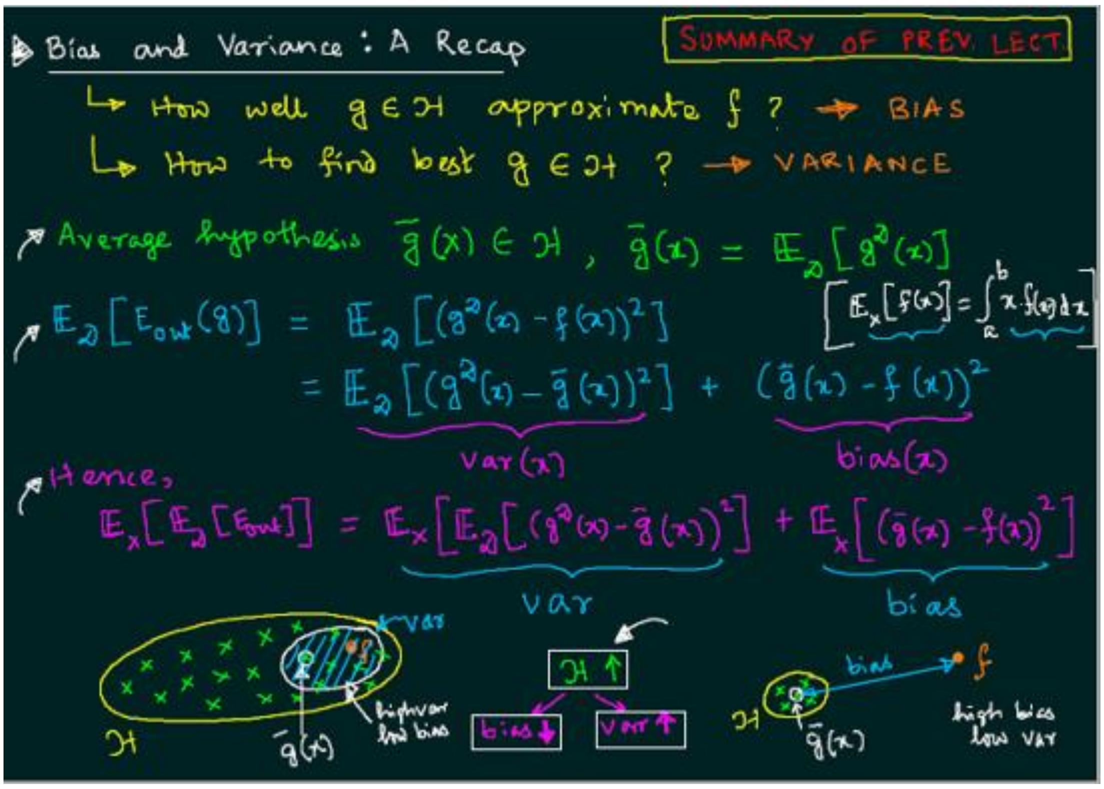
Lecture Scribe for Machine Learning
(CS60050)
20CS91A01

For Lecture Date: 26 Feb 2021 Prepared by: Nwe Ni Kyaw

1.Recap

The last class where we started to understand about the bias and variance in the machine machine learning.

1.1 How well g Є H approximate f?



Bias and Variance : A Recap

SUMMARY OF PREV. LECT.

↳ How well $g \in \mathcal{H}$ approximate f ? → BIAS

↳ How to find best $g \in \mathcal{H}$? → VARIANCE

Average hypothesis $\bar{g}(x) \in \mathcal{H}$, $\bar{g}(x) = \mathbb{E}_{\mathcal{D}}[g^{\mathcal{D}}(x)]$

$\left[\mathbb{E}_x[f(x)] = \int_a^b x\, f(x)\, dx\right]$

$$\mathbb{E}_{\mathcal{D}}[E_{out}(g)] = \mathbb{E}_{\mathcal{D}}[(g^{\mathcal{D}}(x) - f(x))^2]$$
$$= \underbrace{\mathbb{E}_{\mathcal{D}}[(g^{\mathcal{D}}(x) - \bar{g}(x))^2]}_{var(x)} + \underbrace{(\bar{g}(x) - f(x))^2}_{bias(x)}$$

Hence,

$$\mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[E_{out}]] = \underbrace{\mathbb{E}_x[\mathbb{E}_{\mathcal{D}}[(g^{\mathcal{D}}(x) - \bar{g}(x))^2]]}_{var} + \underbrace{\mathbb{E}_x[(\bar{g}(x) - f(x))^2]}_{bias}$$

So this is a question of how good we approximate versus how good we generalize. So how could we approximate means that you have a hypothesis that and we are finding the particular final hypothesis from that hypothesis with respect to the data points or the training examples that are handed over team in machine learning, that is primarily our job. Now, the hypothesis that we find out from the hypothesis said how well or how close it is to the unknown, that is a function.

