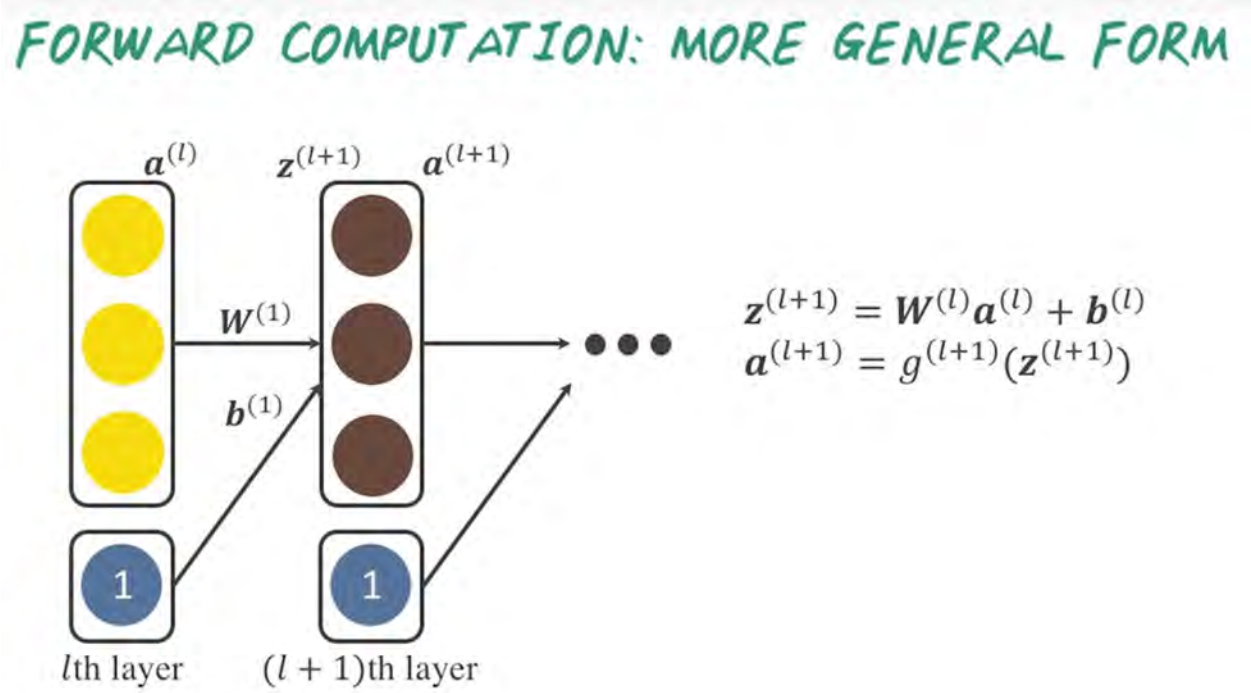
13. [13. Forward Computation, part 2]



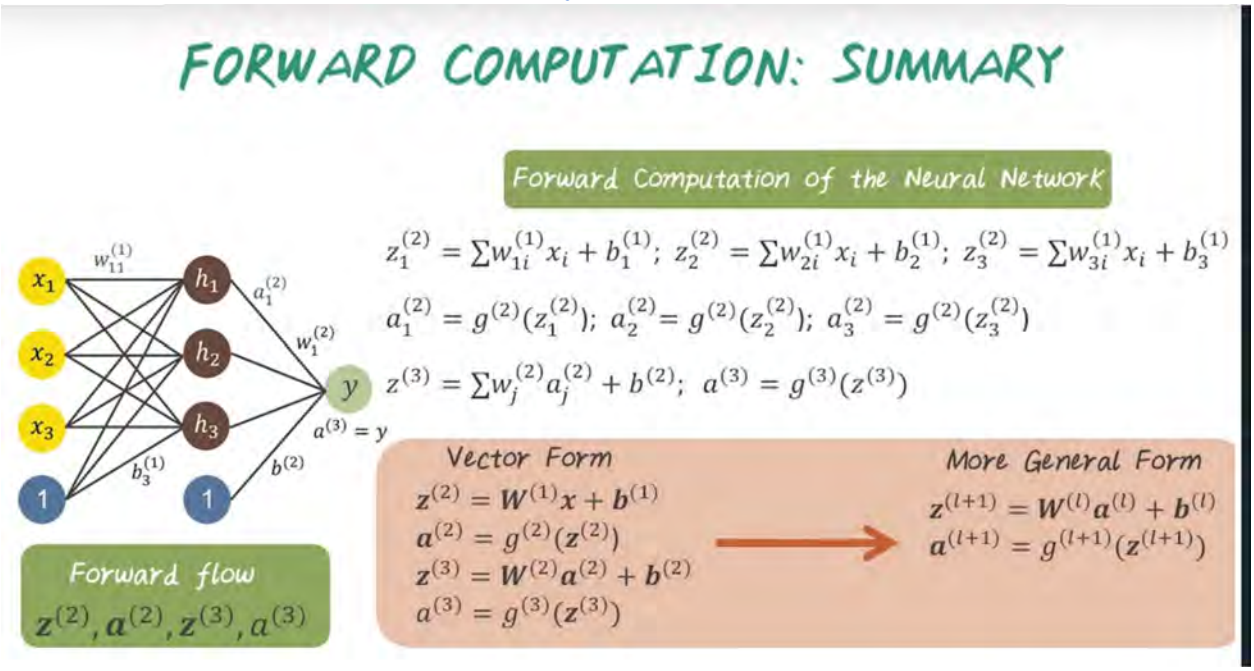
14. [14. Forward Computation: Vector Form]



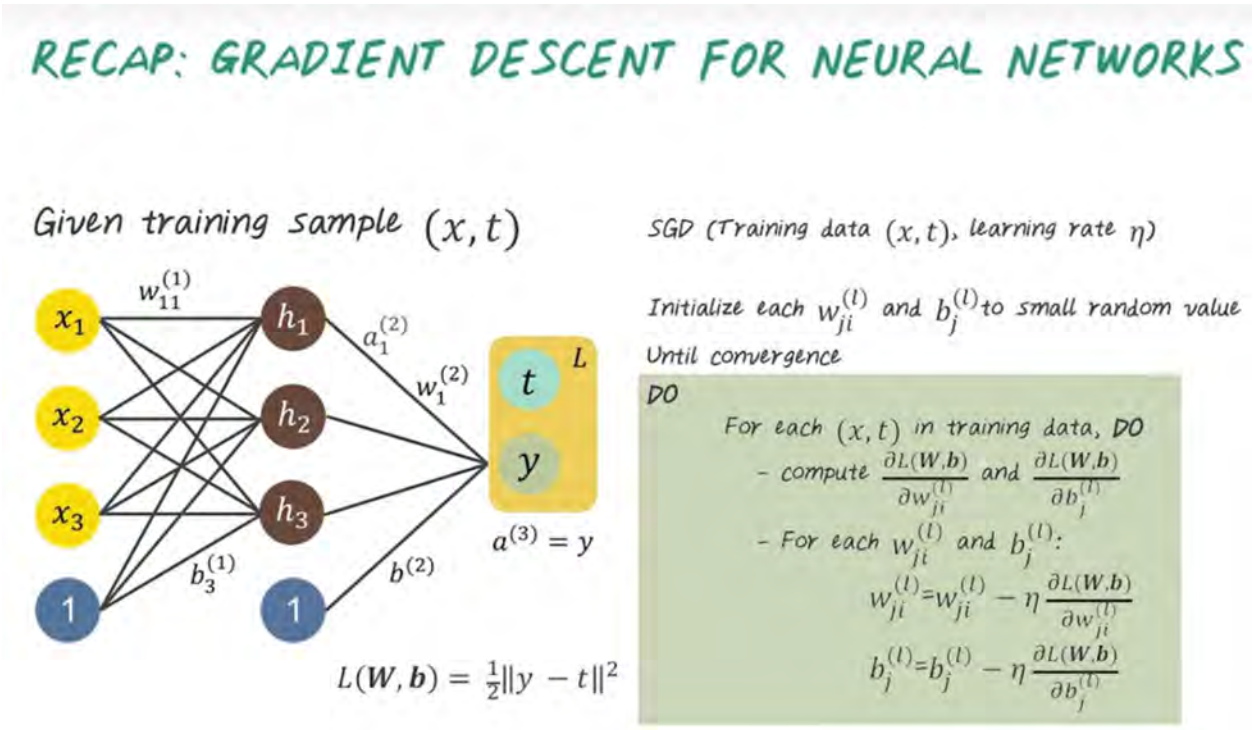
15. [15. Forward Computation: More General Form]



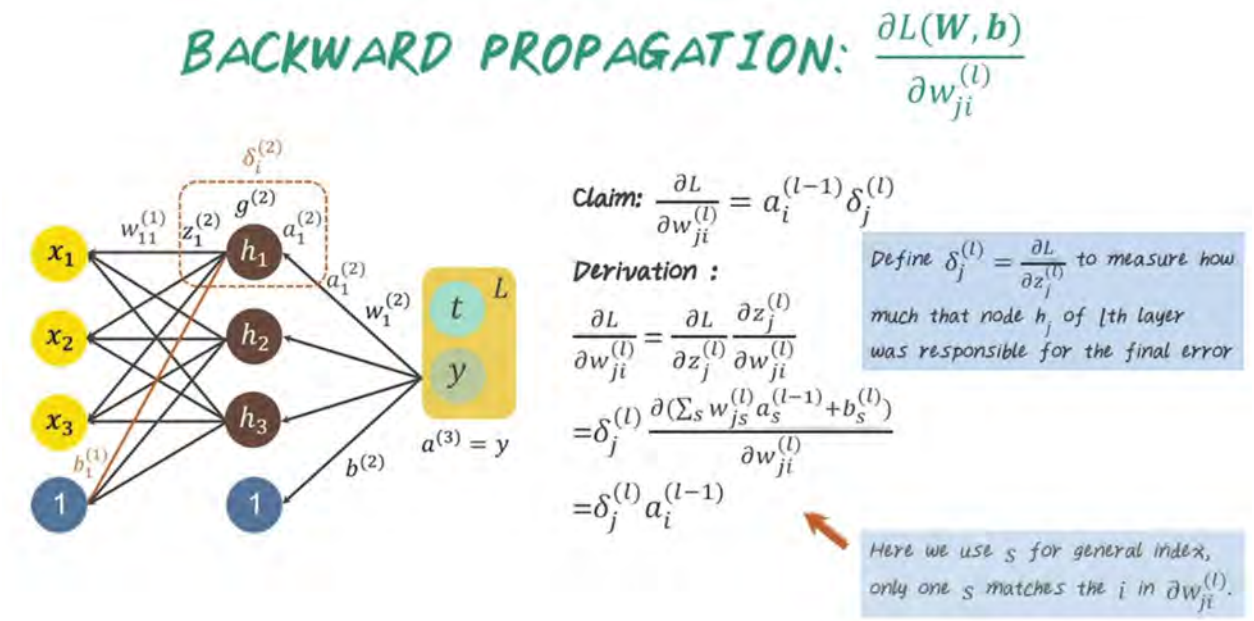
16. [16. Forward Computation: Summary]



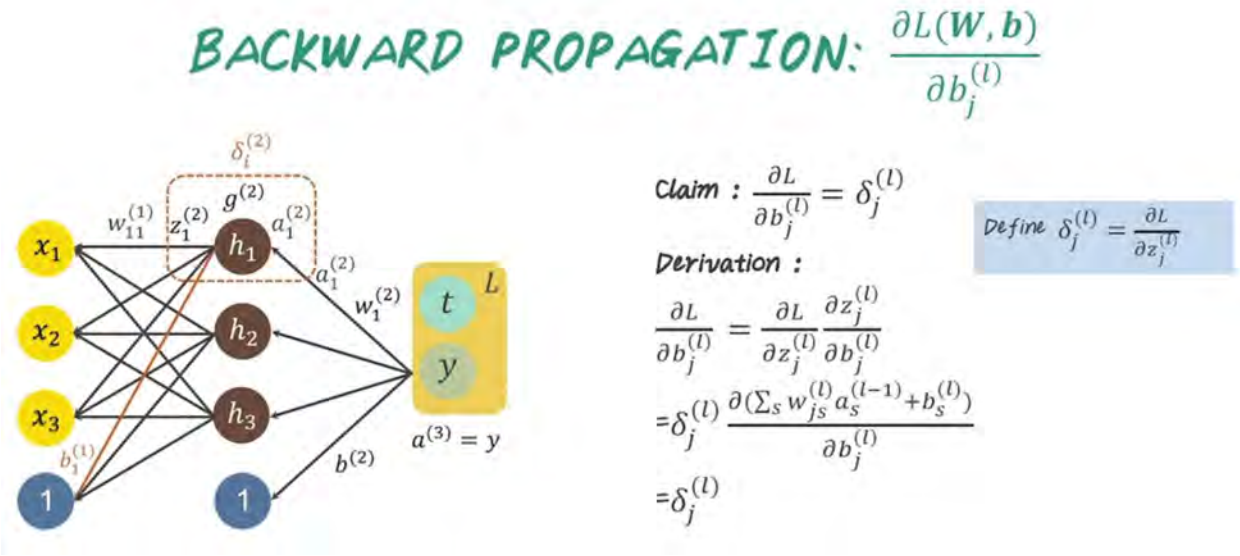
17. [17. Gradient Descent for Neural Networks]



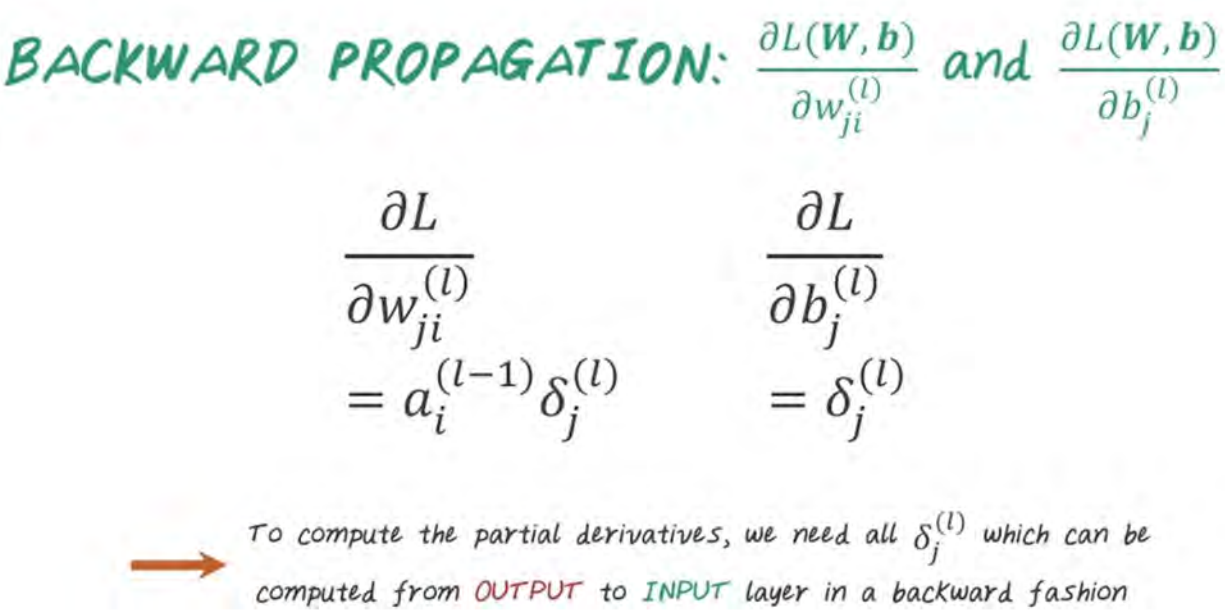
18. [18. Backward Propagation, part 1]



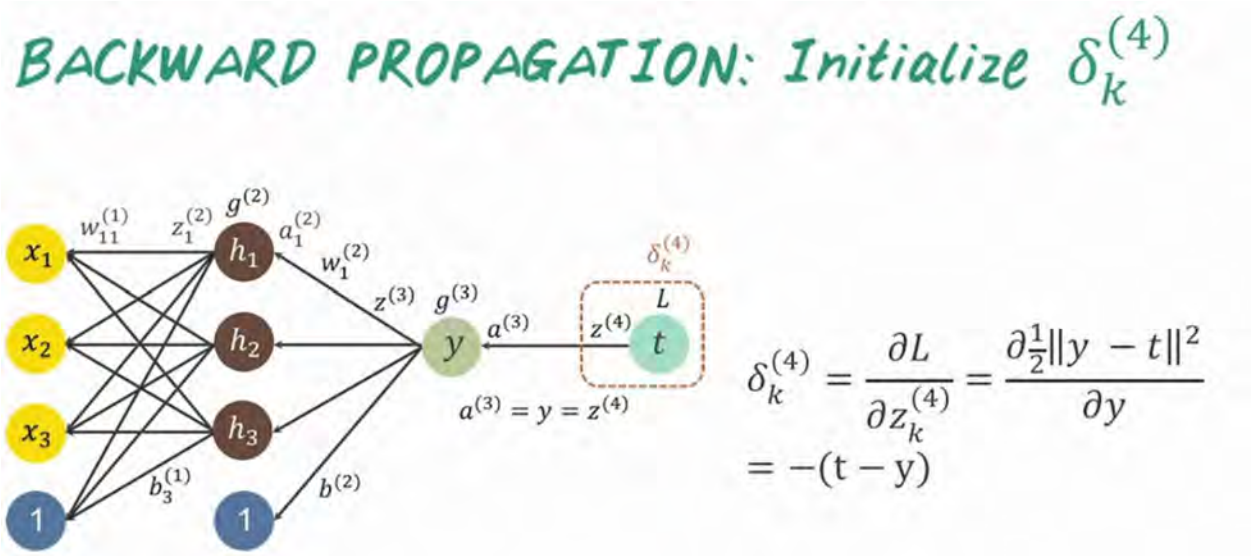
19. [19. Backward Propagation, part 2]



