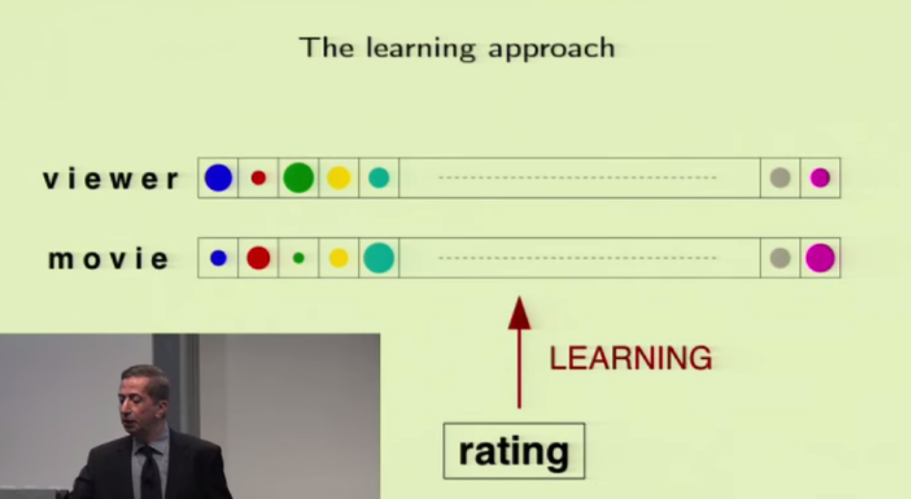
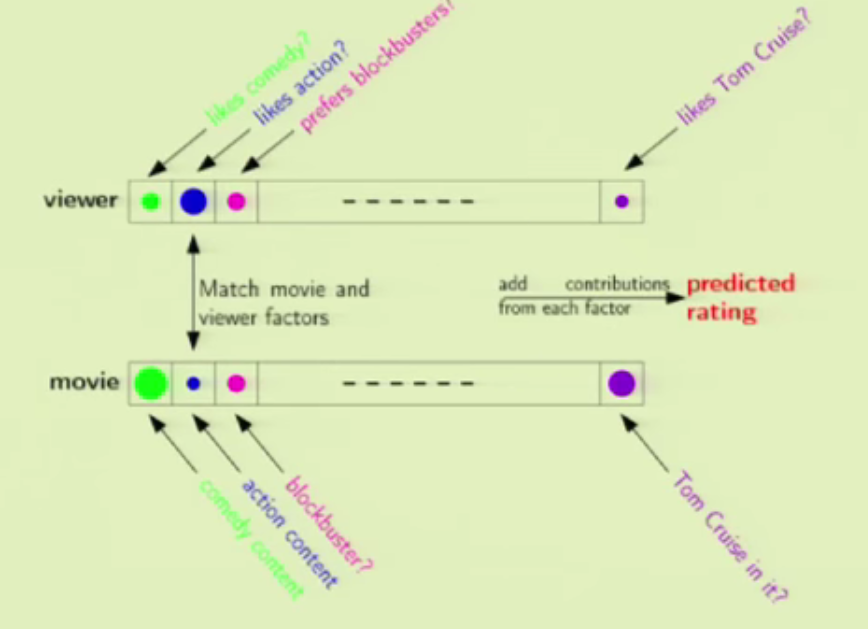
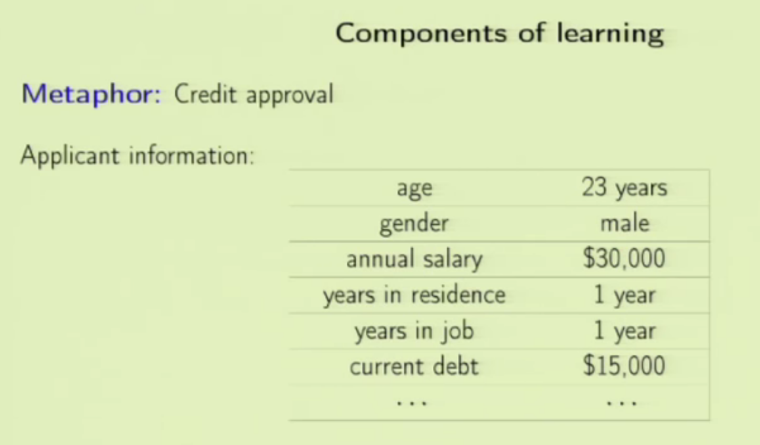
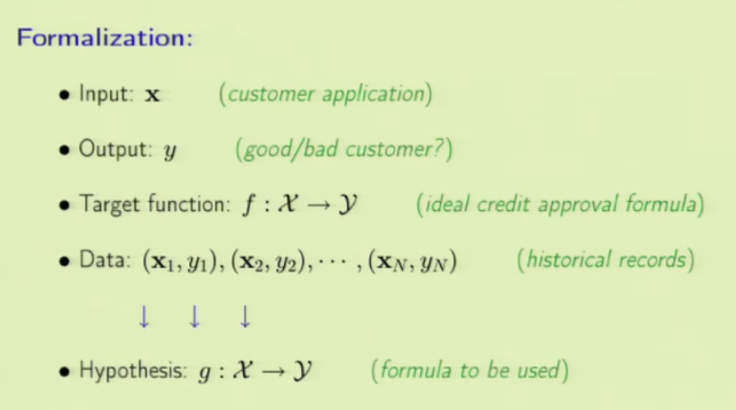
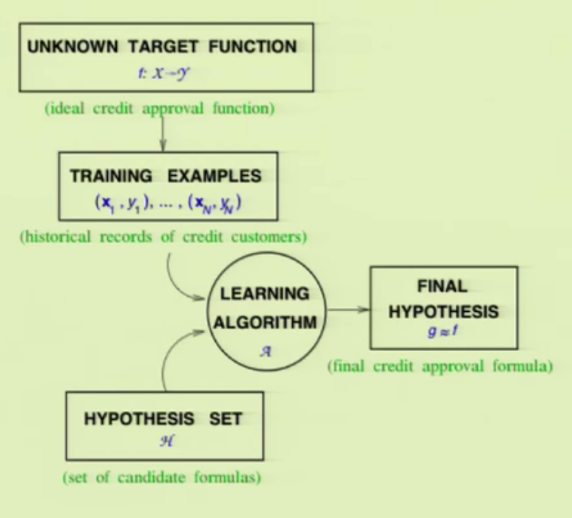
Logo of course, it is a technical figure, wait to know what it is.