Is the first question that is the approximation question that we try to us and that we also call in in colloquial machine learning as the basis of that hypothesis, that the hypothesis is how much biased towards finding a solution which can approximate. On the other hand, we also have seen that if the hypothesis set out the model being too complex, then it is difficult to find the actual or the best hypothesis, even if it is within the realms of diversity. Because the question is that you will only be going to find that particular best hypothesis through your data point, nothing else. So there the challenge lies and we see that many times, even if our hypothesis state is very expressive, but we somehow underplay and find out the hypothesis, which is not the best. So therefore, the best hypothesis.

1.2 How to find best g Є H

So this is the approximation, which is generally the same question. And this generally is in fact, that we also capture in terms of what we call variance and we try to find out a formal expression for that.

we define the 'average' hypothesis $\bar{g}(\mathbf{x})$:

$$\bar{g}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})\right]$$

When we try to find out the average hypothesis from the hypothesis that it is nothing but a particular hypothesis is basically comes out from the given training example or the data point that that has applied. So whatever hypotheses we could find out from any of this study, if I take an expected value of that, that is the expected best hypothesis or we call it as an average hypothesis.

And if you remember,

$$E_x[\ f(x)\ ] = \int_a^b f(x)dx$$

it was really we call something expectations of X, but function effects is if the X range is within the domain of from the from A to B, then usually we calculate effects using this Formula X into a fixed degraded from A to B, from A to B, the range is continuous. And if the range is this quickly, this submission and put an average of that, that's all in from the provability side, you need you need to know what you need to refresh it.

$$E_D\ [\ E_{out}\ (g)] = E_D\ [\ g^D\ (x) - f(x)^2]$$

$$= \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}(\mathbf{x})} + \underbrace{\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2}_{\text{bias}(\mathbf{x})}$$

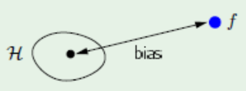
Now, what we want we want to eat out, that is the out of simple estimation of error with respect to the hypothesis that people no remember could be anything depending on the data point that is given to you. It is not a particular data point, but depending on all the data points, you could be different. So we need an expected out of sample performance or that we have seen that what is a particular sample performance minus its difference, we assume is some squared error for convenience. And then what we did is that we took an expected value and we have these computations where we put a minus Deibert X plus Durex in between and had a square custom cancels.

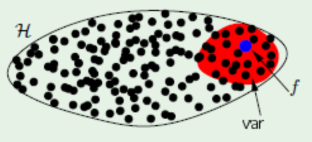
And finally we reach in a scenario where this is the term which says that how good you are generalizing with respect to the hypothesis that you find. So how far it is from the best hypothesis in that particular set that could be found? And this is saying that how good is your hypothesis set in making the best hypothesis from it within the approximate properly the function, the target function that is unknown to us.

Therefore,
$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x}) + \text{var}(\mathbf{x})] \\ &= \quad \text{bias} \quad + \quad \text{var}\end{aligned}$$

So which means that the first term is actually the variance if we take it over every possible X and the expected value, the first time is the variance and the second is the bias, because it says that the whole hypothesis is being biased so that the best one is also far away from iPIX, how far it is. And this says that how can you generalize by choosing a hypothesis with respect to your best hypothesis? So assuming that best hypothesis generalizes best, so how better to generalize?

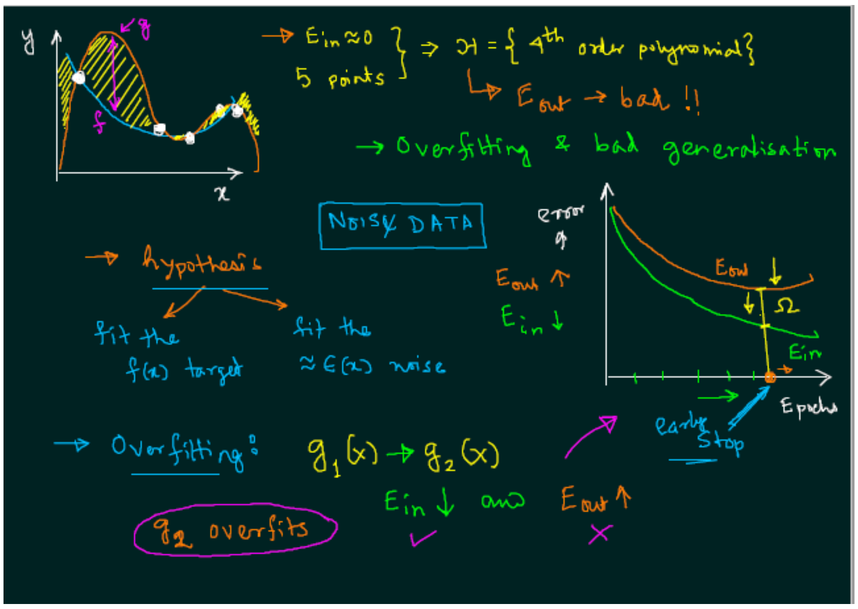
$$\text{bias} = \mathbb{E}_{\mathbf{x}}\left[\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] \qquad \text{var} = \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]\right]$$