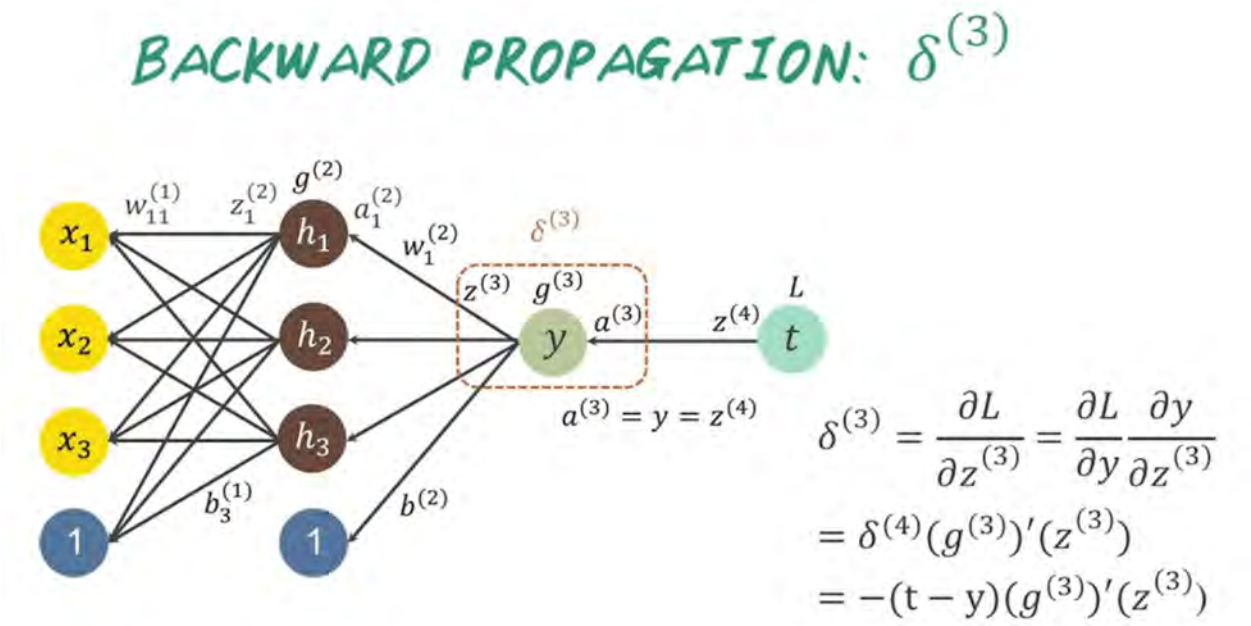
20. [20. Backward Propagation, part 3]



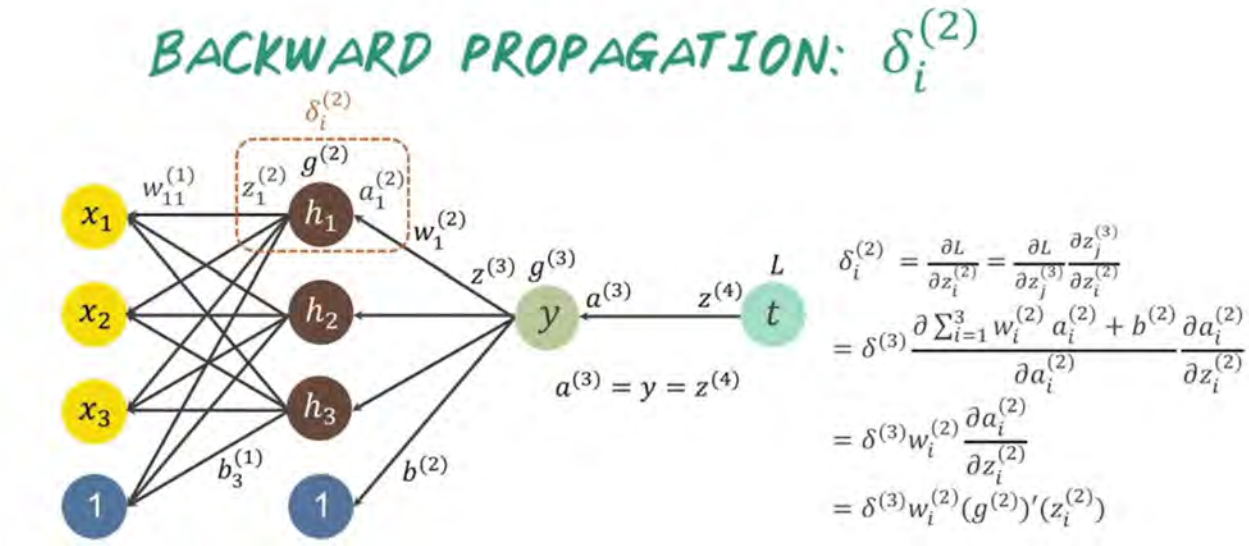
21. [21. Backward Propagation, part 4]



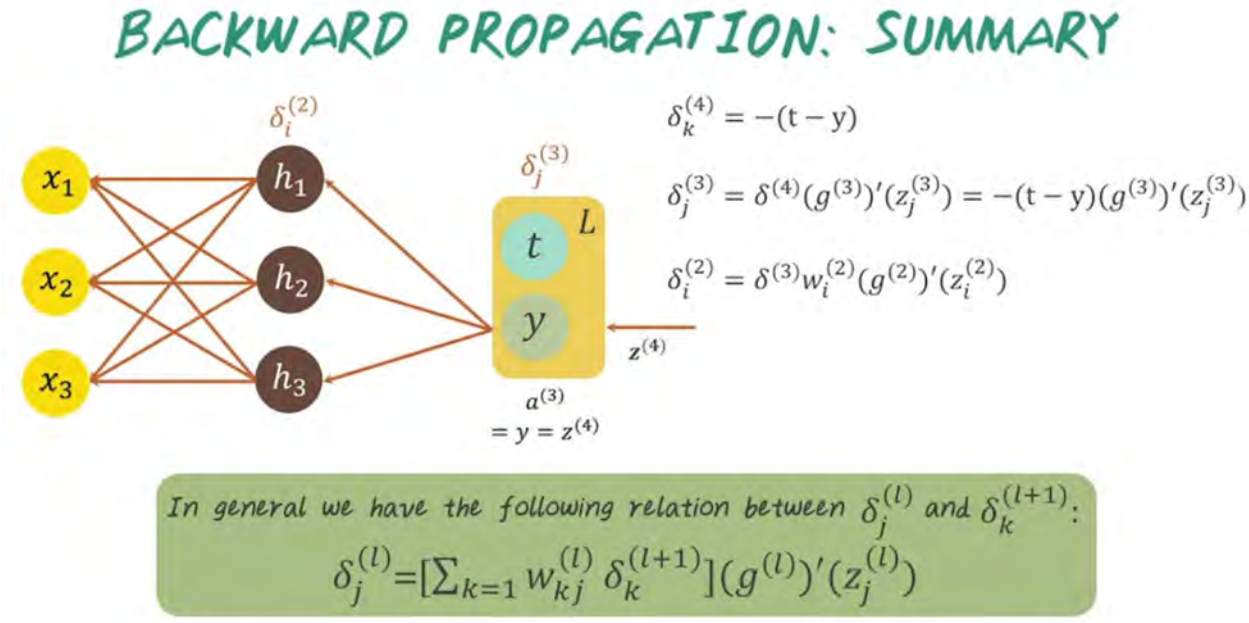
22. 22. Backward Propagation, part 5



23. 23. Backward Propagation, part 6



24. 24. Backward Propagation: Summary



25. 25. Backpropagation Algo

BACKPROPAGATION ALGORITHM

The FORWARD pass

Starting with the input x, go forward to output layer, compute and store the variables $z^{(2)}, a^{(2)}, z^{(3)}, a^{(3)}, \cdots, z^{(L)}, a^{(L)}, z^{(L+1)}$

The BACKWARD pass

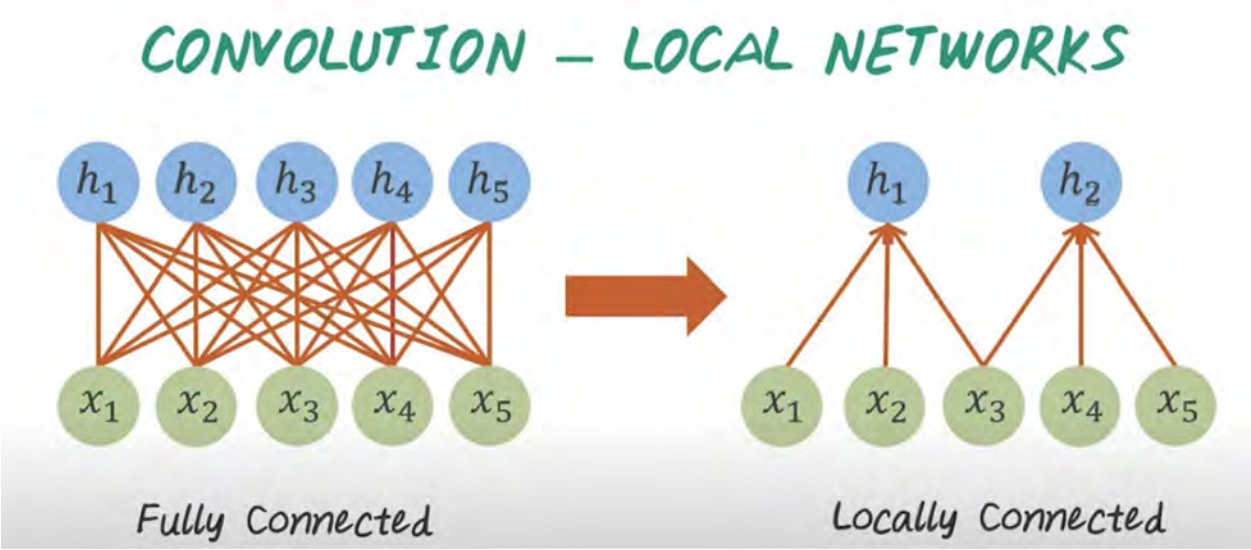
Initialize $\delta^{(L+1)} = -(t - y) = -(t - z^{(L+1)})$

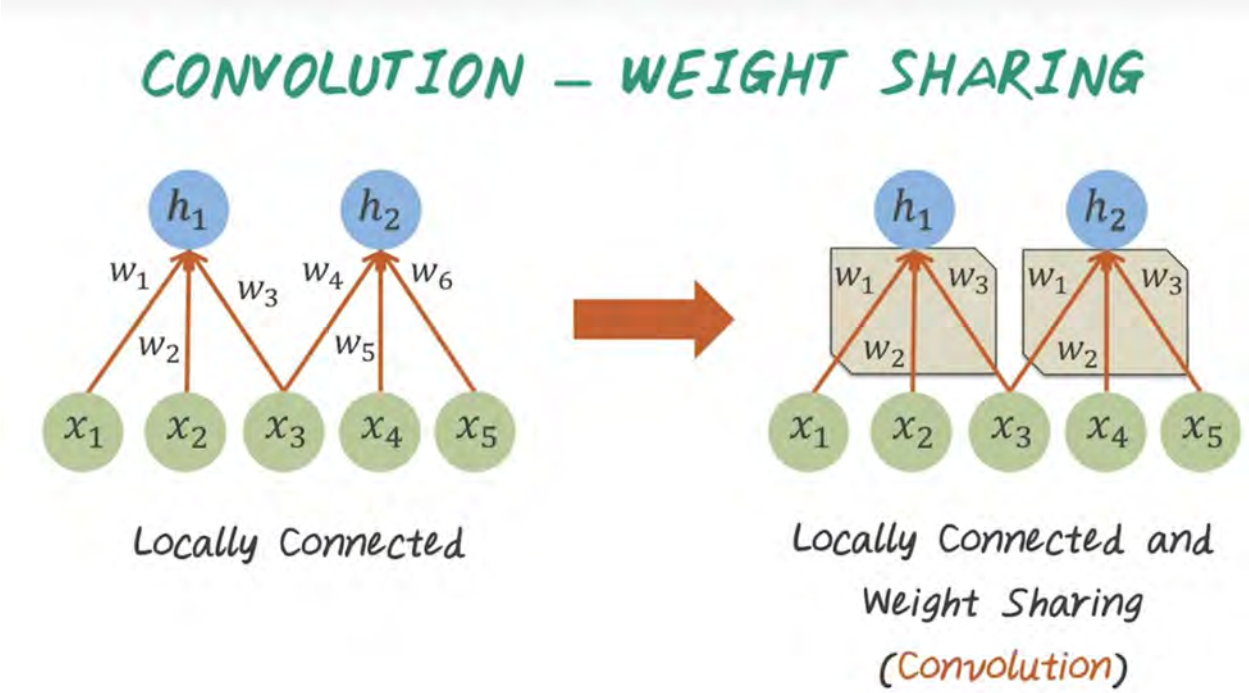
Compute $\delta_j^{(l)} = [\sum_{k=1} w_{kj}^{(l)} \delta_k^{(l+1)}](g^{(l)})'(z_j^{(l)})$ and compute the derivatives at layer l:

1. $\frac{\partial L}{\partial w_{ji}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}$
2. $\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$

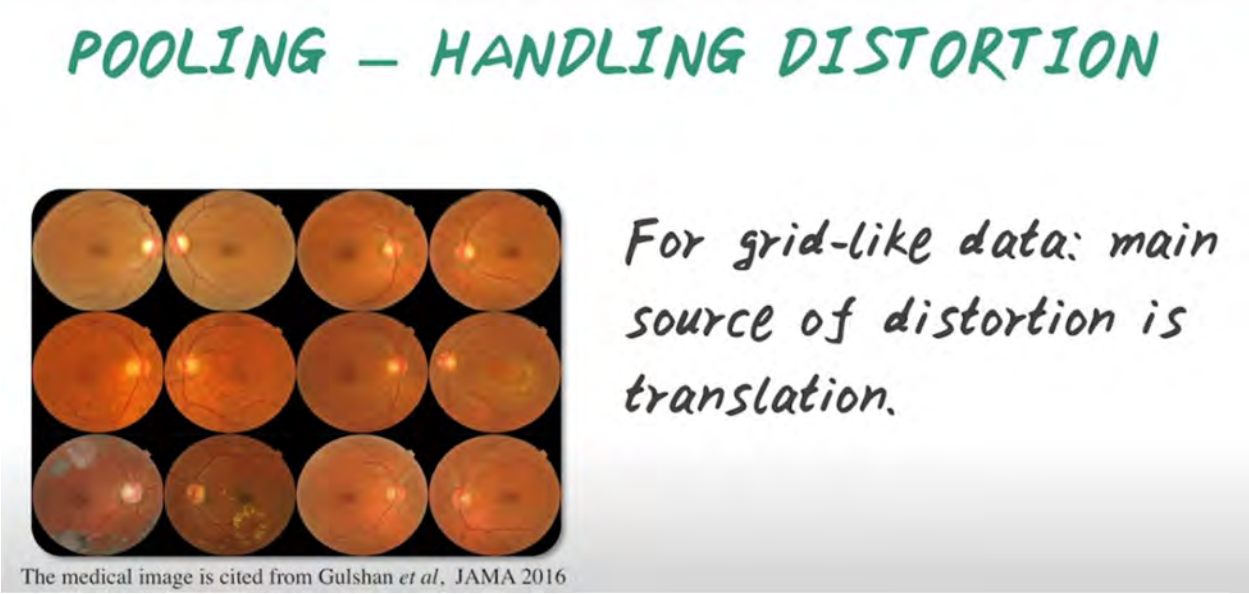
Convolutional Neural Networks (CNN)

