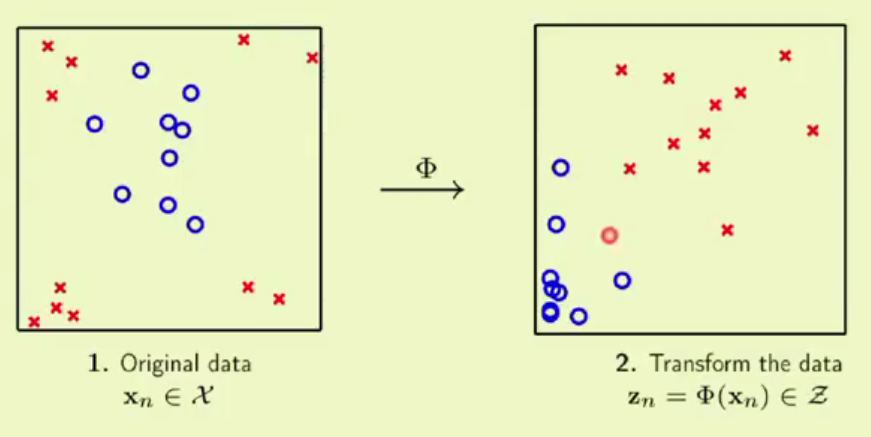
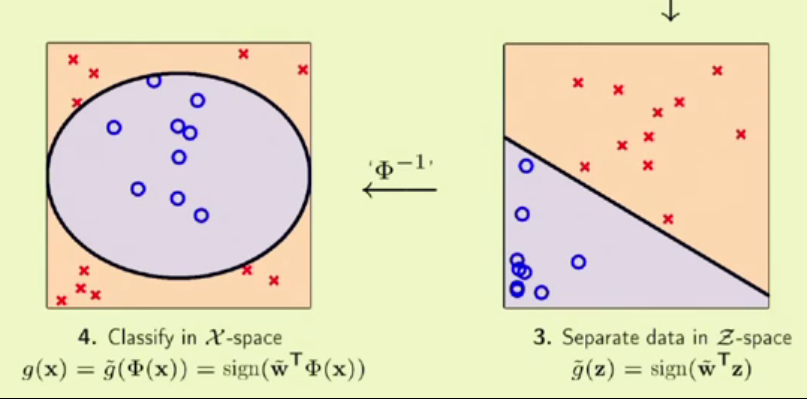
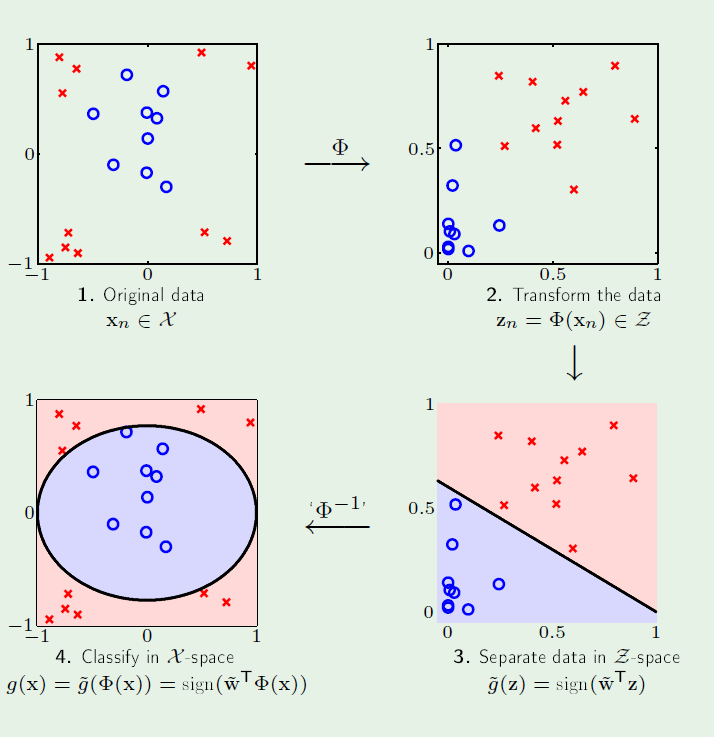
Lecture 4: Error and Noise:

Linear models are like economy cars, they take where you have to go. SVMs are like luxury cars(they come with a cost).