$\downarrow \qquad \mathcal{H}\uparrow \qquad \uparrow$

And also, you have seen pictorially what happens if your hypothesis has become too complex. That is, you allow many possible hypothesis. Then what will happen is that finding out gibberish is difficult. So you have a high variance. However, your body may be this hypothesis that contains the target function itself and is sufficiently close to to say it's a low bias but high variance. Whereas in this case where your hypothesis model is very restricted there, you may easily find out the best one by navigating through the hypothesis space, then for your variance is very low. However, it may be the case that this hypothesis is far away from from the actual target function. So your bias might have some practical experience.



So let's say that I have a very simple target function and since I do not know the target function, just I will give you some data points for it. So let's say I have. This is I give you the data point and so let's suppose my target function is this one, which I do not know.

Anything below, which I do not know and I give you certain data points. Your five data points which are a little bit noisy. So a little noise, I mean, not drastically, it will impact the performance of your final hypothesis, but I am seeing a little noise they have. So therefore the target function does not have this particular data point over it because it had some noise target function plus some nice. And now I need to put it this five datapoints with.

Some kind of a cut, as you can see, that my overall goal would be to minimize that in almost toward zero, so I have five points to fit. And therefore, my immediate conclusion is from this, that let me take in my hypothesis, said Francisco. Fourth order polynomial. Because for total polynomial is most likely to fit all these five points very perfectly.

Let's say you put it this way, so fourth order polynomial may look like this and you put your points in a bowl made plain so that it becomes over the line. If at this point in this. Now, as you can easily understand here, your team is basically these things contribute to each of you, that how bad you are doing with respect to the target function is contributed by. So therefore, you can easily find out that while taking this one, I'm fine, but it is indeed very bad.

Now, if you if you try to look, if you take a look very carefully at what has happened and that that that has given the thing a little bit bad shape, because I have tried to fit as much as possible with equal to zero. The point is, see, there is some noise in the point. And when you try to fit your final hypothesis there, according to that noise, you are also trying to fit in the noise instead of only the point. That's point number one.

2. Overfitting and Bad Generalization

This is basically it results basically, I can see that this is a kind of overfitting. It resolves to bad generalization because my ego is very big, but I could see a general generalization. It's a bad generalization.

So if you look at the Omega, that is the difference between in an Eout boundary that generalization is going back. So at that point, I understand that there is no more that I could do with respect to this kind of activity. So which means that even if I try to fit very large with the large number of data points and gradually at each epoch, I keep on training, training, training and pushing down the end, using gradient descent or whatever else, I will certainly see there is a point where Eout is not behaving as intended and it is going up. So that is the point where you that means this many examples after that you are trying to overpacked that. This is an indication so far we can say that when Eout will go up, irrespective of Ein is coming down, then at this scenario I see very little effect. And as you can see from this figure, if we have such kind of a graph and a validation data with this from this picture, what we derive is that why should we go beyond this to we will try to fit it within this position and we will stop. This is called early stopping. Stopping. This is a thing that we often do to eliminate the type of overfitting that we will deal with these details when we do for the validation and regularization.