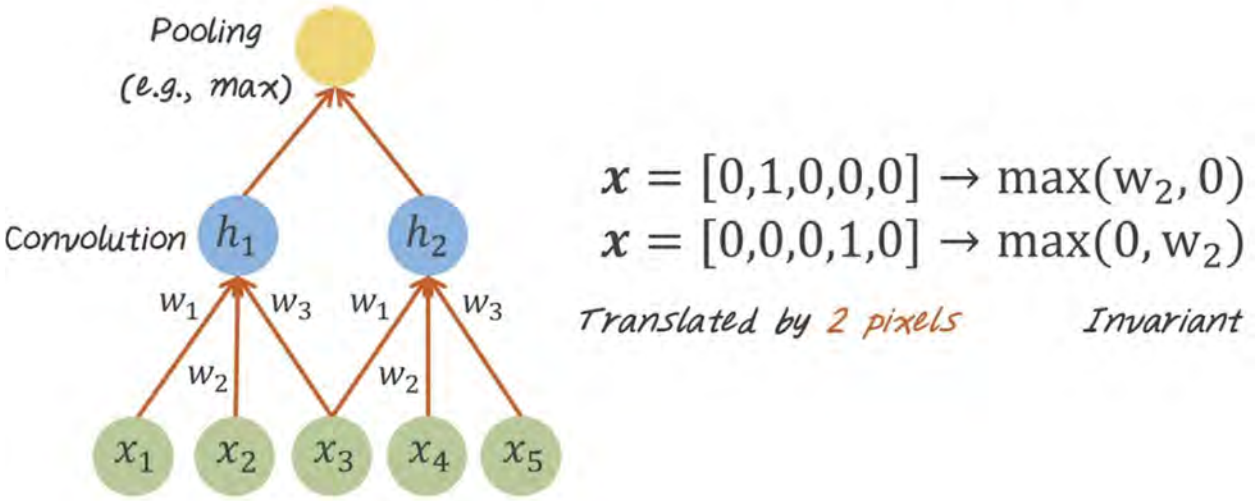
1. 1. Convolution Local Networks



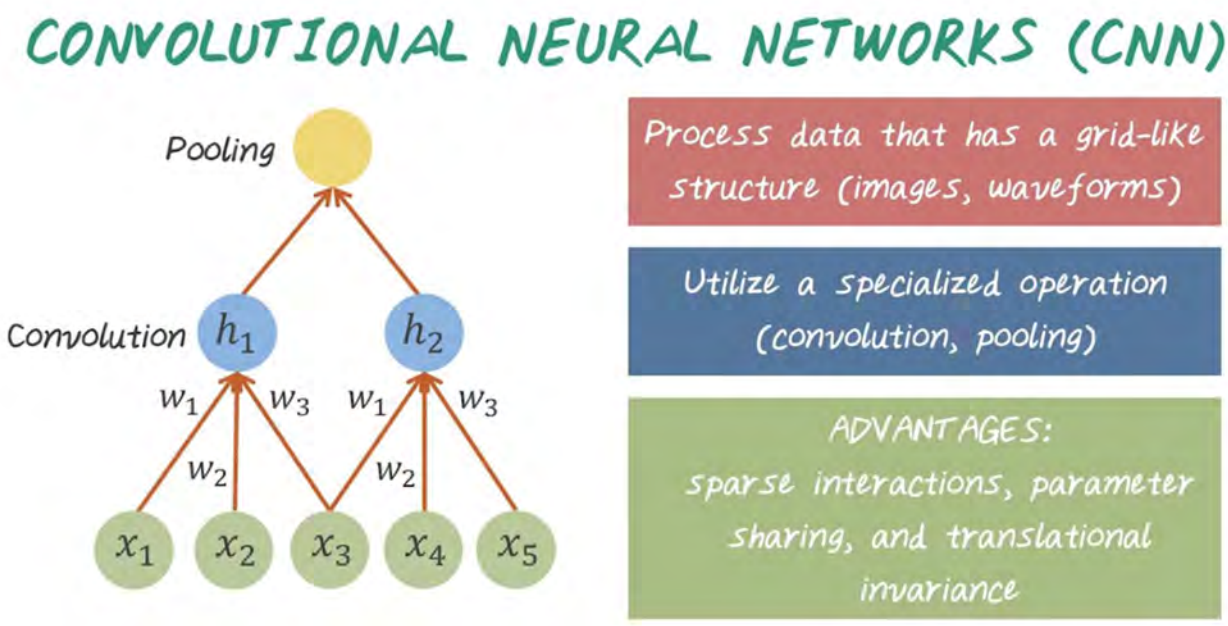


2. [2. Pooling Handling Distortion]

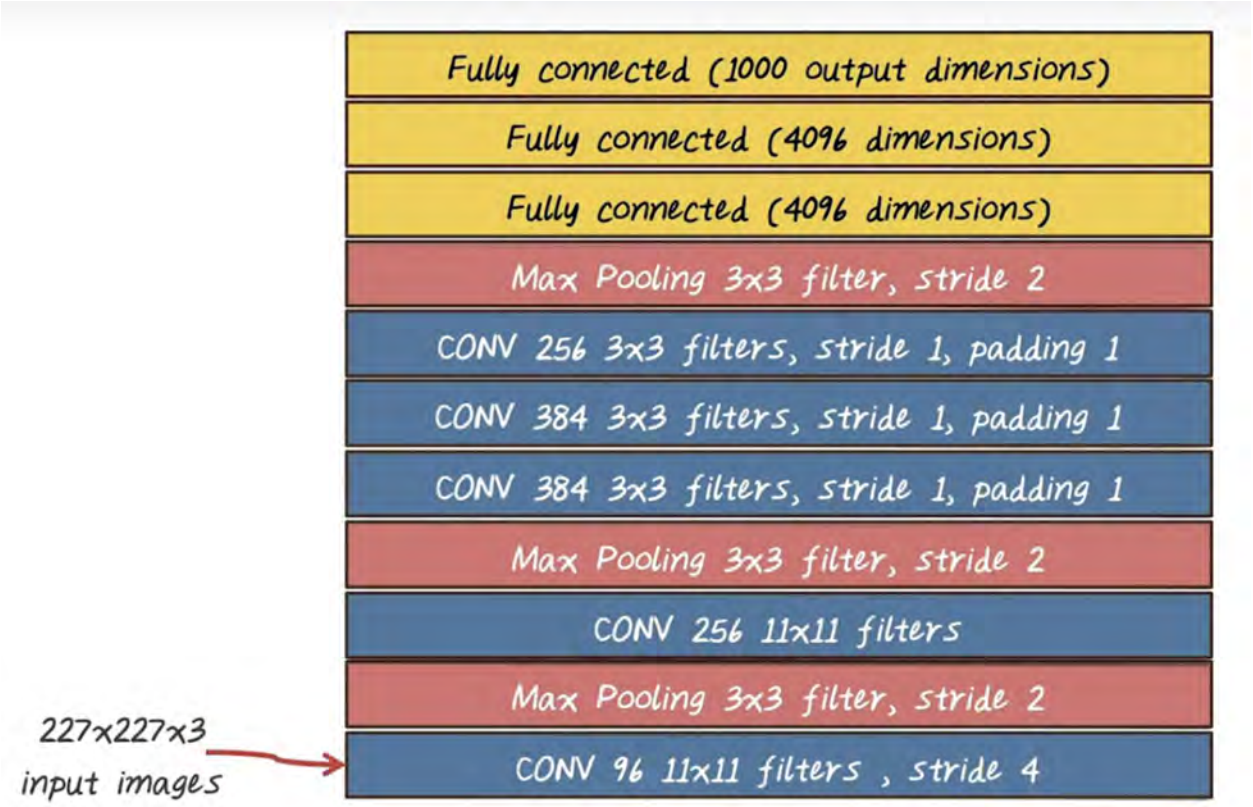




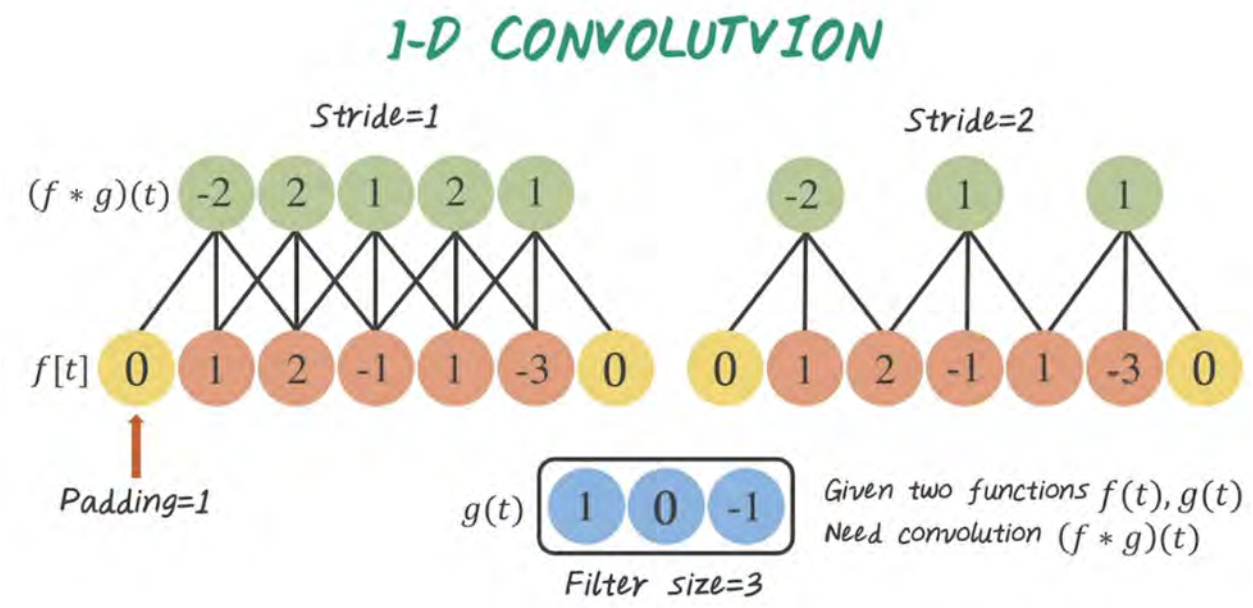
3. [3. Convolutional Neural Networks]



4. [4. Key Stages of CNN]

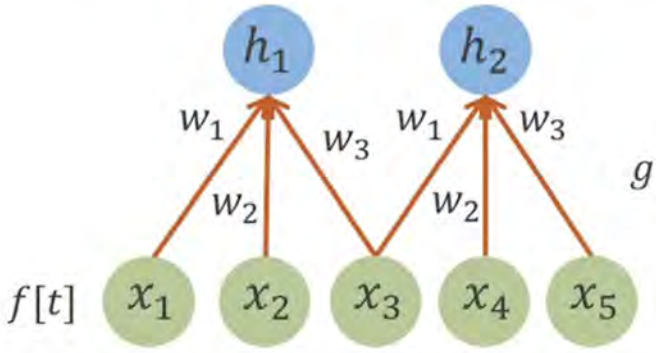


5. [5. 1-D Convolution]



6. [6. Neural Network for 1-D Convolution]

NEURAL NETWORK for 1-D CONVOLUTION

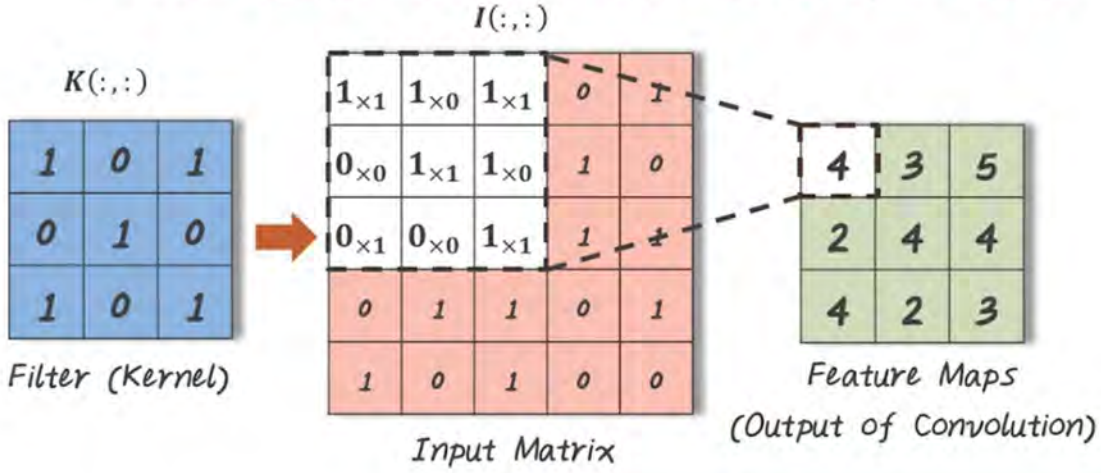


$$g[t] = [w_1, w_2, w_3]$$

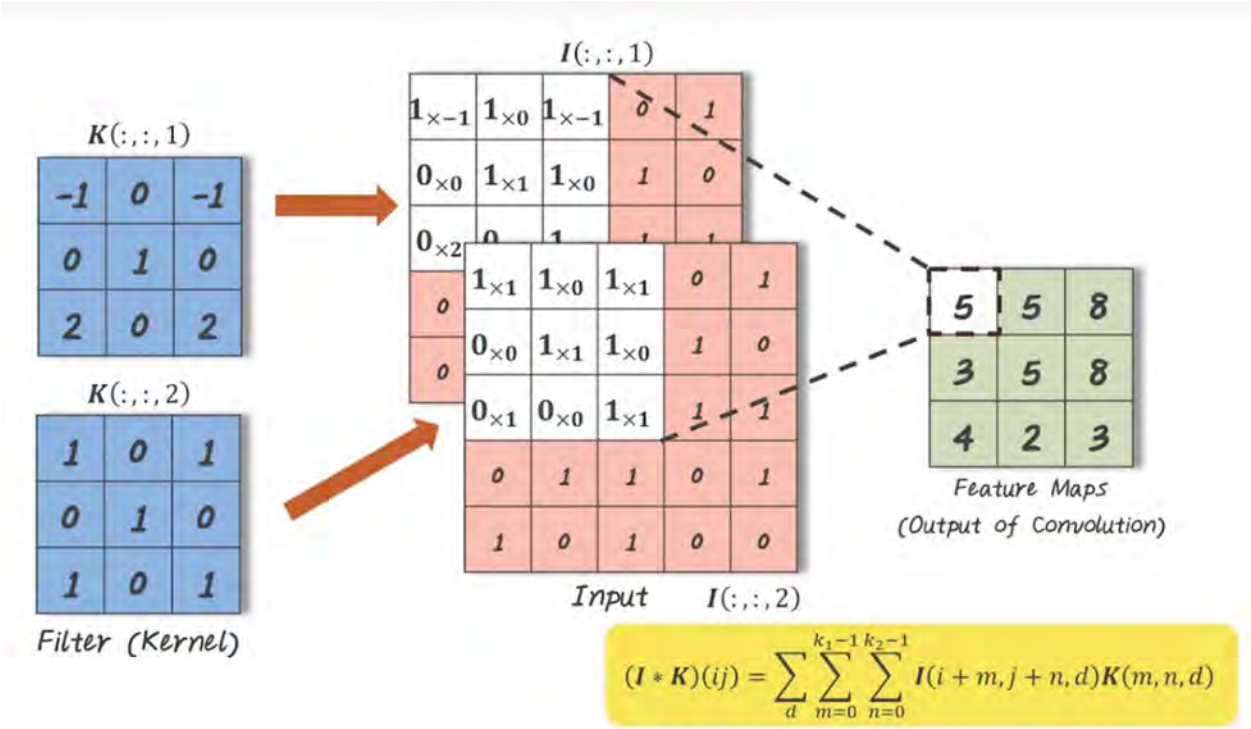
$$h_1 = (f * g)(t)_1 = w_1x_1 + w_2x_2 + w_3x_3$$
$$h_2 = (f * g)(t)_2 = w_1x_3 + w_2x_4 + w_3x_5$$

7. [7. 2-D Convolution Operation]

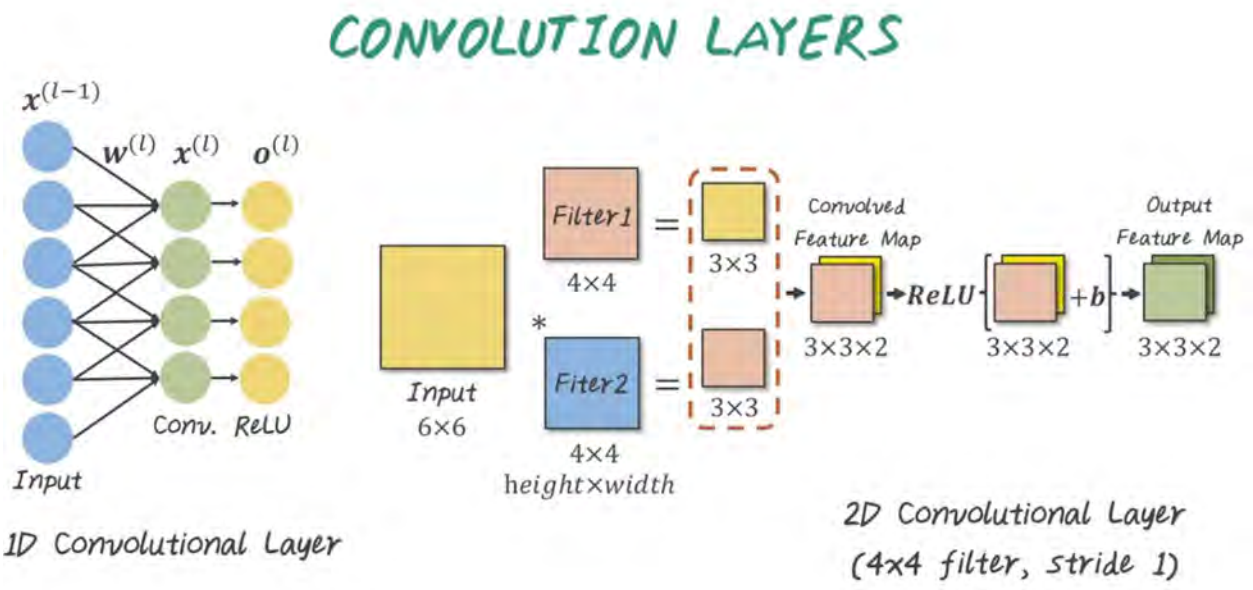
2-D CONVOLUTION OPERATION



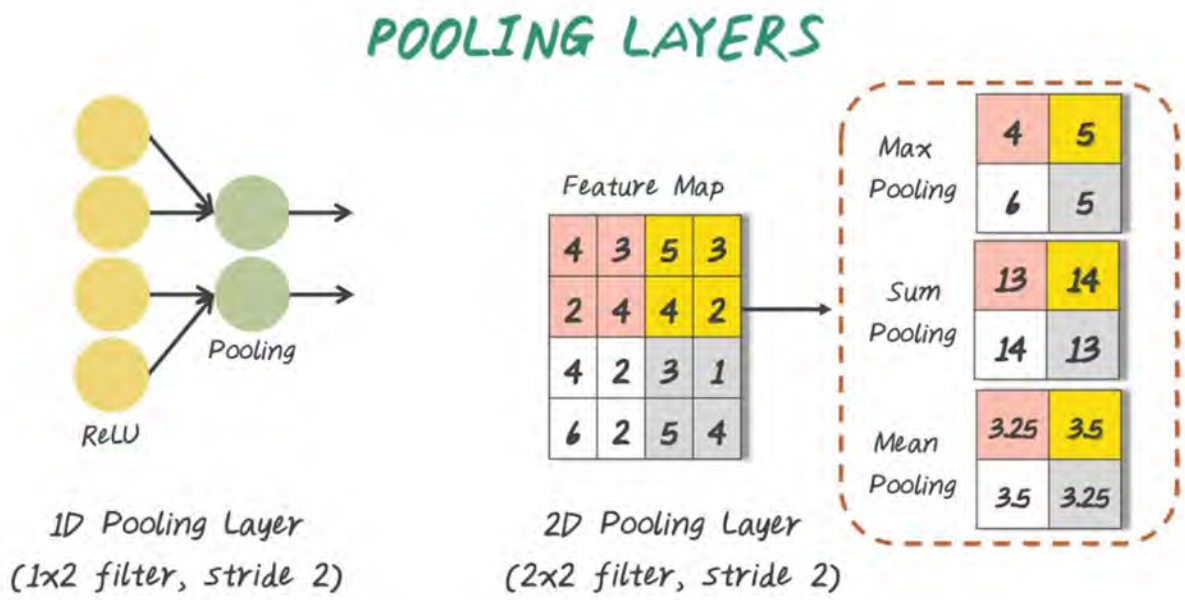
$$(I * K)(ij) = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i+m, j+n)K(m,n)$$