The model on right uses machine learning, on the left uses straight forward way which isn’t efficient.

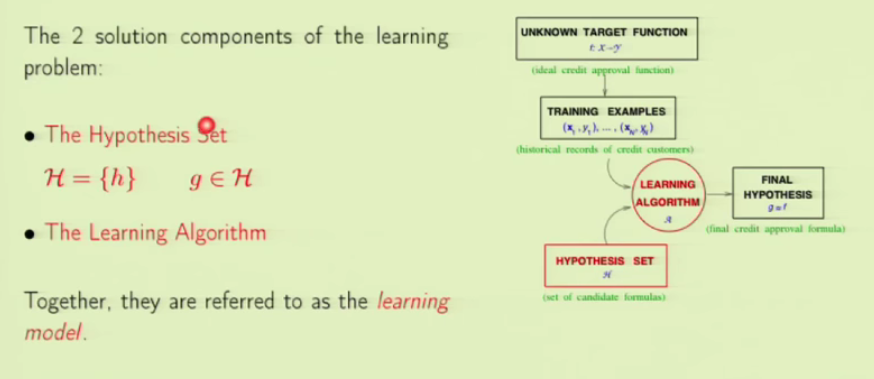
Approve or reject loan request based on historic data.



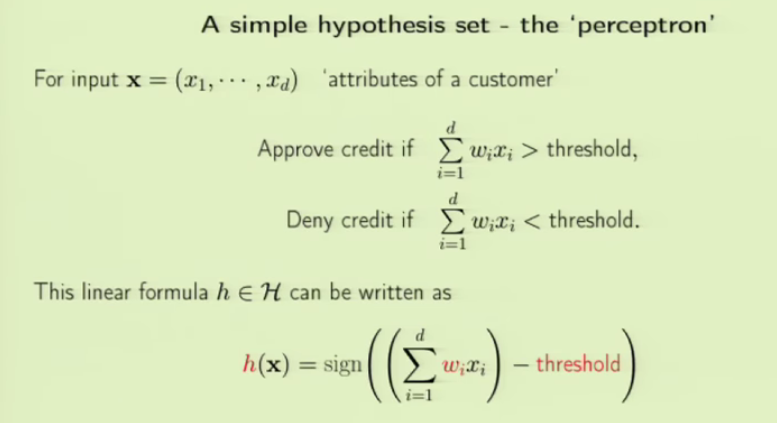
From historical data, we use our approval function to accept or deny loan. We make it better using the historical data so that we will get the usable function g which is our predictor.

From unknown to final, it looks like a normal flow. Why have the hypothesis set?