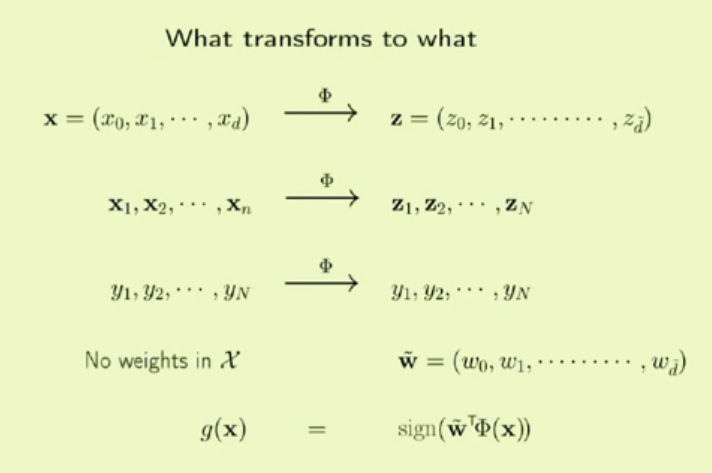


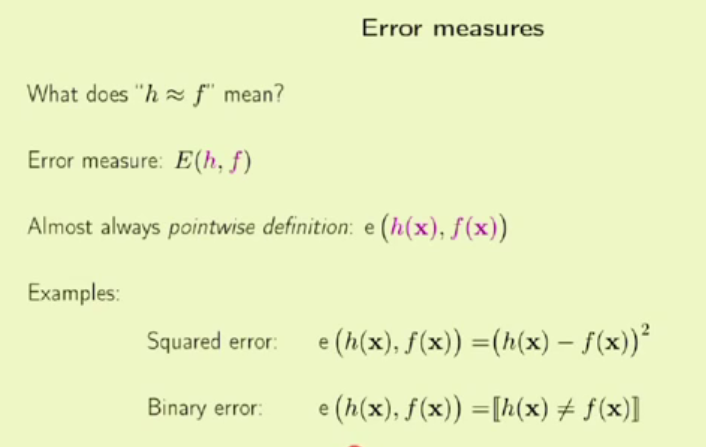
When we cant fit a straight line to separate our data, its non linear in form. To handle such data, we can transform it to linear. This can be done in several ways. Take different quadratic forms of each observation as shown in figure. That will enable us to fit a straight line.

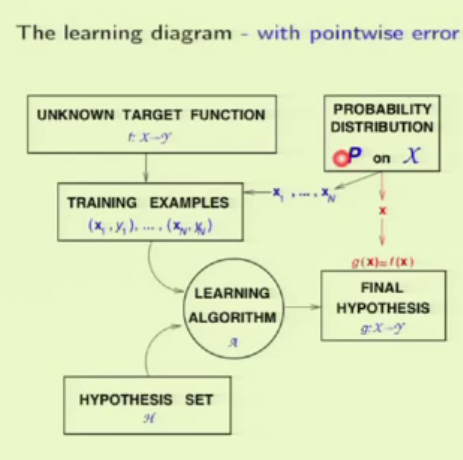
When we do this, we are not only transforming our data, we are also transforming to a different space. Once we are able to separate our data in the new space , we should get back to the original space by using the inverse of transformation we used.

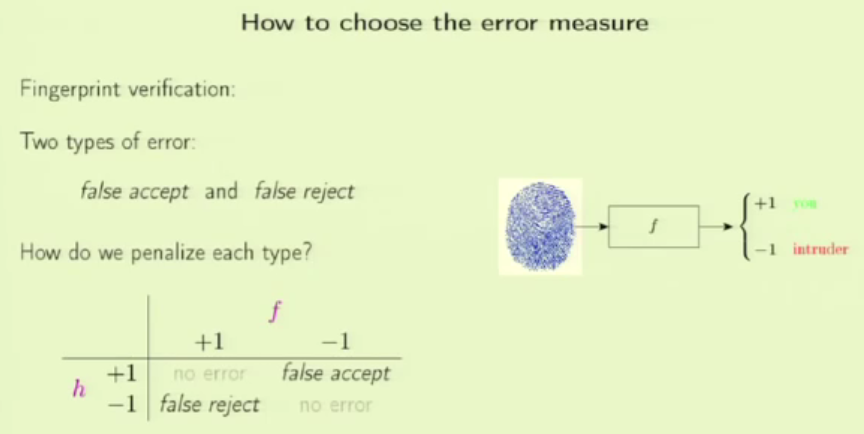
We take the original data, transform it to new quadratic form in a new space. We separate the transformed data in the new space, and use that boundary and H for making predictions.