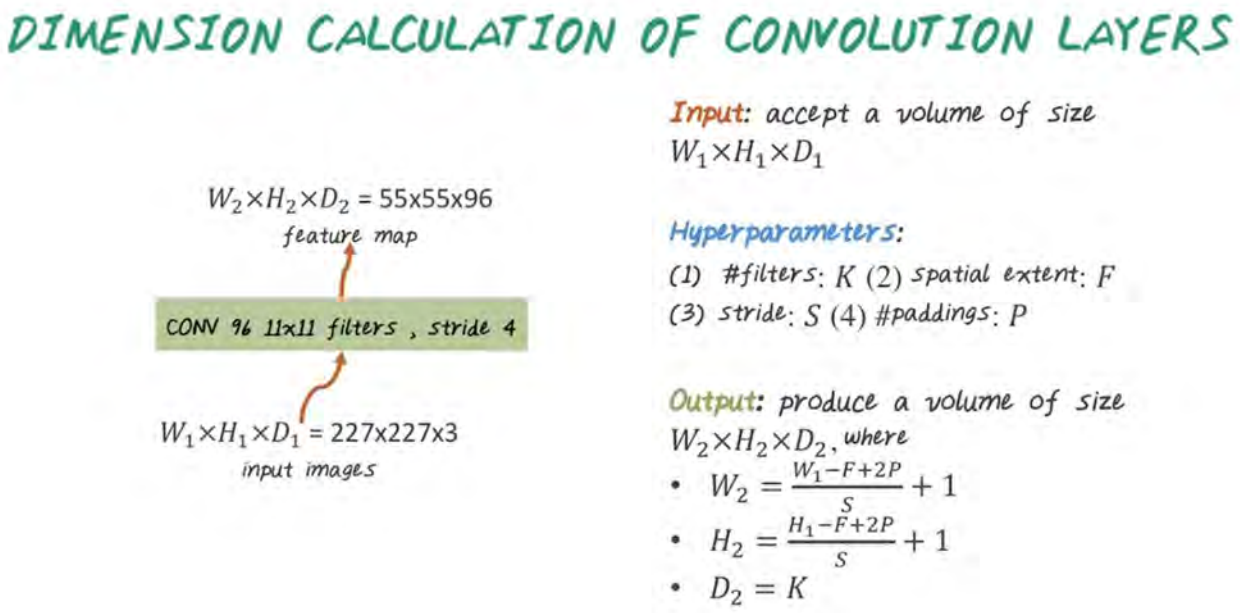
8. [8. Convolution Layers]



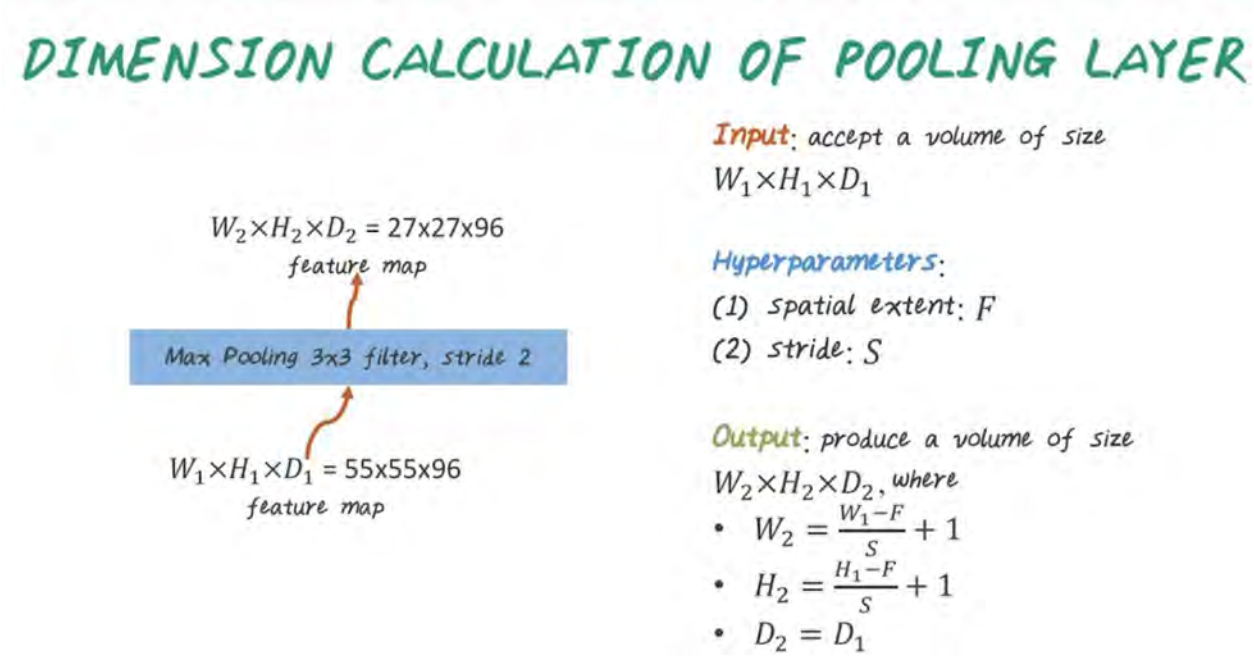
9. [9. Pooling Layers]



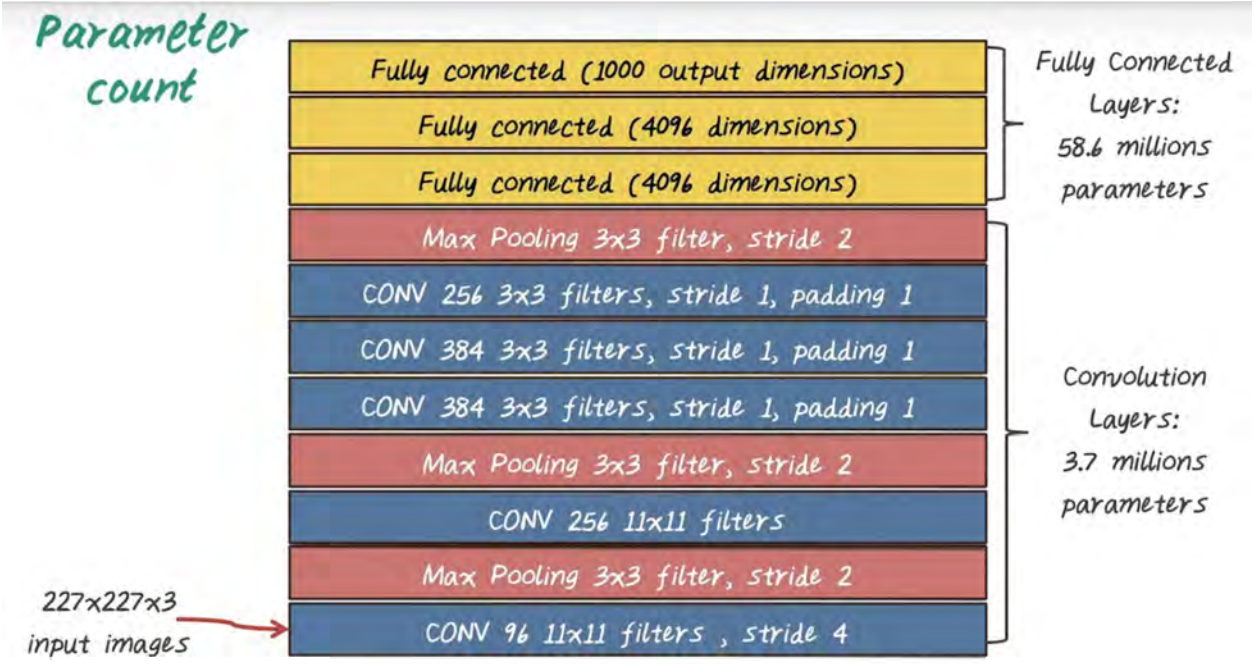
10. [10. Dimension Calculation of Convolution Layers]



11. [11. Dimension Calculation of Pooling Layer]

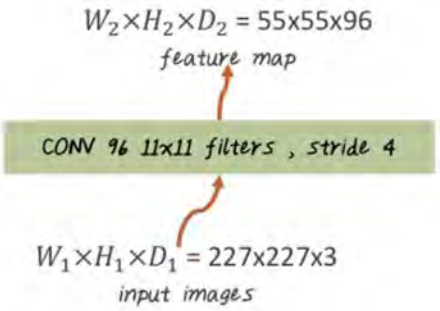


12. 12. Parameter Count



13. 13. Forward Calculation of Convolution

FORWARD CALCULATION OF CONVOLUTION LAYERS



Input: accept a volume of size
$W_1 \times H_1 \times D_1$

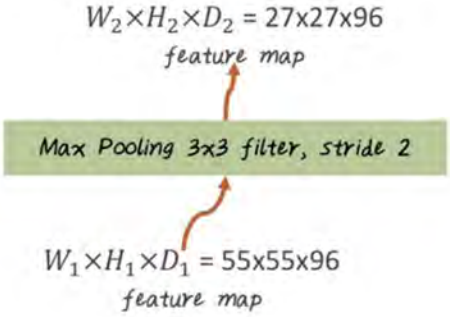
Hyperparameters:
(1) #filters: K (2) spatial extent: F
(3) stride: S (4) #paddings: P

The number of calculations:
$(W_2 \times H_2 \times D_2) \times (F \times F) \times D_1$, where
output size, filter size

- $W_2 = \frac{W_1 - F + 2P}{S} + 1$
- $H_2 = \frac{H_1 - F + 2P}{S} + 1$
- $D_2 = K$

14. 14. Forward Calculation of Pooling Layers

FORWARD CALCULATION OF POOLING LAYERS



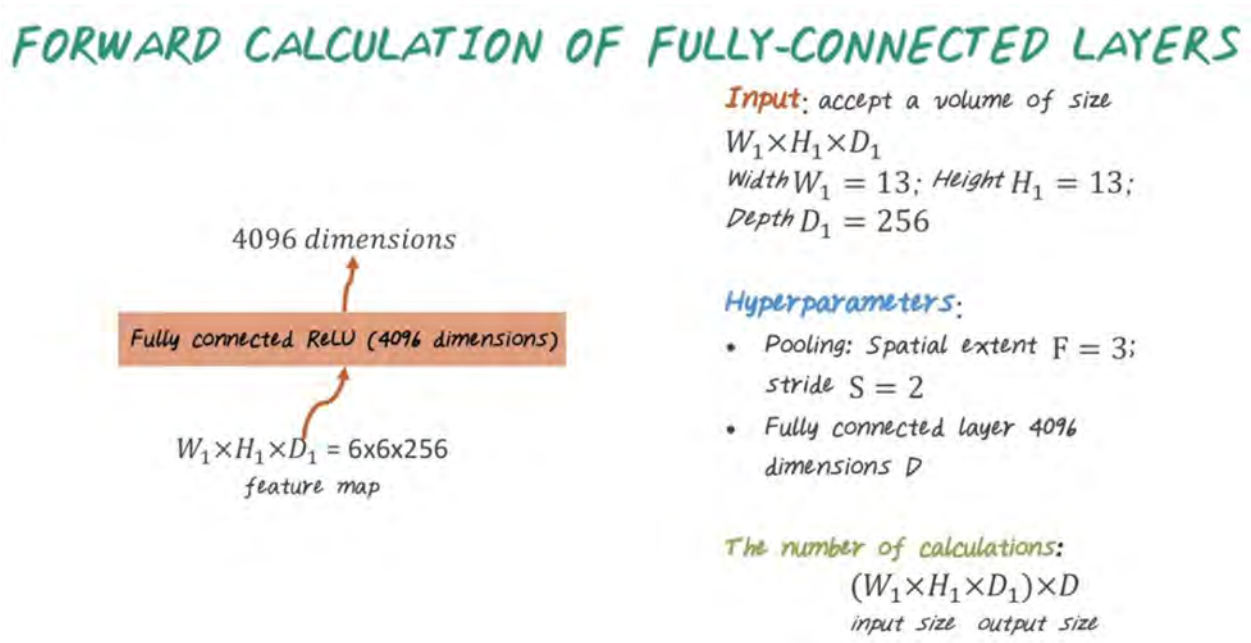
Input: accept a volume of size
$W_1 \times H_1 \times D_1$

Hyperparameters:
(1) spatial extent: F
(3) stride: S

The number of calculations:
$(W_2 \times H_2 \times D_2) \times (F \times F)$, where
output size, filter size

- $W_2 = \frac{W_1 - F}{S} + 1$
- $H_2 = \frac{H_1 - F}{S} + 1$
- $D_2 = D_1$

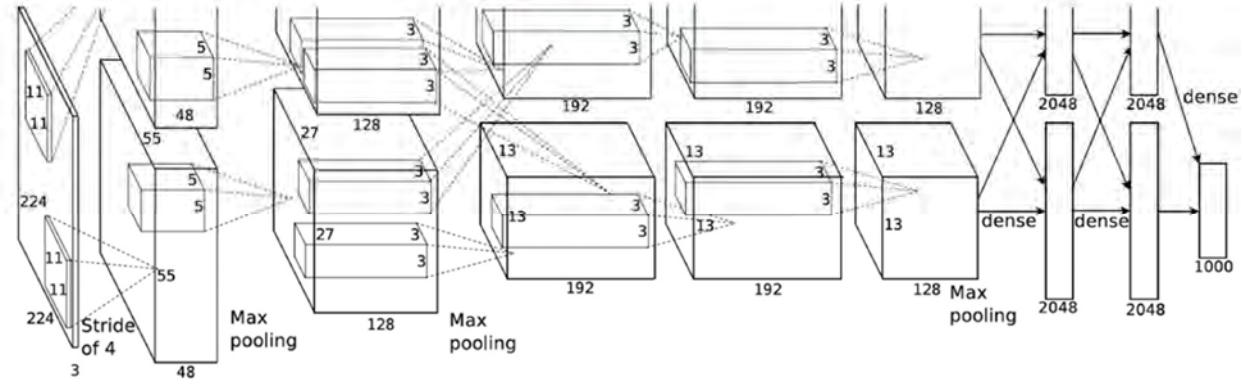
15. 15. Forward Calculation of Fully-Connected Layers



16. [16. Operation Count]

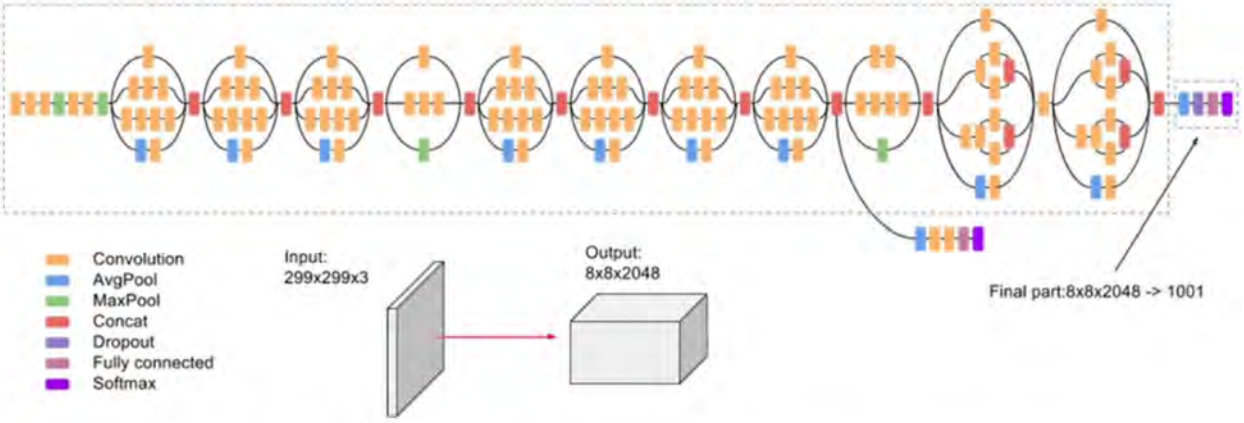
17. [17. CNN Architectures and Healthcare Applications]