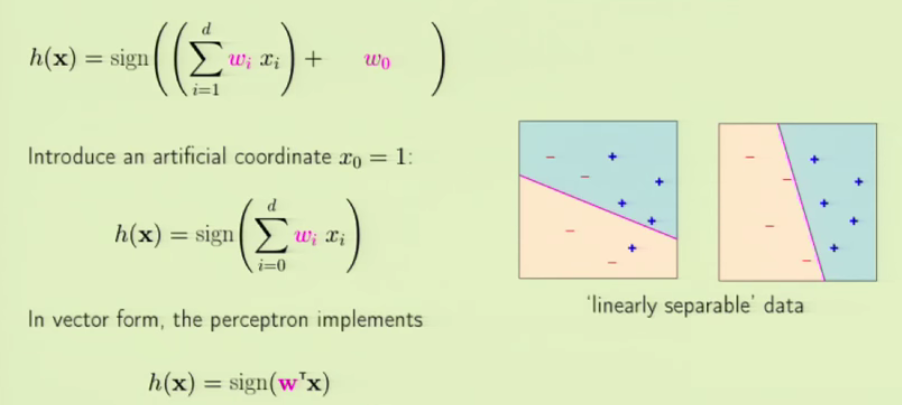
There is no downside with HT and there is an upside. Trying different modes will only make the learning better.



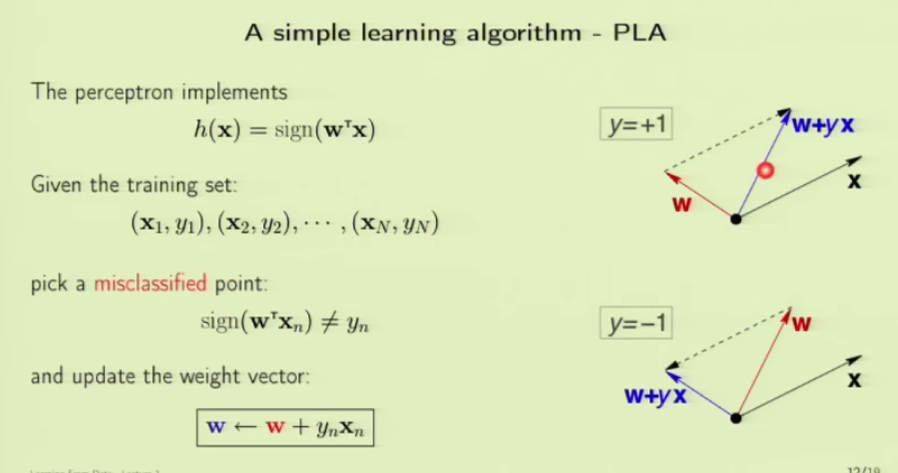
In the above component diagram, what we have control over is learning algo and hypothesis set. So they are the solution components. Learning model consists of both hypotheses set and learning algorithm. As in, perceptron model will be our hypothesis, perceptron learning algo is our learning algo. Neural network + back propagation, support vector machine + quadratic program, every model has these two components.

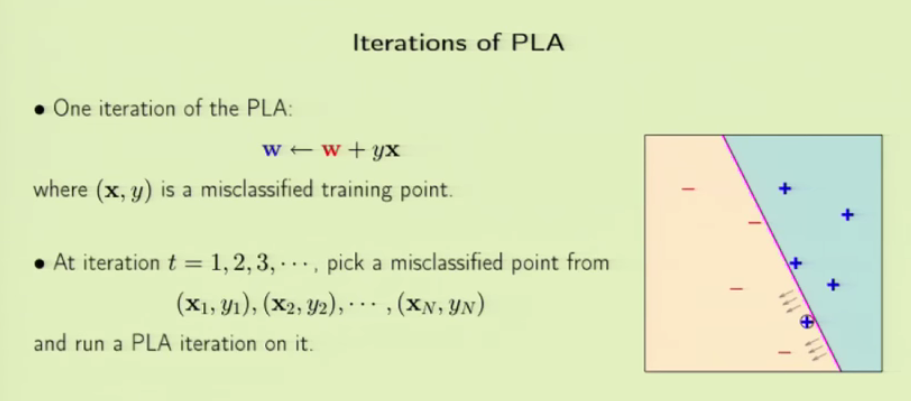
Example model: Perceptron, a simple model

Depending on the type of attribute or feature, a weight is assigned(like for the feature outstanding loan, negative weight is given meaning not desirable for further loan sanction). We calculate the credit score and compare it with threshold score. As we see in the hypothesis, what determines the output is the threshold we chose and the weights assigned to each feature.