2.1 Hypothesis

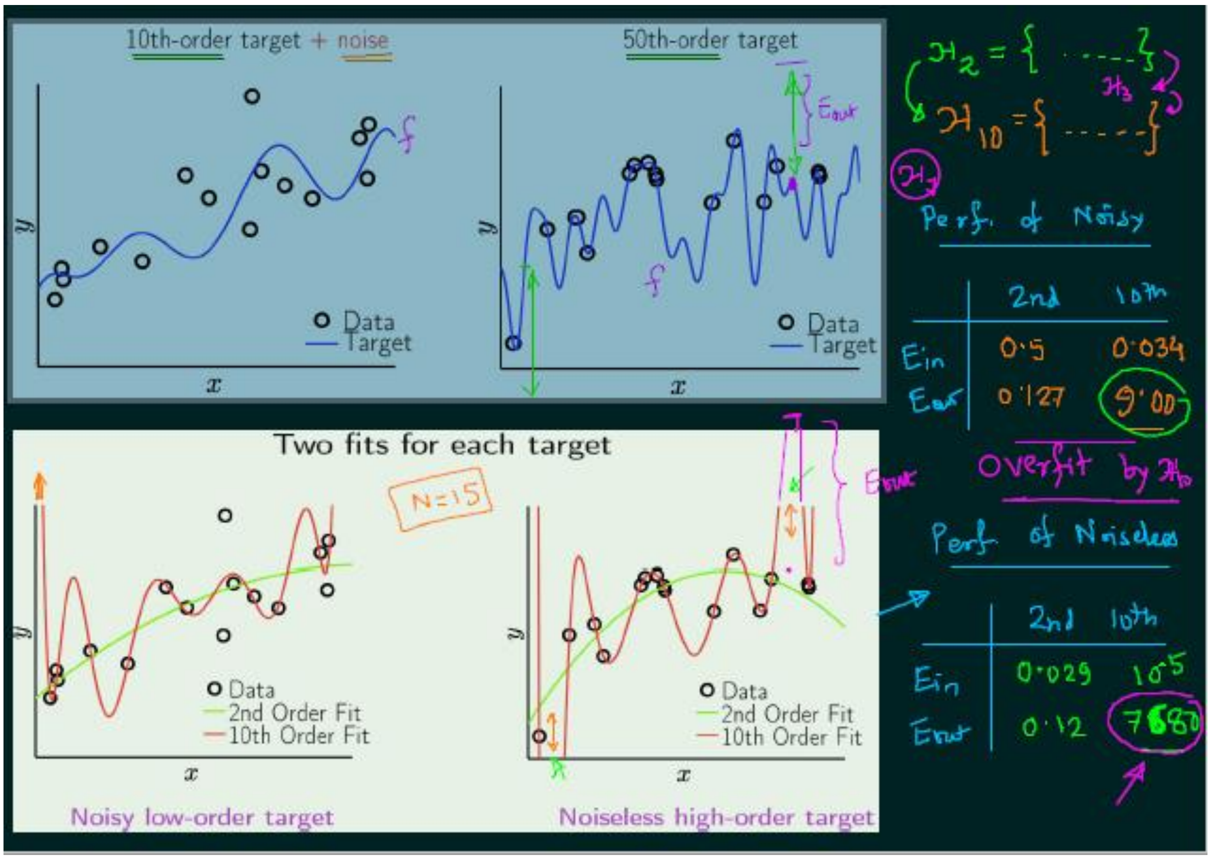
The hypothesis that you're trying to project is to is trying to do two things. One, it is trying to fit the effects target, which is unknown. And it also tries to fit the deviation in terms of every point, the deviation sigma X noise, and that gets things becomes very bad with respect to the higher order polynomial. So this is a scenario where you have noisy data.

2.2 Overfitting

Overfitting is a respected term that overfitting with respect to work, so therefore I can say that I have two hypothesis $g_1(x)$ and $g_2(x)$. g_1 overfit, then g_2 overfit when I will find that whenever I go from g_1 to g_2 , Ein is reducing and Eout is increasing.

g_2 because when I move from g_1 and g_2 and make it complicated, maybe a hypothesis, I see that I have a drastically improved performance in Ein, but similarly, I have a very bad performance with respect to your as we see in this carvelle. So then we define that overfitting means this g_2 to the hypothesis that we derive is overfitting.

With respect to g_1, maybe you want to work with respect to some other, but since it's overfitting, so it should be some kind of comparative competitive behaviour between two hypotheses.



So here are two examples. The top figure, this one is basically the correct target function is in blue. The correct target function, let's correct that at the 10th-order target, whatever we may correct, target functioning the 10th-order target, a target that I don't know I don't know about the target function, it's a 10th-order target with some noise like this.

The other example that we will take is the 50th-order target function. Trying to fit in a second order polynomial hypothesis and another learner, which is in the red here, is trying to fit into the 10th-order polynomial. For example, let's say the learner in the red is thinking that maybe the target is the 10th-order. Let me put it to the 10th-order polynomial.

So the hypothesis of one learner consists of only second order polynomial, the hypothesis of other learners consist of all 10th-order polynomials. Both hypotheses are infinite because the coefficient is can be continuous and you can take any second order polynomial, many such equations. So therefore, if you try to fit like this now with respect to the blue curve you see here and with respect to the blue curve, you see the red curve as well as the green curve, we will find that the of 10th-order polynomial is a chance. There is a chance that it may overfit because it is trying too hard to fit the points. And indeed, noise is there and hence noise is also coming into picture.

Polynomial target can also be pulled short because it's a f58th-order target that we need to reach. So any 10th-order polynomial can also in second order polynomial is let's say somebody tries with the lock that let us try to take another step. But however, here you see the the way it fits the 10th-order polynomial, looking at the 15 points. So all of these cases, I have even 15 points in both cases. So it tries to fit in too tightly.