AlexNet

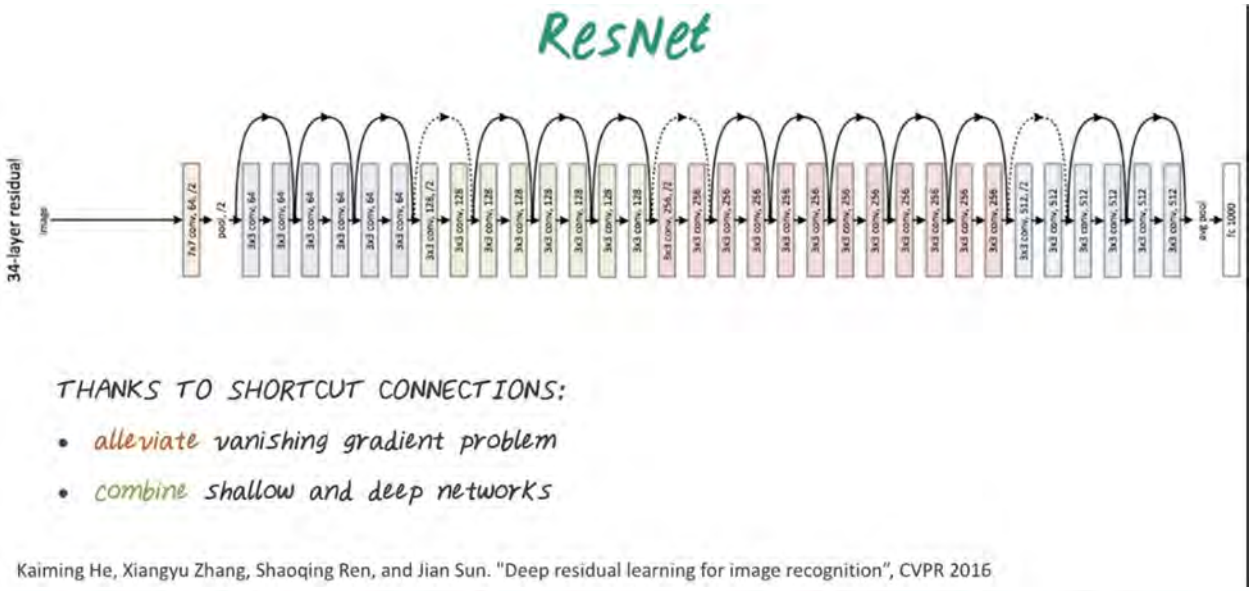


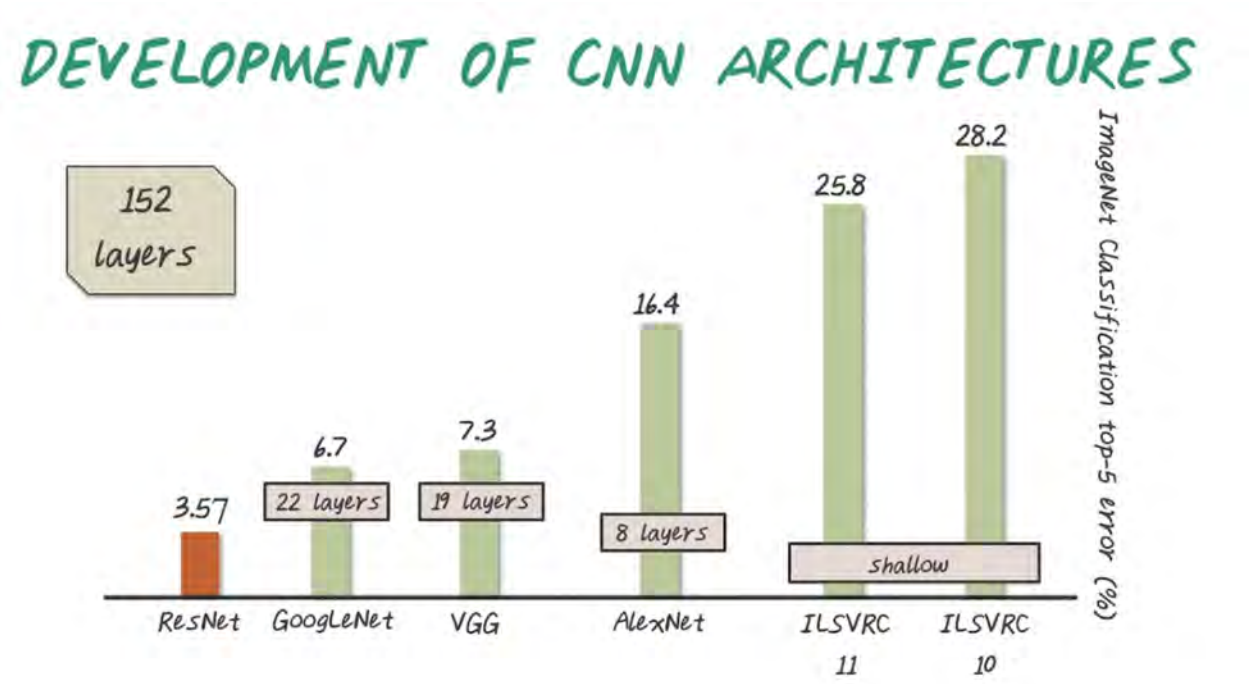
AlexNet: Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks.", NIPS 2012
VGGNet: Karen Simonyan, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition.", ICLR 2015

INCEPTION



Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions.", CVPR 2015





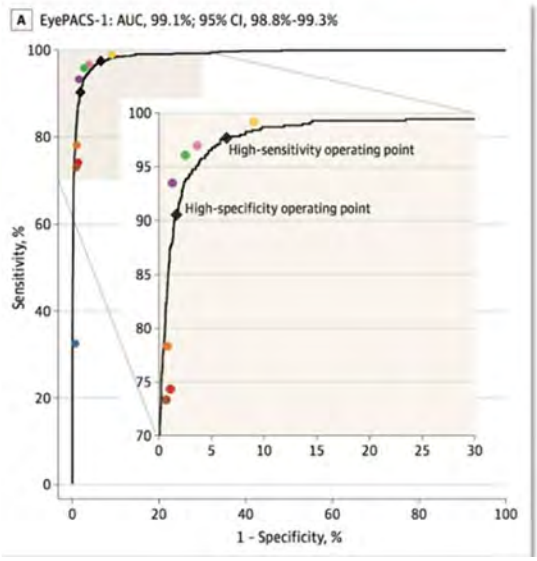
18. [18. Diabetic Retinopathy Diagnosis]

Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs

Gulshan V, Peng L, Coram M, et al. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*. 2016;316(22):2402–2410.

DIABETIC RETINOPATHY DIAGNOSIS

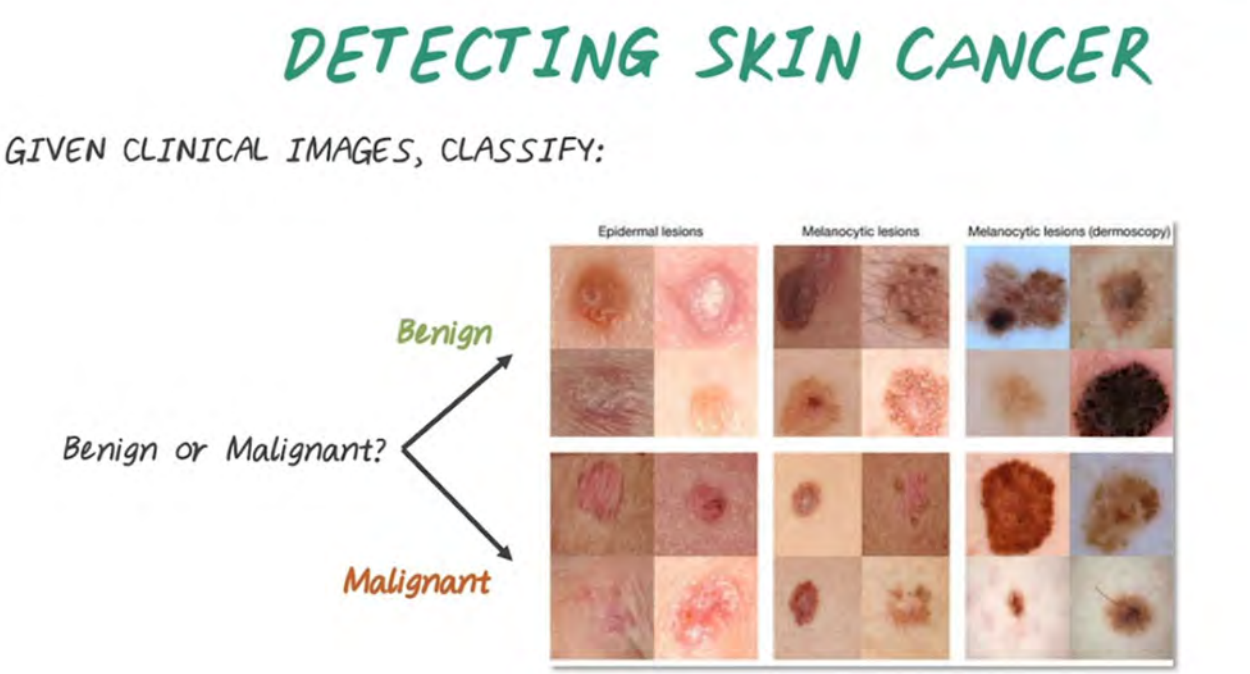
Figure 1. Examples of retinal fundus photographs that are taken to screen for DR. The image on the left is of a healthy retina (A), whereas the image on the right is a retina with referable diabetic retinopathy (B) due a number of hemorrhages (red spots) present.

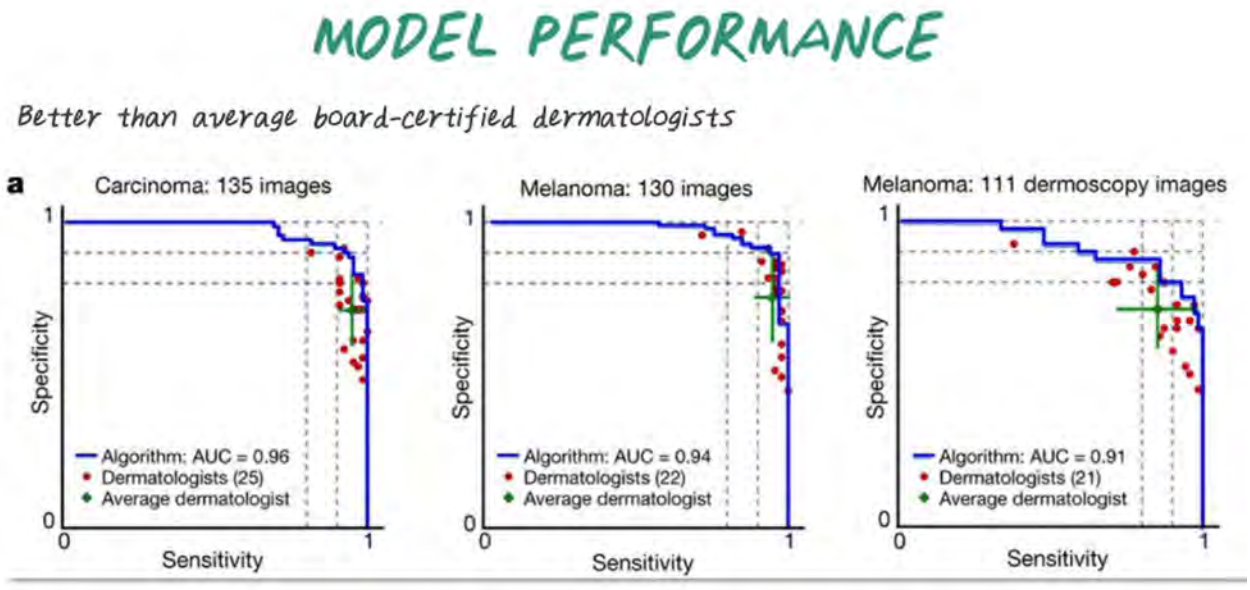


19. [19. Dermatologist Classific]

Dermatologist-level classification of skin cancer with deep neural networks

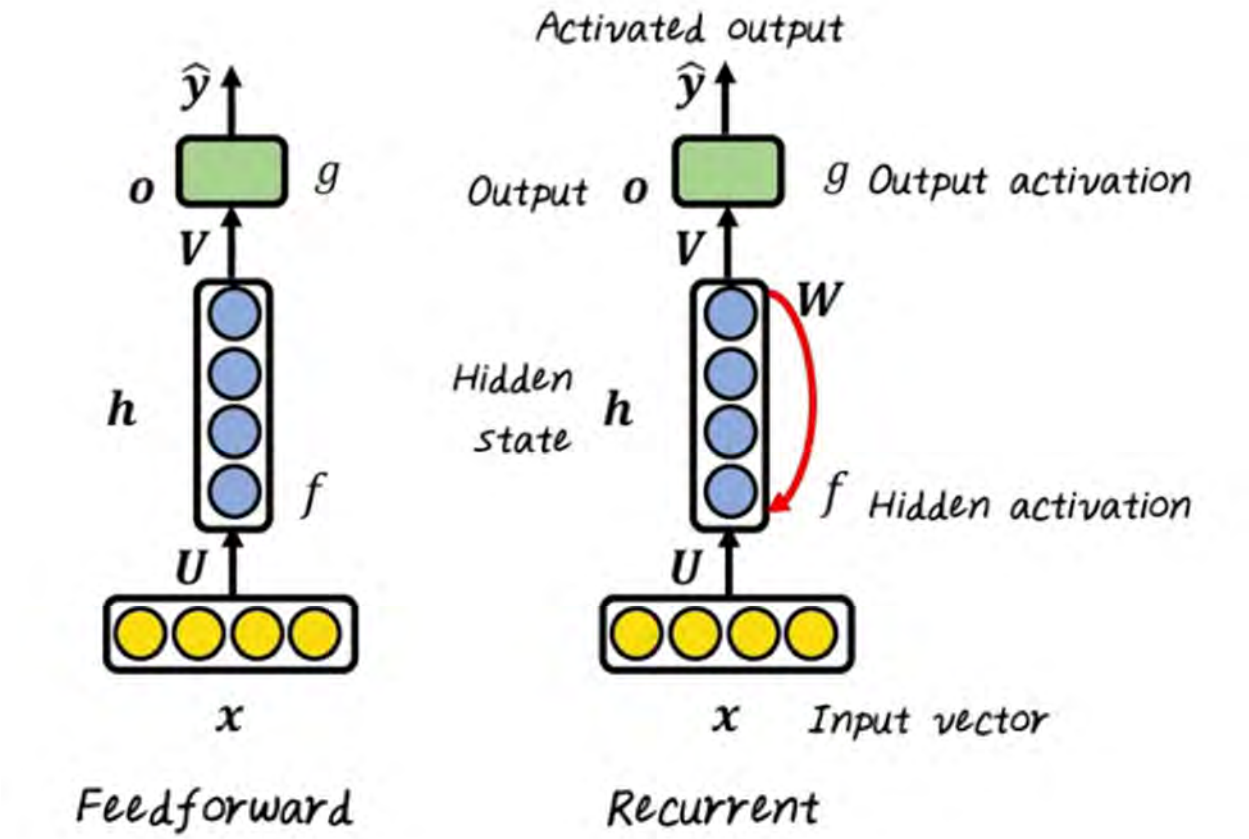
A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau , S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017)