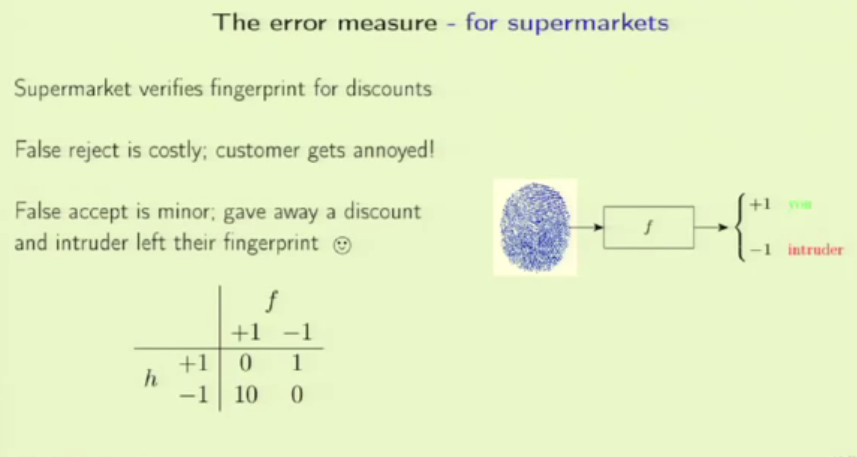
Here we considered a 2D example. Even with many multiple dimensions, the same can be applied, though the gives us some complex boundary.

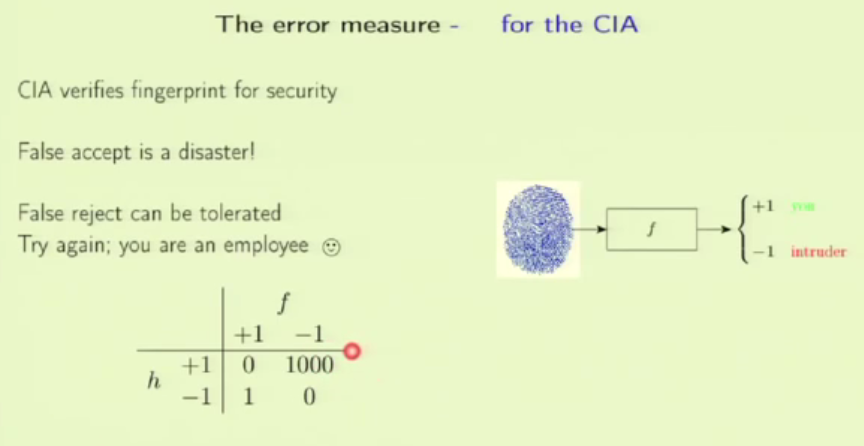
The input data x gets transformed to z. Each of z is some non linear function of each of its corresponding x. z1 need not be the same linear function as z2. Using the car analogy again, the x can be a simple economy car, whereas what we have after transformation is a big truck! You should be careful though, because you may not be able to drive or having a wrong transformation might end up in a crash. Amazing how he teaches and breaks down the concepts to make total sense. The outputs are not transformed, they stay the same. We don’t have weights in x space before transformation. We have weights in z space. The Hypothesis function is a function of z(where z in nothing but a transformation of x)

Error measures answers the following questions: how good or bad is one function when compared to another. EM gives the quantitative measure of how h measures f. can also be used for pointwise error measure. Squared error is an example of EM. We find difference between actual and predicted and square them to give a value for error. In binary classification too, we have EM. Whether the predicted value is 0 or 1. If h and f return same binary value, we have error measure as 0, 1 otherwise.