3. The role of Noise

3.1 Performance of Noisy

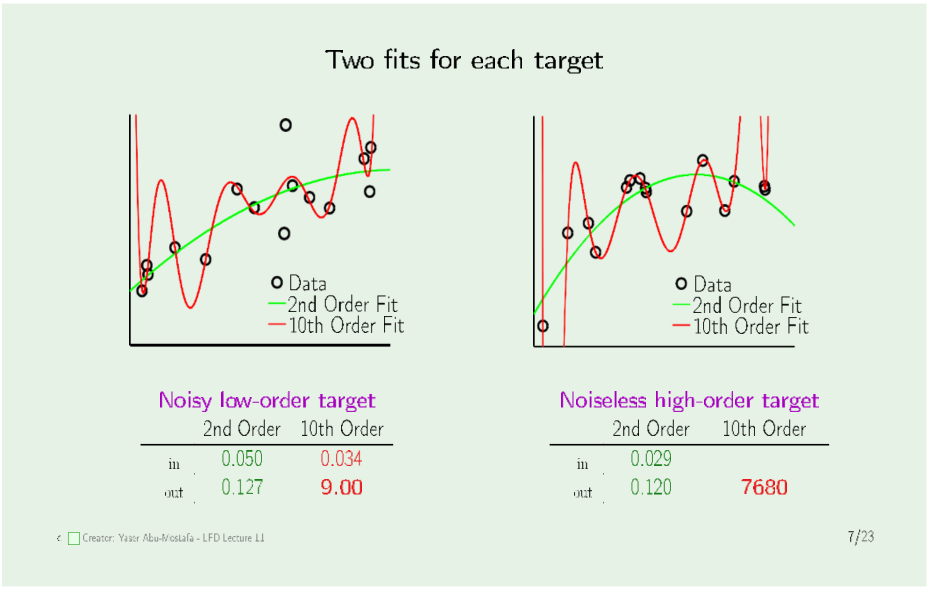
Now I have the figure with me, you can just experiment with any of the linear regression techniques or Second-order and get these figures with you. Obviously consider polynomial. That's better, because if the target is 1oth-order, this will obviously better fit for the given data points that I have. But here you see, it's horrible. In the main reason being, if you look at the blue curve with respect to the blue curve and there are positions like this position, it shoots up, it shoots up so much that it indeed carries this kind of bad out of generalisation.

3.2 Performance of Noiseless

And on the other hand, if we try to see the performance of Noiseless said that is this one.

The same thing we will try to see here that let's say this is second order and 10th-order. You will see the first one in the second order drastically, it also tries to reduce it very well and depending on the data point distribution, it does very well. 1oth-order is marvelous. It does 10^{-5}. It's marvelous. However, if you go for Eout. It is this and do you know what the experiment figure out of the 10-th order 7680.

The best 2nd and 10th order fits are shown in Figure, and the in-sample and out-of-sample errors are given in the following table-