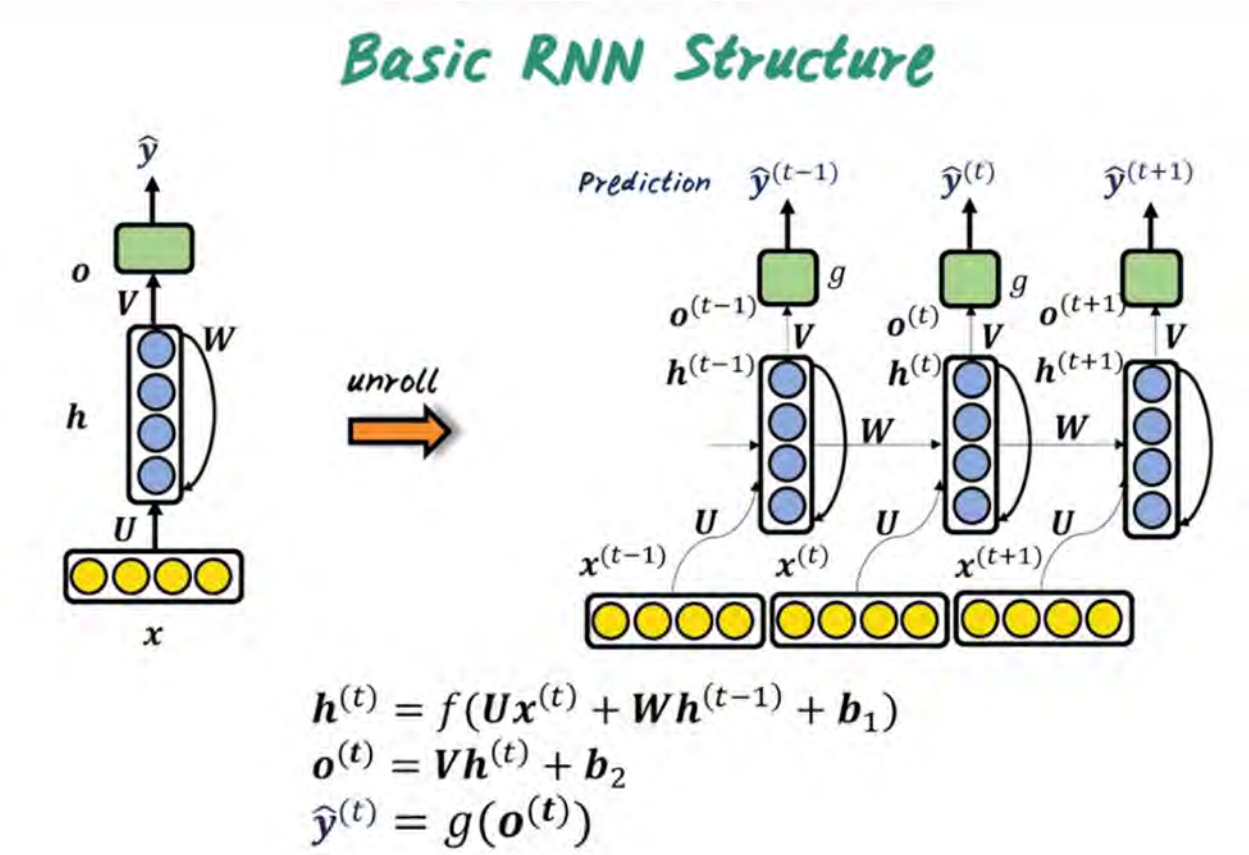


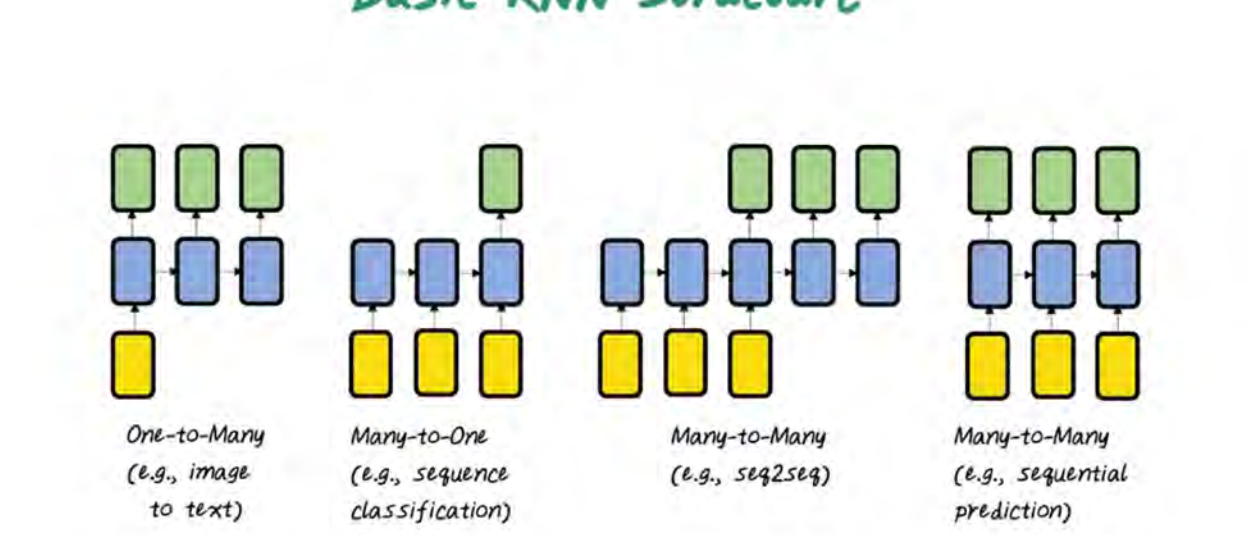
Recurrent Neural Networks (RNN)

1. [1. Basic Concepts of RNN]



2. [2. Basic RNN Structure]

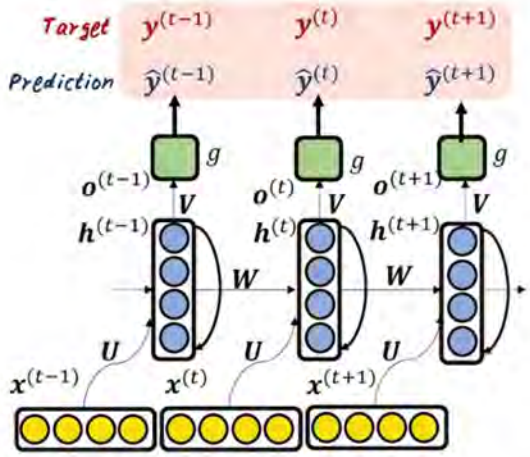




The figure is inspired by page 12 on http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

3. 3. Forward Computation

Forward Computation



$$\boldsymbol{z}^{(t)} = \boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_1$$
$$\boldsymbol{h}^{(t)} = f(\boldsymbol{z}^{(t)})$$
$$\boldsymbol{o}^{(t)} = \boldsymbol{V}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2$$
$$\hat{\boldsymbol{y}}^{(t)} = g(\boldsymbol{o}^{(t)})$$
$$L = \sum_t L^{(t)} = -\sum_t \log p(\boldsymbol{y}^{(t)} | \{\boldsymbol{x}^{(t)}, \cdots, \boldsymbol{x}^{(T)}\})$$

e.g.,

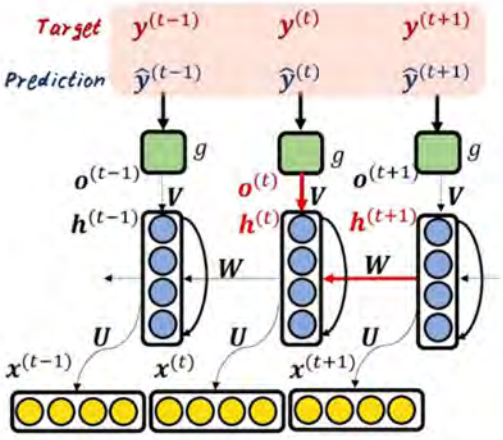
- f is tanh
- g is softmax, which produces a normalized probability over output classes
- $L^{(t)}$ is negative log-likelihood loss

For binary classification

$$L = -\sum_t y^{(t)} \log \hat{y}^{(t)} + (1 - y^{(t)}) \log(1 - \hat{y}^{(t)})$$

4. [4. Backpropagation through time]

Backpropagation through time (BPTT): $\nabla_{h^{(t)}} L$



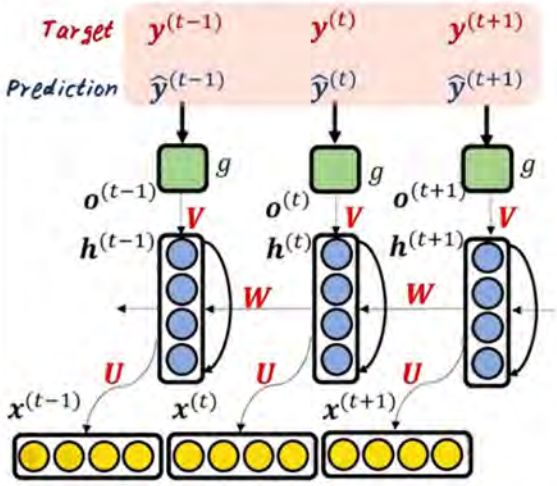
Last time stamp

$$\nabla_{\boldsymbol{h}^{(T)}} L = V^T \nabla_{\boldsymbol{o}^{(T)}} L$$

Other time stamp

$$\nabla_{\boldsymbol{h}^{(t)}} L = \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^T \left(\nabla_{\boldsymbol{h}^{(t+1)}} L\right) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^T \left(\nabla_{\boldsymbol{o}^{(t)}} L\right)$$

BPTT: $\nabla_V L$, $\nabla_W L$, $\nabla_U L$, $\nabla_{b_1} L$, and $\nabla_{b_2} L$



$$\nabla_V L = \sum_t \sum_k \left(\frac{\partial L}{\partial o_k^{(t)}}\right) \nabla_V o_k^{(t)} = \sum_t \left(\nabla_{o^{(t)}} L\right) h^{(t)T}$$

$$\nabla_{b_2} L = \sum_t \left(\nabla_{o^{(t)}} L\right)$$

$$\nabla_W L = \sum_t \sum_j \left(\frac{\partial L}{\partial h_j^{(t)}}\right) \nabla_W h_j^{(t)}$$

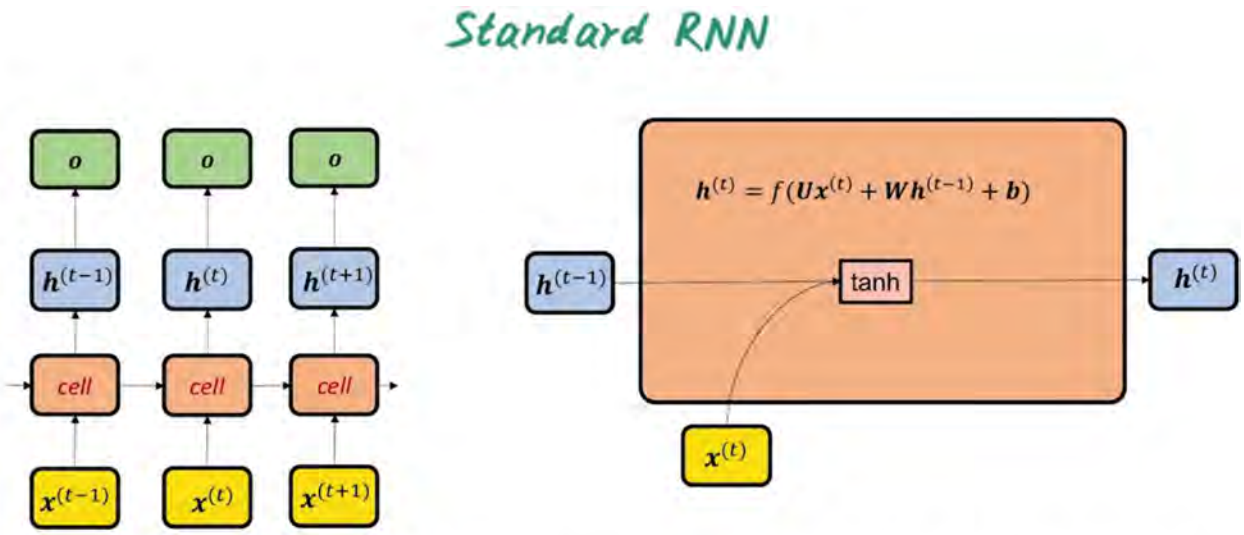
$$= \sum_t diag\left(1 - \left(h^{(t+1)}\right)^2\right) \left(\nabla_{o^{(t)}} L\right) h^{(t-1)T}$$

$$\nabla_U L = \sum_t diag\left(1 - \left(h^{(t+1)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right) x^{(t)T}$$

$$\nabla_{b_1} L = \sum_t diag\left(1 - \left(h^{(t+1)}\right)^2\right) \left(\nabla_{h^{(t)}} L\right)$$

- Gradient can become very small over a long sequence
- Standard RNN will have difficulty to remember state from early history

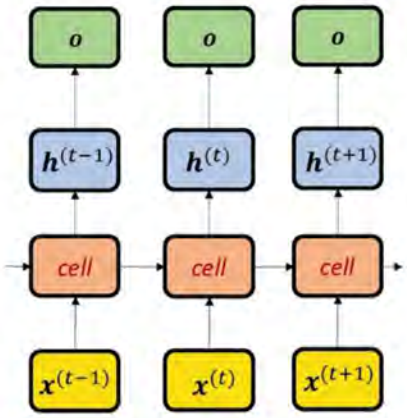
5. 5. Standard RNN



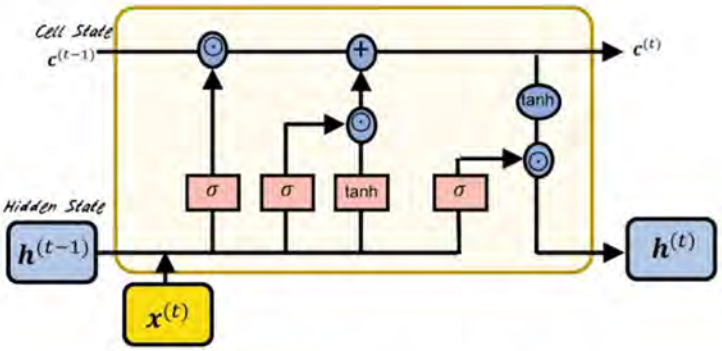
- Standard RNN has a simple computation cell from input x to latent state h

6. [6. Long Short Term Memory Networks]

Long short term memory networks (LSTM)



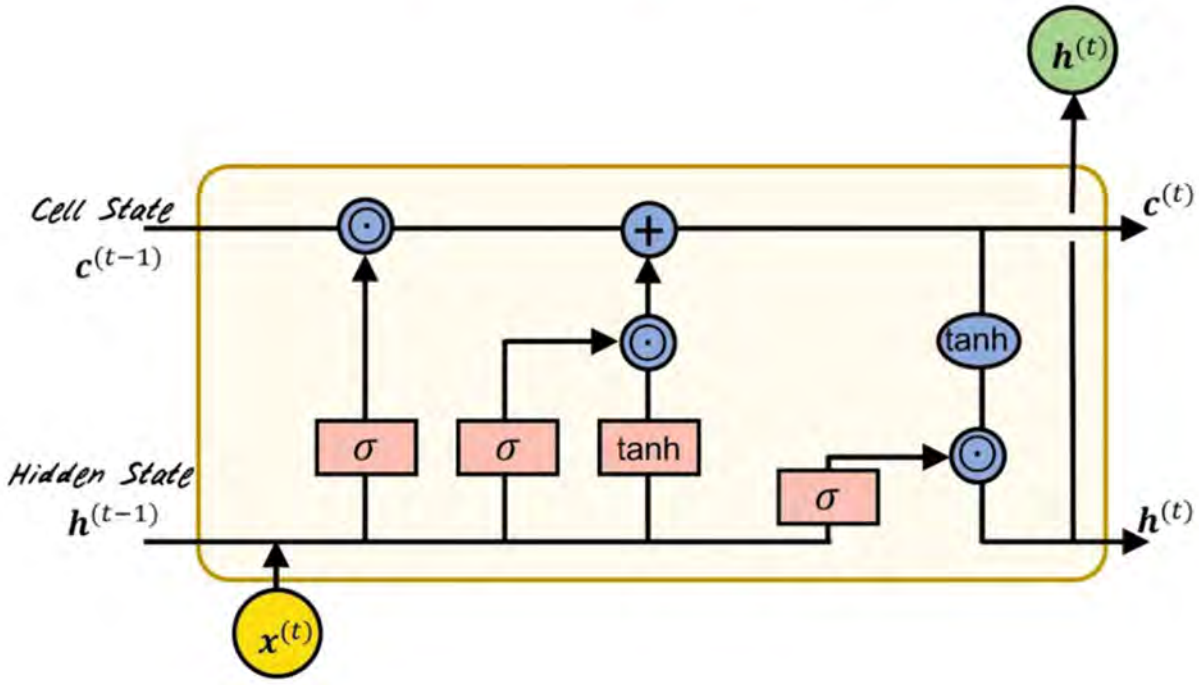
$$h^{(t)} = f(Ux^{(t)} + Wh^{(t-1)} + b)$$



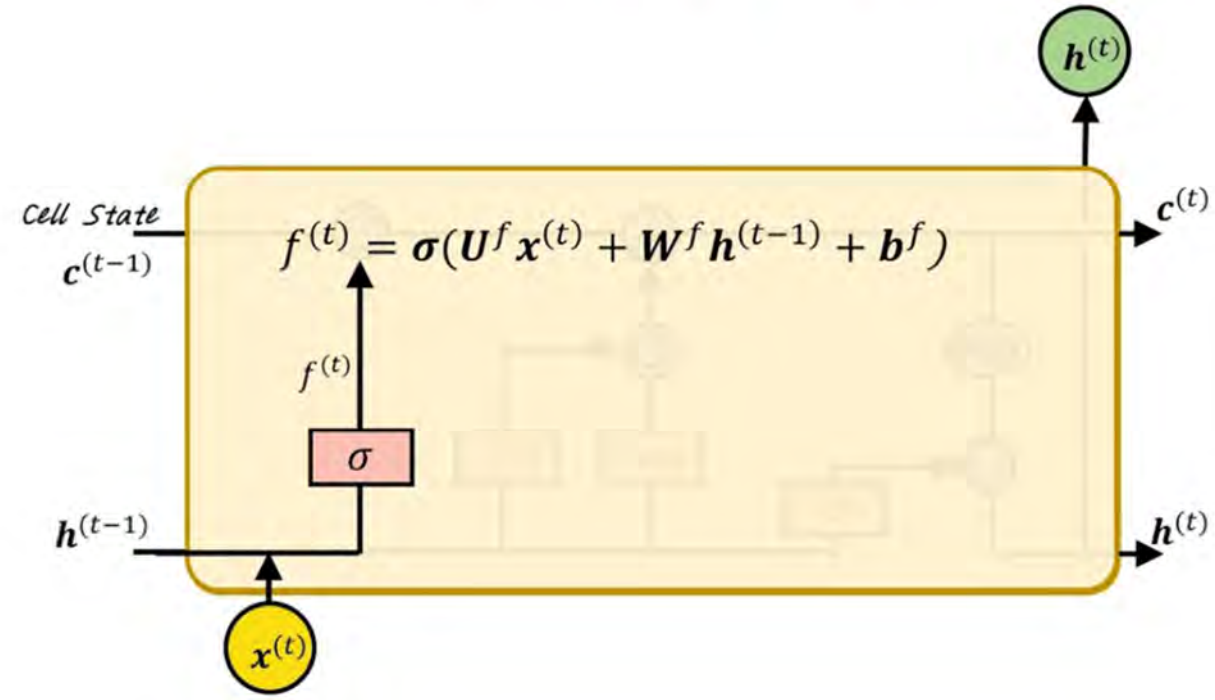
- Standard RNN has a simple computation cell from input x to latent state h
- LSTM provides a more sophisticated cell