When we work on linear data, the model we used will draw a separator between the classes of output. As per our choice of weights and threshold, the separating line changes as seen in the picture. The notation is changed, notice it. WE used w0 instead of threshold. WE took w0 as negative of threshold. W0 = -(threshold). WE do that because we introduce an artificial coordinate x0 = 1. By doing so, our hypothesis function gets simplified.

Once we have the hypothesis, we try to implement it with the learning algorithm using the historical data. What we do is, we feed in the x value to our H, and see what it predicts for a assigned weight W. As per the outcome, we try to adjust the weights W and then predict again. This way we keep doing it, until the model H predicts well. If model predicts -1 instead of 1, we multiply W with actual output y and add it to weight W to form new adjusted W. This is repeated for some iterations.

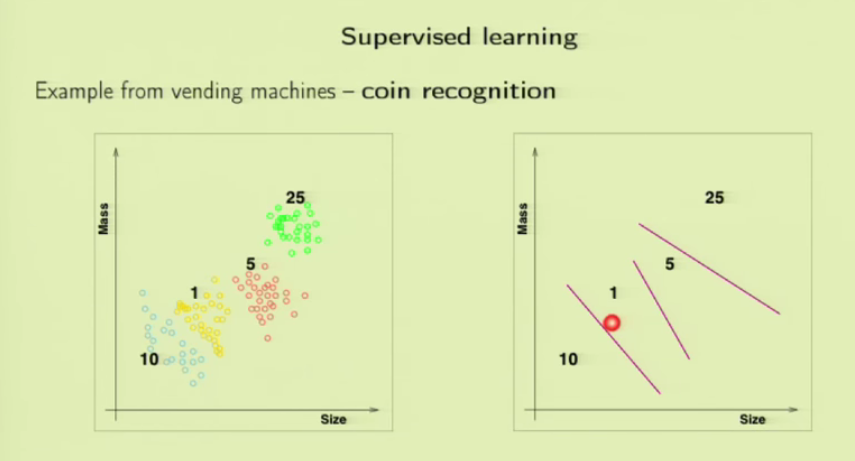
There will be different iterations in PLA- perceptrons learning algorithm. We see that there is one misclassified sample. WE iterate H again for that sample. SO this way, using data of existing and old customer details, we predict whether or not to approve a loan of a new applicant.

This is only one type of learning, there are several others.

Using a set of observations to uncover an underlying process = learning

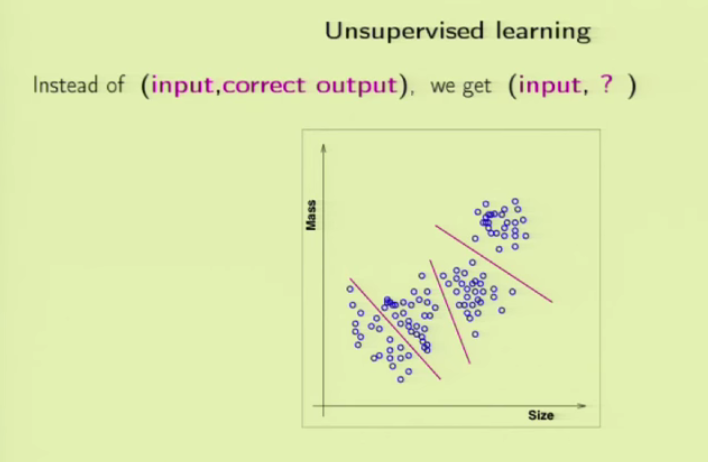
There are many variations in it.

In Statistics, they use probability distribution as the underlying process. The observations are samples generated from that distribution. We take the samples and predict what the probability distribution is.

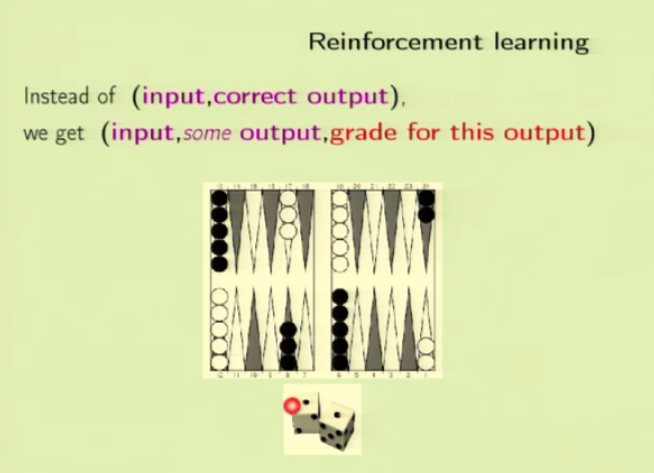
Supervised learning, unsupervised learning and reinforcement learning are three types of learning.