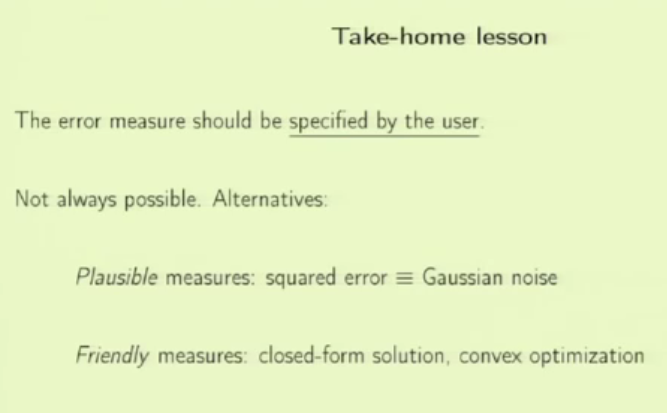
The overall error can be obtained by using average. For in sample error, we take the average and for out of sample error we take the expected value of error measure. We can add EM to revise the learning diagram. WE use probability distribution to generate our data and we use learning algorithm and its hypothesis to obtain a final hypothesis. Now to get error measure, we take some x from the same probability distribution and check if g(x)=f(x) ie if predicted value is same as actual value. It is recommended to use the same Probability distribution function for both data generation and h testing.

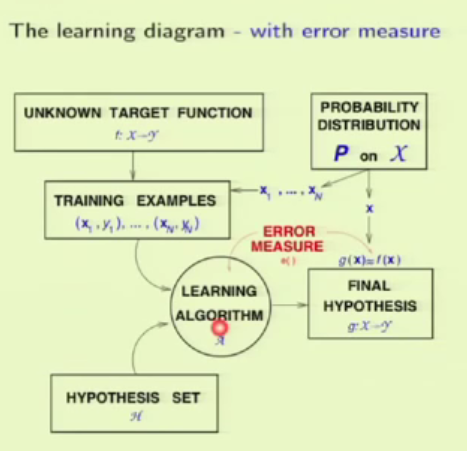
How to choose the EM? What value should we return for showing the measure between g and f. Lets1 take the example of fingerprint authentication. For proper verification, it returns +1, -1 otherwise. We have two types of error. False accept and false reject. We should penalize each type accordingly. False accept is when an intruder gets accepted. When user is rejected its called false reject. When there is a error, we should give some penalty. There are 4 possible scenarios. The question of which error measure to choose is not analytical, it is domain dependent.

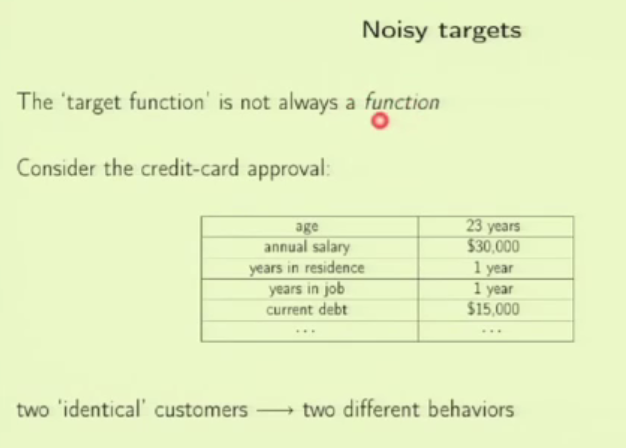
EM for supermarkets: verify fingerprints to claim discounts. False reject in this case is an important customer being rejected, the supermarket might lose the customer. False accept is comparatively minor issue, an intruder getting free discounts. We should penalize more in the first case. In the matrix, the columns are for, user and intruder. Along the row, the labels are for accepted and rejected. So when its you and you get rejected the P is 10. When its an intruder and accepted, the P is 1.