,

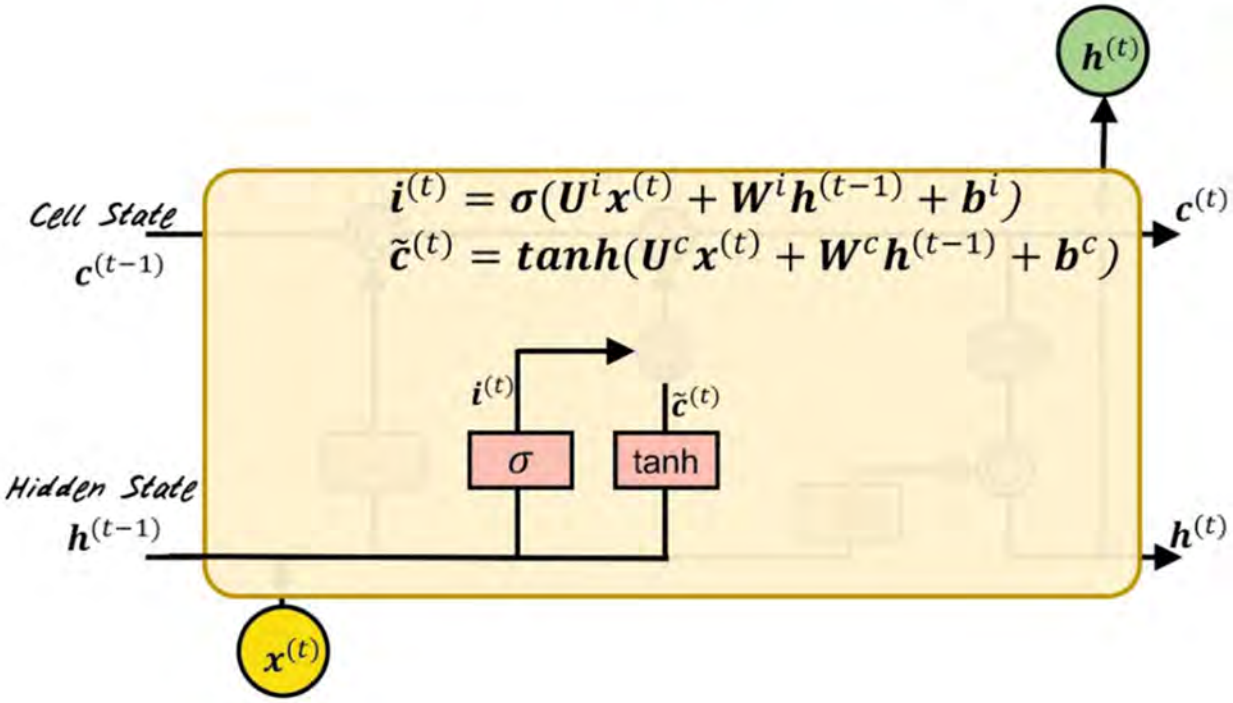
LSTM: Cell Structure



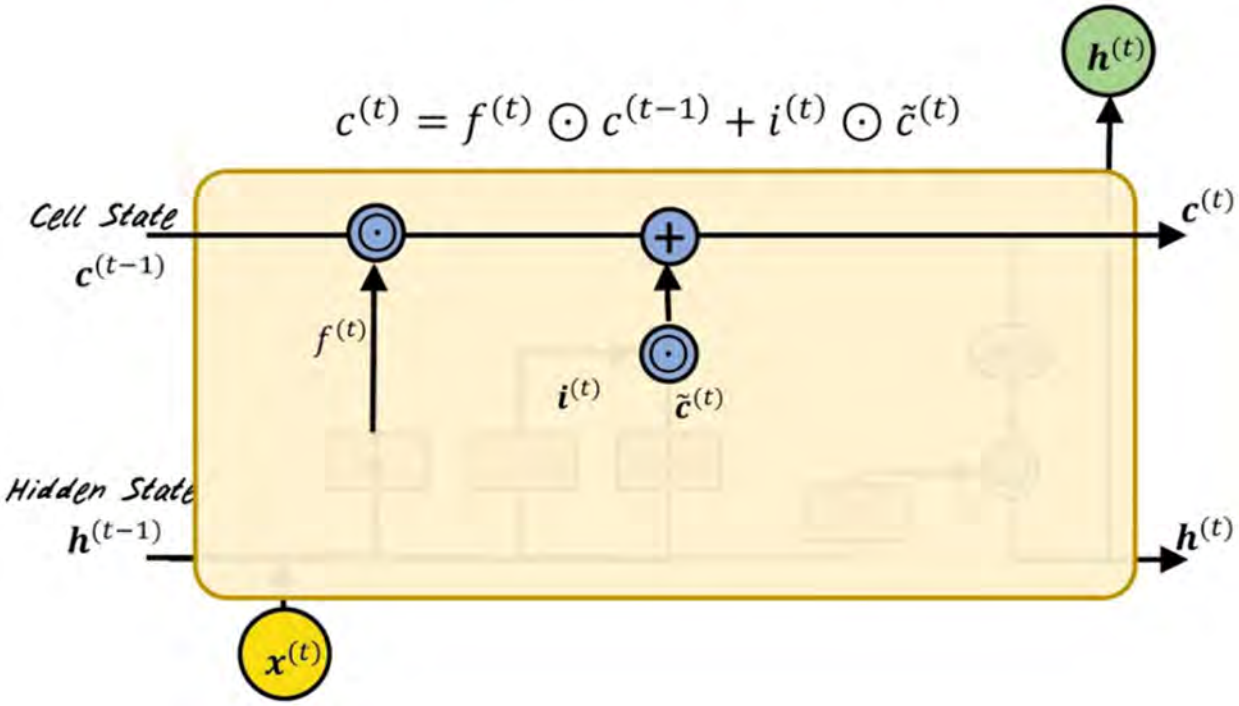
LSTM: Forget Gate



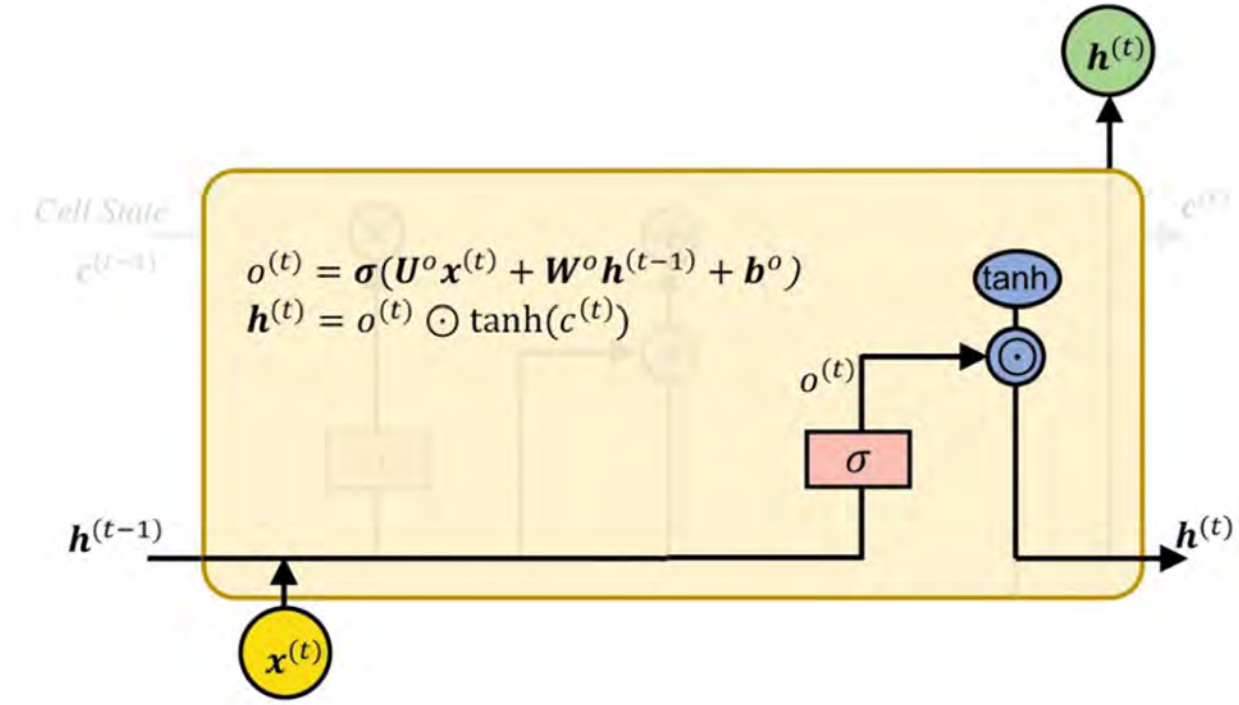
LSTM: Input Gate



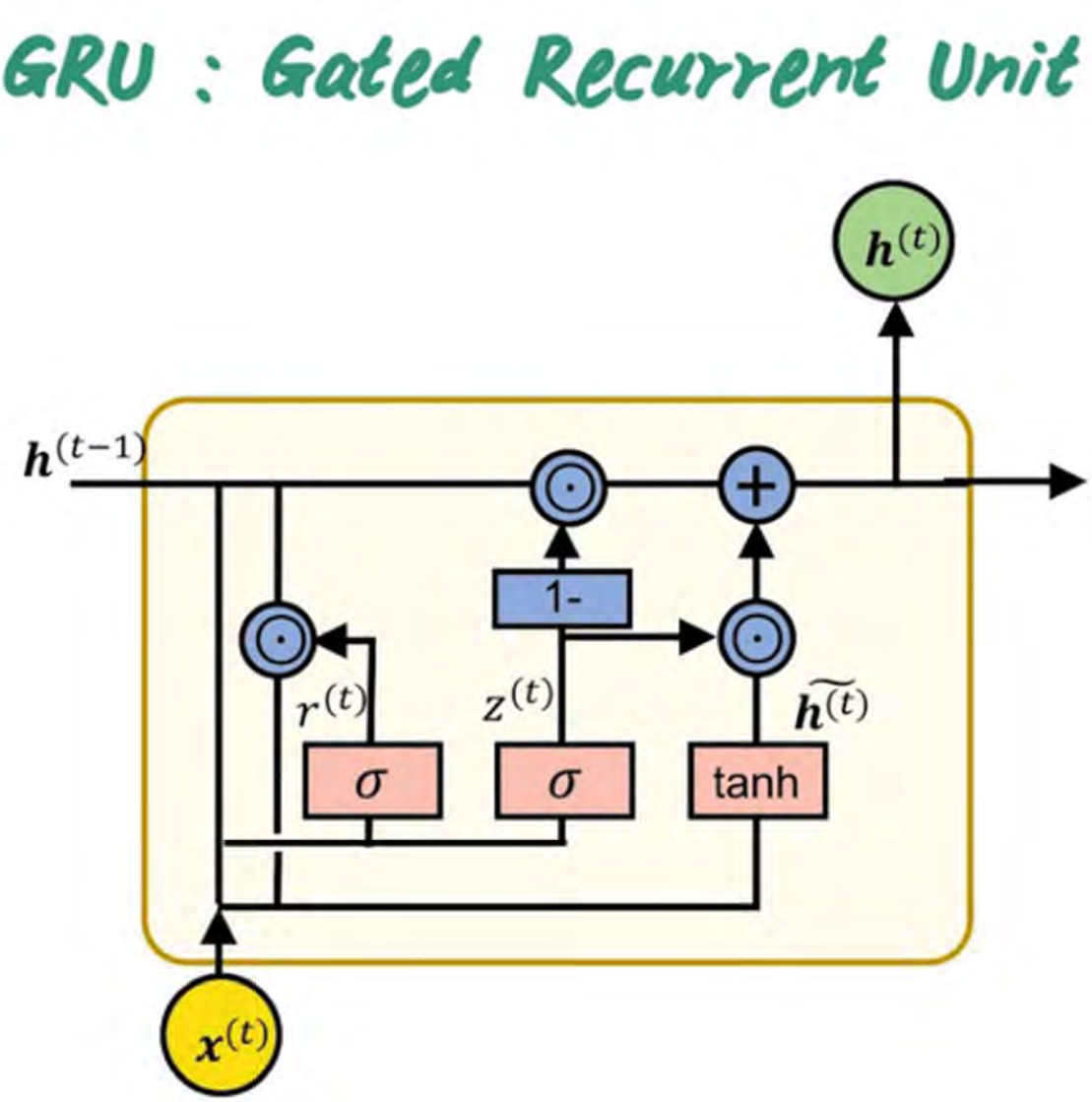
LSTM: Update cell state



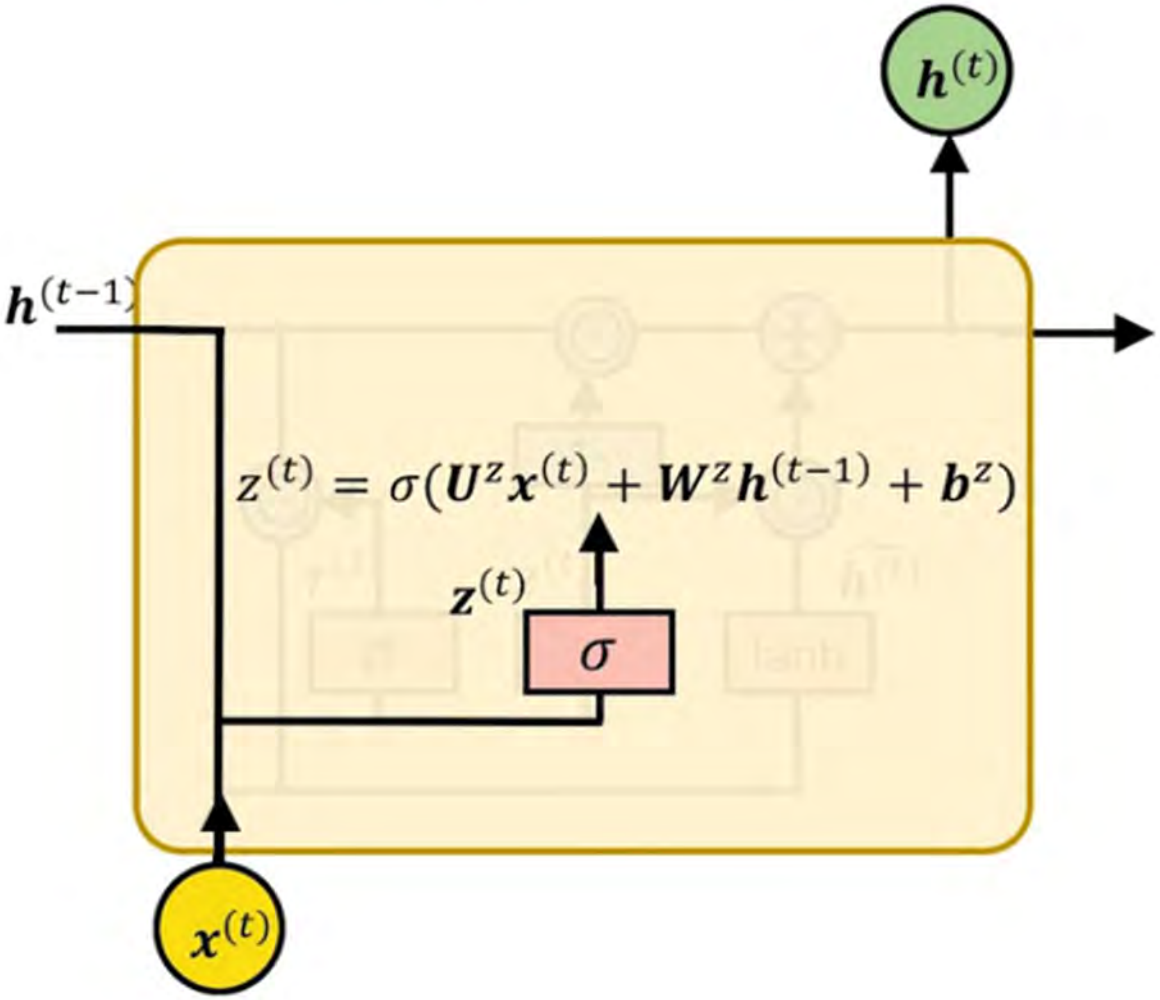
LSTM: Output Gate



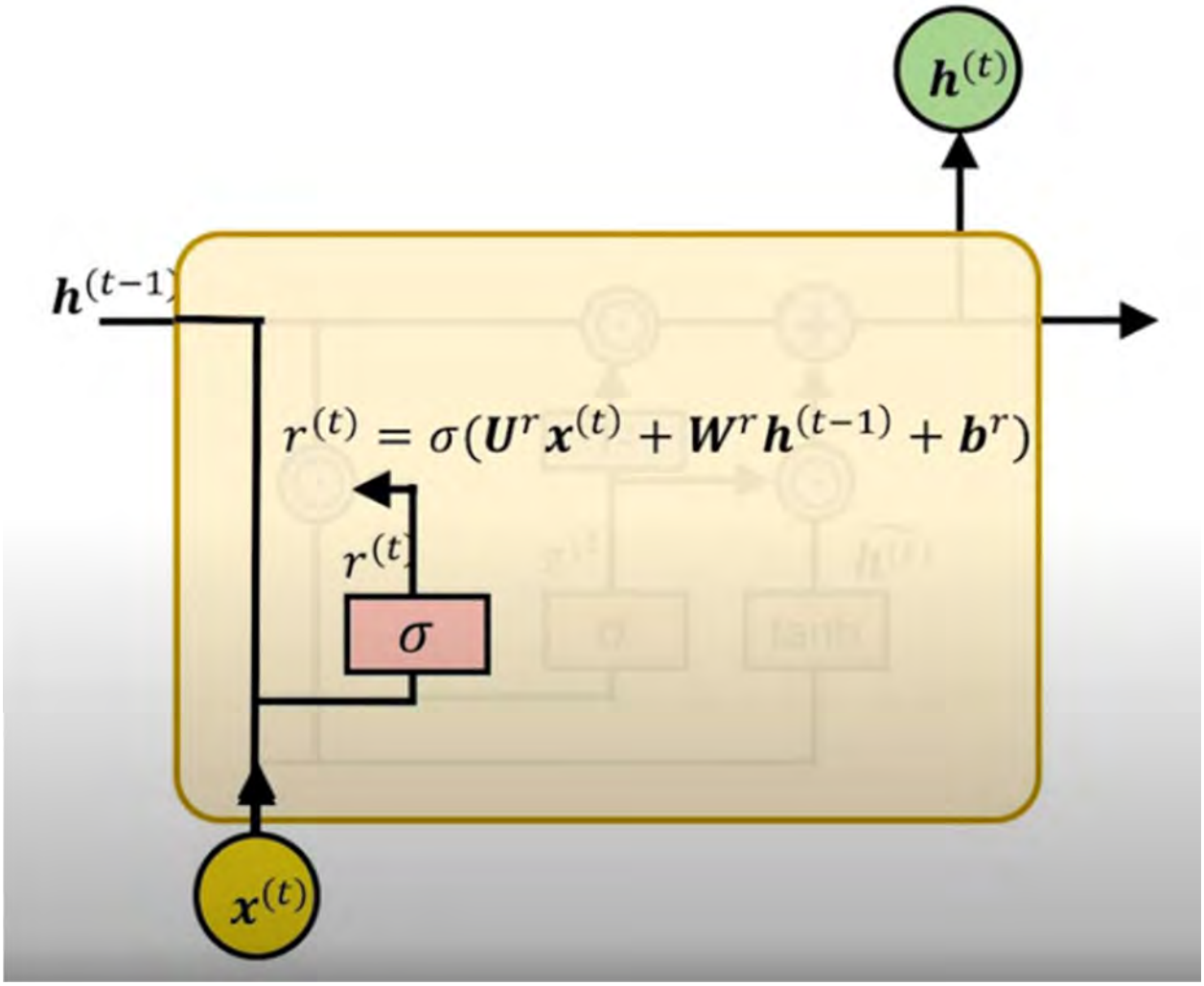
7. 7. Gated Recurred Unit



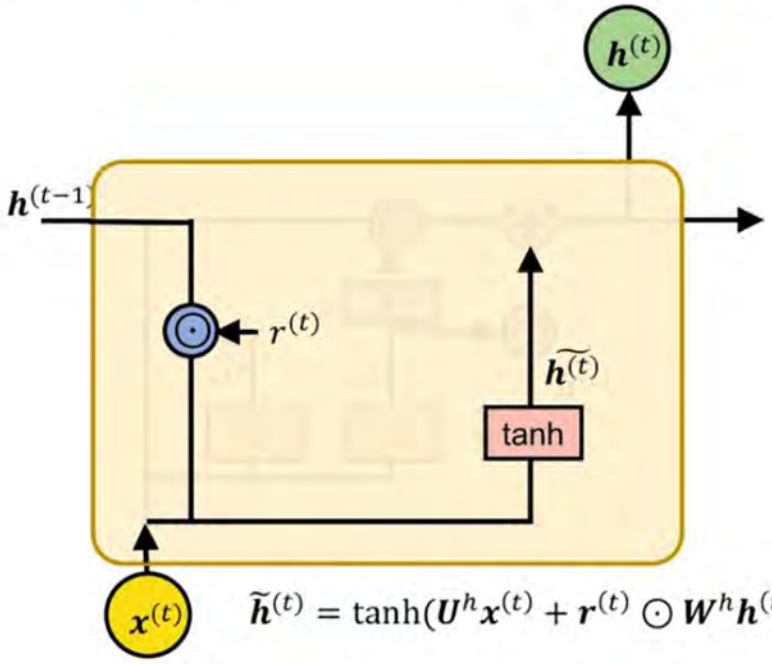
GRU: Update Gate



GRU: Reset Gate



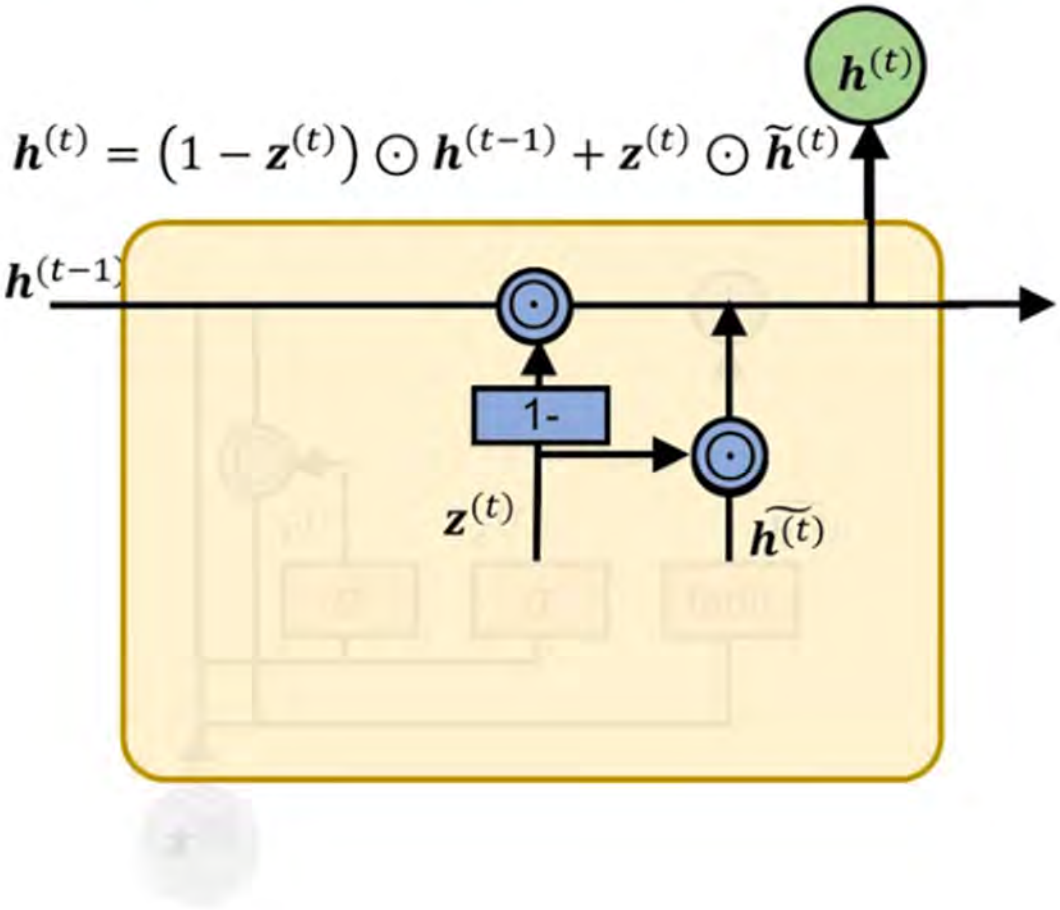
GRU: New information to the hidden state



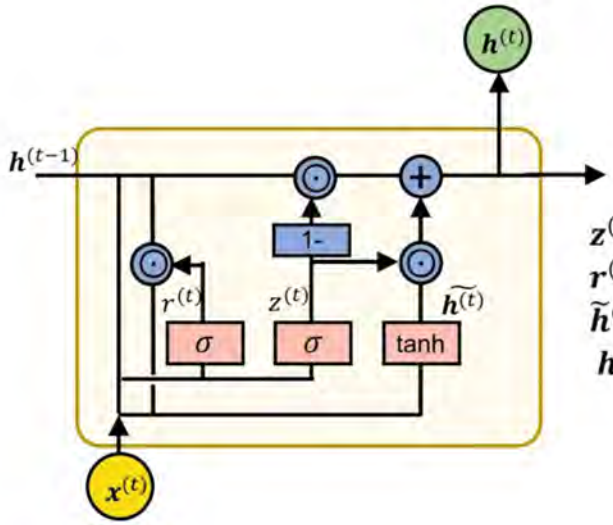
$$\widetilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{U}^h \boldsymbol{x}^{(t)} + \boldsymbol{r}^{(t)} \odot \boldsymbol{W}^h \boldsymbol{h}^{(t-1)} + \boldsymbol{b}^h)$$

GRU: Final New Hidden State

$$h^{(t)} = \left(1 - z^{(t)}\right) \odot h^{(t-1)} + z^{(t)} \odot \tilde{h}^{(t)}$$



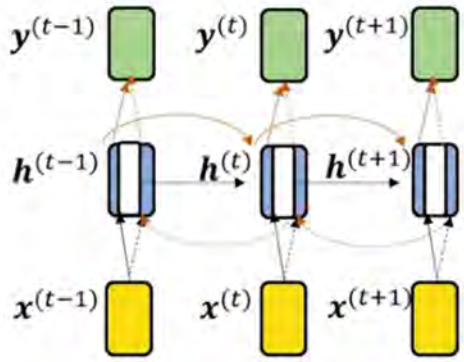
GRU: Summary



$$z^{(t)} = \sigma(U^z x^{(t)} + W^z h^{(t-1)} + b^z)$$
$$r^{(t)} = \sigma(U^r x^{(t)} + W^r h^{(t-1)} + b^r)$$
$$\tilde{h}^{(t)} = \tanh(U^h x^{(t)} + r^{(t)} \odot W^h h^{(t-1)} + b^h)$$
$$h^{(t)} = (1 - z^{(t)}) \odot h^{(t-1)} + z^{(t)} \odot \tilde{h}^{(t)}$$

8. 8. Bidirectional RNN

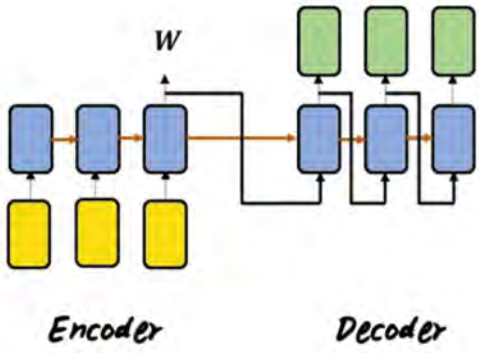
Bidirectional RNN



$$\overrightarrow{h}^t = f(Ux^{(t)} + Wh^{(t-1)} + b_1)$$
$$\overleftarrow{h}^t = f(Ux^{(t)} + Wh^{(t+1)} + b_1)$$