In supervised learning, we have kind of a supervisor which helps us with the problem solving.

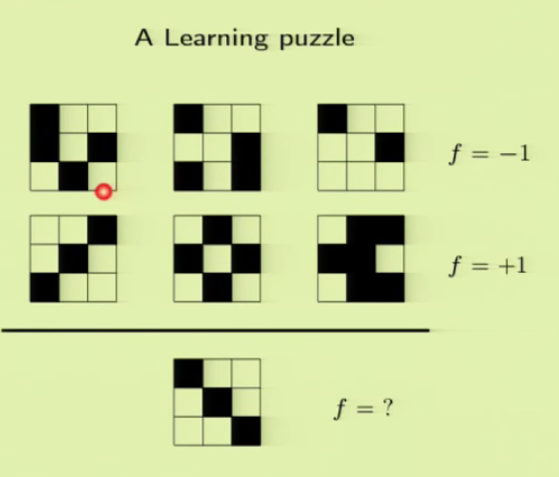
Here we have already classified information of coins. We make prediction of new coin using supervised learning.



Incase of unsupervised learning, there wont be any supervisor, that is no output value or label of each sample. We simply have data from which we should draw some outcome. WE don’t know what categories they belong to. The unlabelled data will be plotted into clusters and can be separated. Unsupervised learning is like trying to learn a new language. Listening to a radio station in an unknown language, you will form some abstract and high level representation of the language.



In reinforcement learning, like in unsupervised learning, we are not given the exact output of each sample. Instead, you are given the information about some output and the grade of its output. It is interesting because it is in a way close to how humans learn. Learn by doing and improving from good or bad actions. Think of a toddler who touches a hot cup. The first time, on touching the cup the kid realizes its not a good action and learns from it. The pain here helps the toddler to learn. One of the most important application of RL is in games. The target function in this case, in a game is how good a move is given a state. The computer will learn from each move and remember which one is a good move and which one isn’t. That wat it gets smarter.

 We have 2 samples for which we know the output. Predict for the unlabelled data. It is said that we don’t know the target function. So it can be 1 or -1. Lets say, if our top left square is 1(black) then output is -1. Frm this we can predict that out unlabelled data is -1 since its top left cell is -1(black). On the other hand we might learn that its 1 if there is symmetry which exists in data sample 2. Our new data contains both features so it can be 1 and -1.

Even if this approach works, it will be memorizing and not learning. Memory cant be used outside the data samples on new data. So the problem we have here is that the target function is unknown. So is learning not possible with unknown target function? It is possible.

**Q & A Session:**

How to know if data is linear or not? There are algorithms and techniques. There is a technique to deal with both. In case of nonlinear data, there are techniques to take the nonlinear data and map them to linear data. So how to determine if data is linearly separable or not? One way is to get started with an assumption that it is not linearly separable. The perceptron algorithm with a minor change called Puppet rule(?) can handle nonlinear data as well. But if linear model is applied to nonlinear data, the outcomes are going to be bad.

Perceptron is taught because its simple. Its not efficient.

Look at the data – determine its linearity and then model it 🡪 NO NO, not possible

Given some data, try to use the metadata, know about the features but not the actual data at STEP 0.

The hypothesis set can be anything – continuous, discrete

How do we decide whether or not to approve a loan? The data contains different scenarios. Like there might be data about applicants who got approved and considered to be good customers. What about those who get rejected? There wont be much details as we have in the first case. This way, some kind of bias is included in the dataset. Though this does exist, the data available from approved applicants and old customers can set optimal boundaries for the model to make correct predictions. More about this will be discussed during Sampling bias lecture.

How much data is ENOUGH? Will be answered in the theoretical lectures. But in practical you don’t always get the data you need. You should work on whats given.

How do you decide the size of the hypothesis set?

3 Qs from the same student

For ML the bottleneck has never been computational time. It is to generalise the data outside the given data set for new data.

Perceptron algo is bad computationally but good in generalizing.

Distinguish between hypothesis test and learning algorithm. In general LA has the form of reducing the error. In perceptron the LA tries to minimise the misclassification or error using the update rule.

Are there problems for which we have a lot of data, but still cant be learned?

We can use the 3 points which are the essence of machine learning.

The choice of hypothesis set: selection, reduction, Bayesian principle