Noisy low-order target

	2nd Order	10th Order
in	0.050	0.034
out	0.127	9.00

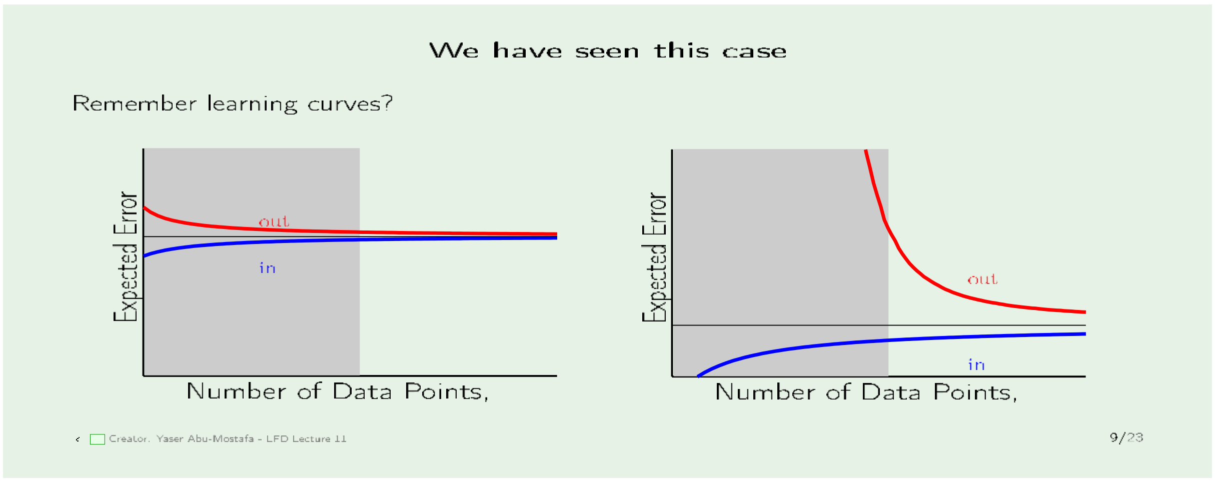
Noiseless high-order target

	2nd Order	10th Order
in	0.029	
out	0.120	7680

What the learning algorithm sees is the data, not the target function. In both cases, the 10th order polynomial heavily overfits the data, and results in a nonsensical final hypothesis which does not resemble the target function. The 2nd order fits do not capture the full nature of the target function either, but they do at least capture its general trend, resulting in significantly lower out-ofsample error. The 10th order fits have lower in-sample error and higher out-ofsample error, so this is indeed a case of overfitting that results in pathologically bad generalization.

3.3 Learning curve

Figure : Overfitting is occurring for N in the shaded gray region because by choosing 1-l1 0 which has better Ein, you get worse Eout ·



The models H2 and H10 were in fact the ones used to generate the learning curves in earlier lecture , and we use those same learning curves to illustrate overfitting in above Figure. If you mentally superimpose the two plots, you can see that there is a range of N for which H10 has lower Ein but higher Eout than H2 does, a case in point of overfitting.

4. Impact of “Noise”

How the noise level , the target complexity Q_f, and the number of data points N relate to overfitting. We compare the final hypothesis g_{10} belongs to H_{10} (larger model) to the final hypothesis g_2 belongs to H_2 (smaller model) . Clearly, Ein (g_{10}) ≤ Ein (g_2) since g_{10} has more degrees of freedom to fit the data. What is surprising is how often g_{10} overfits the data, resulting in Eout (g_{10}) > Eout (g_2). Let us define the overfit measure as Eout (g_{10})≤ Eout (g_2). The more positive this measure is, the more severe overfitting would be.

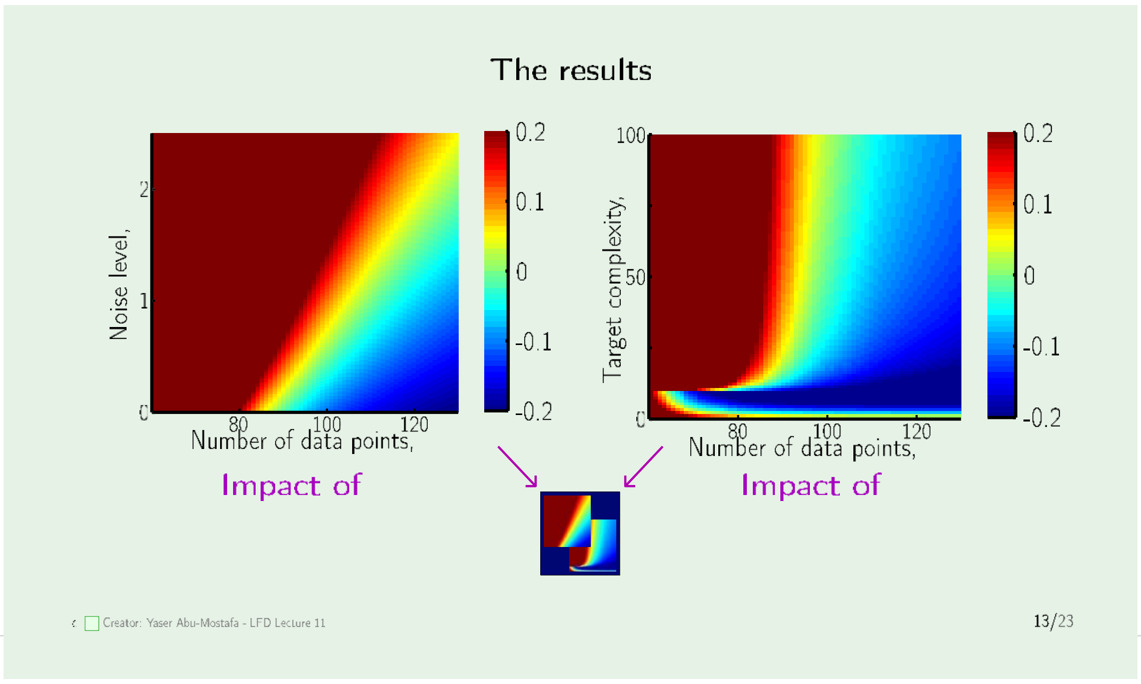
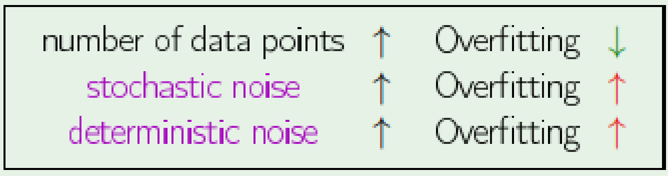


Figure shows how the extent of overfitting depends on certain parameters of the learning problem . In the figure, the colors map to the level of overfitting, with redder regions showing worse overfitting. These red regions are large overfitting is real, and here to stay.