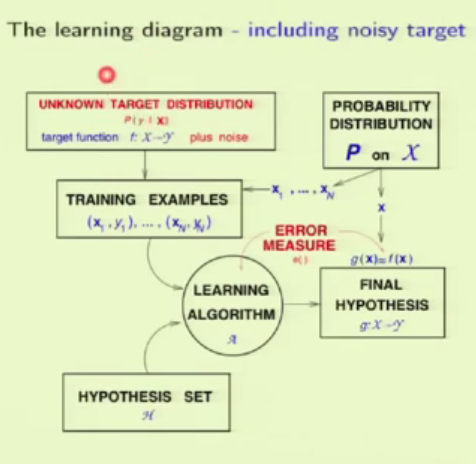
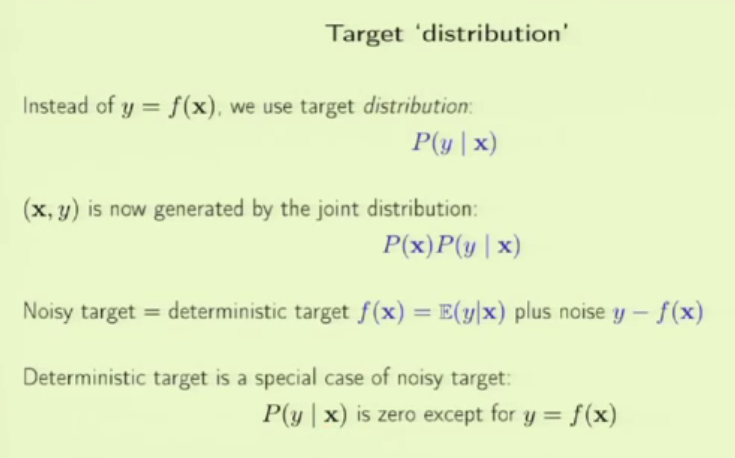
Now the same fp verification example can be taken with a change: it is for giving access to CIA. See how the scenarios change. The false accept is a disaster, unauthorized person given access. It’s a big blow. Whereas the false reject is comparably a lot better, CIA agent will simply asked to verify once more.

See how the penalty changes in this case.

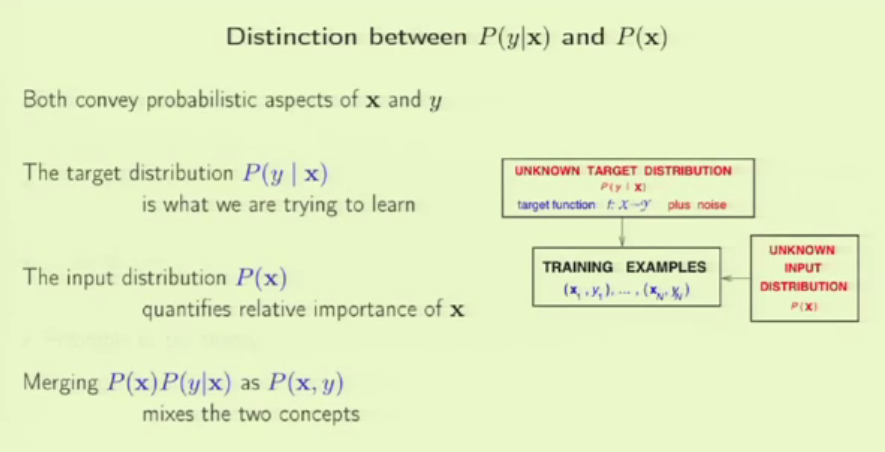
So the EM should be specified by the client. Depending on the domain, one should prioritize what should be avoided. Sometimes its not possible at all to detect and penalize the error.There are also some alternatives for this. Use some analytic arguments. Plausible measures are like squared error or entropy. Friendly measures are not meritorious but are easy to use. We can further update our learning diagram.

The EM gives a quantitive value for how well g approaches f. Based on that value, the learning algorithm evolves and changes its learning.