$$y^{(t)} = g(V[\overrightarrow{h}^t; \overleftarrow{h}^t] + b_2)$$

9. 9. Sequence-to-Sequence RNN

Encoder-Decoder Sequence-to-Sequence Model



$$P(\boldsymbol{y}^{(t)}|\boldsymbol{y}^{(t-1)},\cdots,\boldsymbol{y}^{(1)},\boldsymbol{W}) = g(\boldsymbol{h}^{(t)},\boldsymbol{y}^{(t-1)},\boldsymbol{W})$$

10. [10. Healthcare Applications]

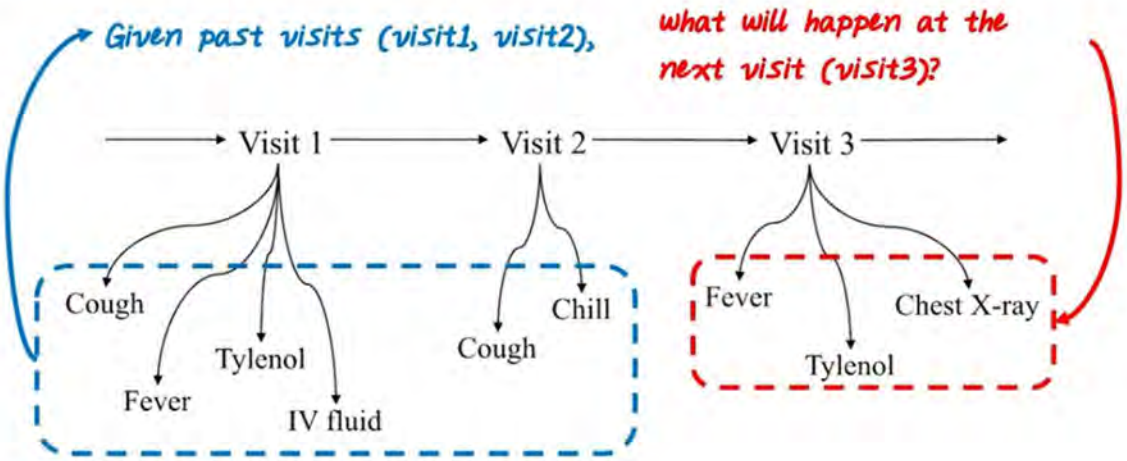
Doctor AI: Predicting Clinical Events via Recurrent Neural Networks

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, Jimeng Sun

Machine Learning for Healthcare Conference, 2016

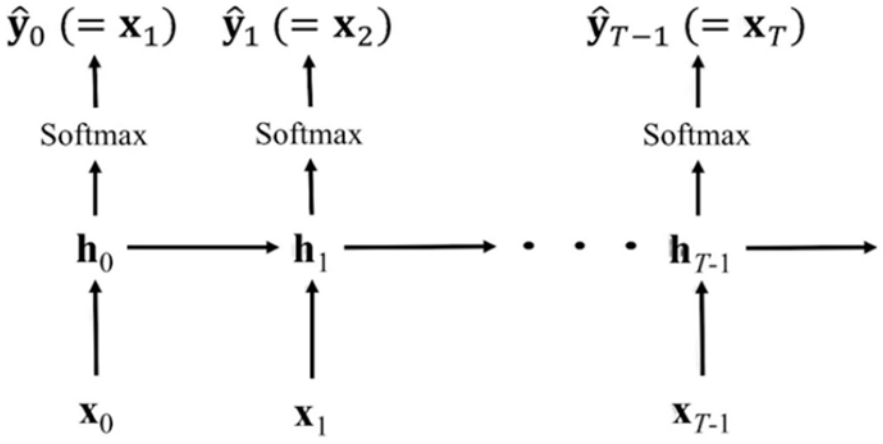
Doctor AI: Task

Sequential Disease Prediction



Doctor AI: Model

- Feed visits into the RNN
  - One visit at each timestep
  - Predict next events at each timestep

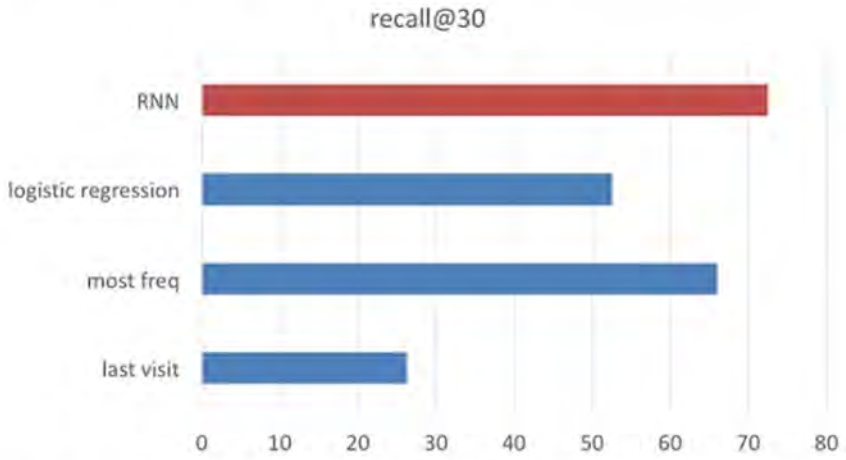


Doctor AI: Data

- Data
  - 260K patients from Sutter Health
  - Patient records over 10 years
  - Input codes
    - Diagnosis codes, medication codes, procedure codes (38,000 codes)
  - Output labels
    - 1,183 diagnosis codes

Doctor AI: Sequential Prediction

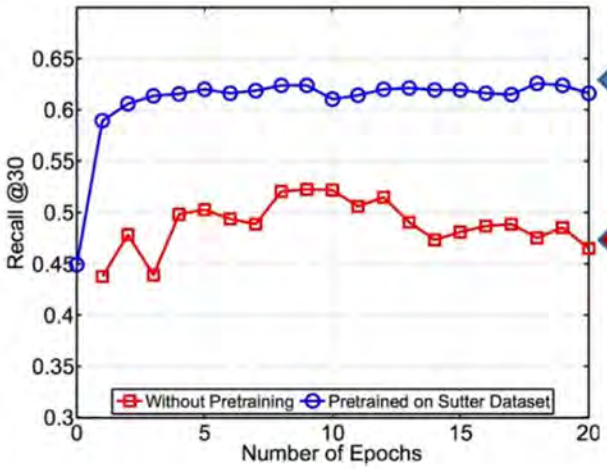
Predicting diagnoses in the next visit



$$\text{top-}k \text{ recall} = \frac{\#\text{ of true positives in the top } k \text{ predictions}}{\#\text{ of true positives}}$$

Doctor AI: Knowledge Transfer

Generalize RNN model from one hospital to another



Perform better when warm start the existing model

Perform worse when cold-start from random initialization