Figure part (a) reveals that there is less overfitting when the noise level drops or when the number of data points N increases . Since the 'signal' f is normalized to E[f^2] = 1, the noise level is automatically calibrated to the signal level. Noise leads the learning astray, and the larger, more complex model is more susceptible to noise than the simpler one because it has more ways to go astray.

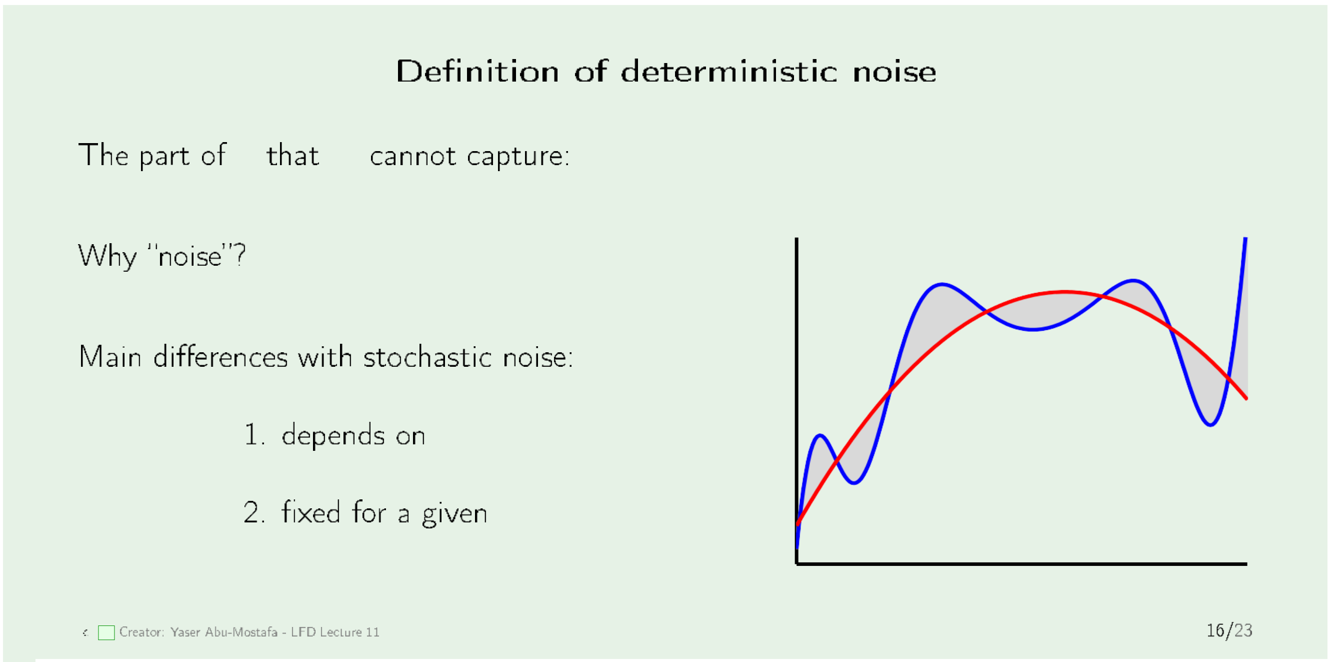
Figure part (b) reveals that target function complexity Q f affects overfitting in a similar way to noise, albeit nonlinearly. To summarize,



number of data points	↑	Overfitting	↓
stochastic noise	↑	Overfitting	↑
deterministic noise	↑	Overfitting	↑

5. Deterministic Noise

Why does a higher target complexity lead to more overfitting when comparing the same two models? The intuition is that for a given learning model, there is a best approximation to the target function. The part of the target function 'outside' this best fit acts like noise in the data. We can call this deterministic noise to differentiate it from the random stochastic noise. Just as stochastic noise cannot be modeled, the deterministic noise is that part of the target function which cannot be modeled. The learning algorithm should not attempt to fit the noise; however, it cannot distinguish noise from signal. On a finite data set, the algorithm inadvertently uses some of the degrees of freedom to fit the noise, which can result in overfitting and a spurious final hypothesis.



Definition of deterministic noise

The part of that cannot capture:

Why "noise"?