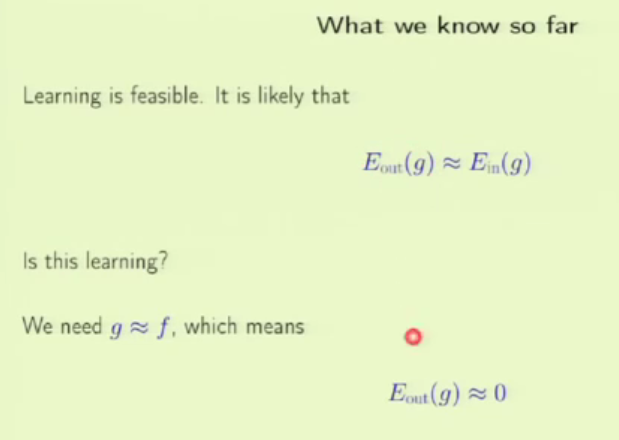
Noisy targets: We seldom get a clean data. The target function is not always a function. Because as in the case of credit example, we might have two very identical customers with same feature values, but different outputs, one accepted and the other rejected. This from the target functions perspective is: the same function mapping to different values for the same input, which is never the case with a function.

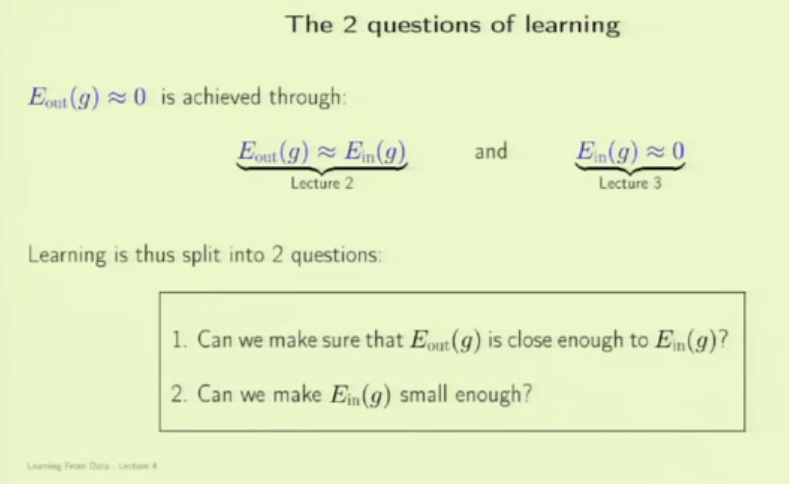


To handle this, we can use the target distribution. Originally both x and y are generated from PD. But here we use the PD to generate x and y is generated from P(y|x) which is the target distribution.



Given some salary, the target distribution predicts yes or no for loan sanction. If the salaries fed to this model are all high, it will learn to predict yes. After this, if borderline salaries are given, it might not do well. Here we are mixing two different concepts to form P(x,y)

The out of sample performance is close to in sample performance, that corresponds to feasibility of learning(this is what we have). Is this really learning? If the final h handed to the client does well and generalizes well, then yes. G == F means out of sample error is close to 0. (What we want is this).

Question 1 is theoretical whereas question 2 is practical. Until now we equated Ein(g) = 0 as the in sample error. But in real, there is no way we can have a out sample error to be 0 which means Ein cant be 0 either. For example, lets take financial forecasting – even if we can get 53% correct consistently, then that’s a good one. Even with 75% correctness, we can have wrong predictions or forecast.

M is number of hypothesis. The bigger is the M, looser is the bounder. If M gets infinite then the bound gets meaningless. As the model complexity increases(more parameters) Ein decreases and also Eout-Ein increases. This is where **Regularization** comes in which is a very important part of machine learning.

How does P(x) impact algorithm? Training points from one distribution and testing from another..which part of the data space should it learn better?

Will P(x|y) play any role? The integral part of learning is to know y given x. P(x) plays some technical role but not that important.

Eout – Ein is the generalization error(this increases as complexity of model increases)

How to choose the penalty? Depends on the problem and requirement.

False accept and false reject shouldn’t be scaled on same magnitude, it should be relative.

What if y is not balanced? It happens because of P(x) and P(y|x)

The transformation we did is to make the process of fitting a boundary to our input features easier. Whereas PCA is for feature selection and extraction. As in the case of digit recognition, we reduced features to 2 from 256.

The target is f(x) but not wTx which is final hypothesis to approximate target function. TF and TD is done even without any learning taking place. IF we have a target function which is noisy, it can be handled by- taking the expected value assuming it’s a numeric function, expected value under probability distribution gives us the average y value for a particular x. This is a function and we are calling it f(x). the remaining part which is y – f(x) will be pure noise.