Main differences with stochastic noise:

1. depends on

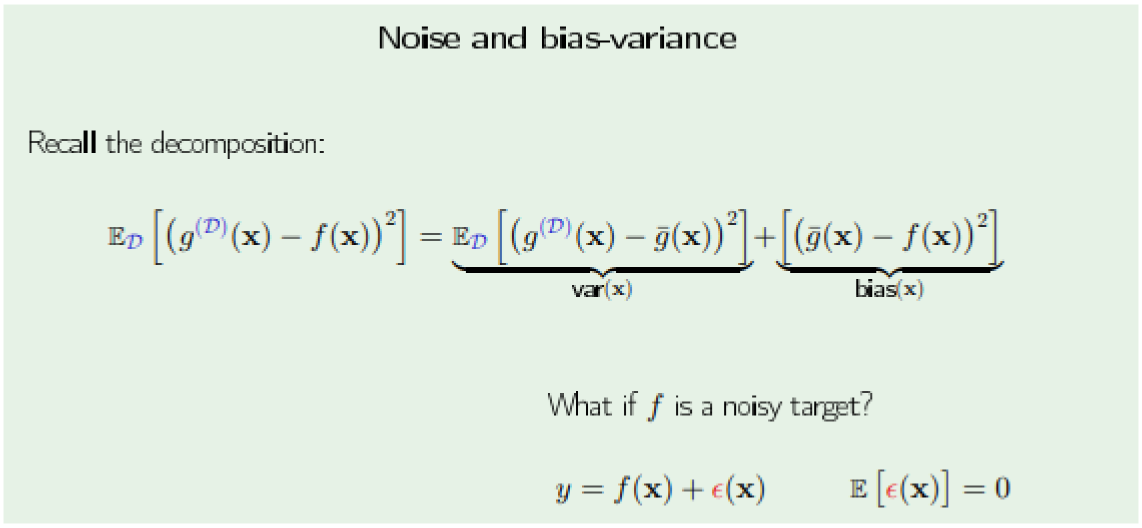
2. fixed for a given

Creator: Yaser Abu-Mostafa - LFD Lecture 11

16/23

Figure 1: Deterministic noise. h is the best fit to f in H2. The shading illustrates deterministic noise for this learning problem.*

Graph in Figure illustrates deterministic noise for a quadratic model fitting a more complex target function. While stochastic and deterministic noise have similar effects on overfitting, there are two basic differences between the two types of noise. First, if we generated the same data (x values) again, the deterministic noise would not change but the stochastic noise would. Second, different models capture different 'parts' of the target function, hence the same data set will have different deterministic noise depending on which model we use. In reality, we work with one model at a time and have only one data set on hand. Hence, we have one realization of the noise to work with and the algorithm cannot differentiate between the two types of noise.



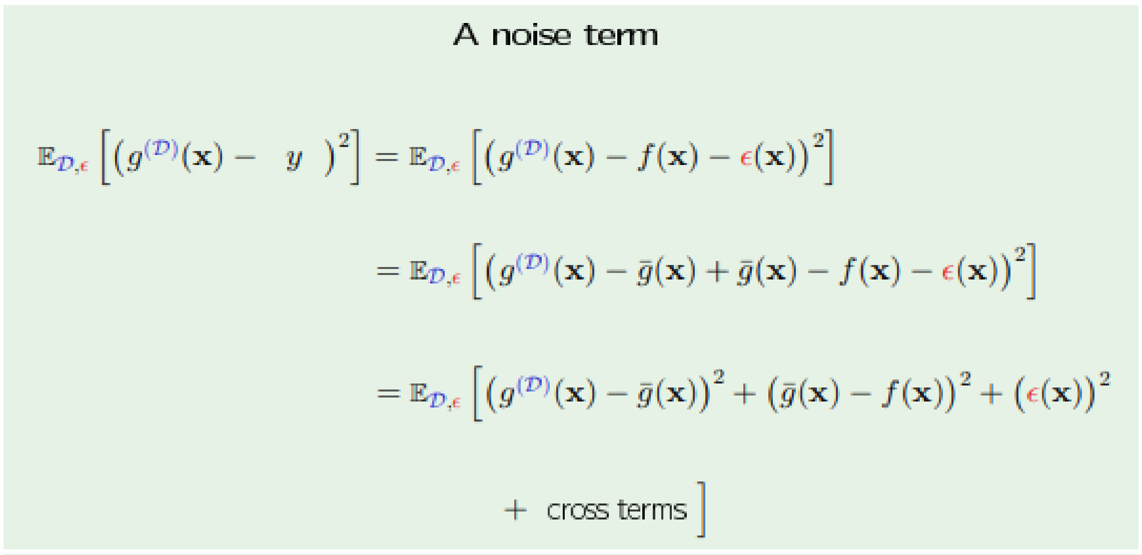
Noise and bias-variance

Recall the decomposition:

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-f(\mathbf{x})\right)^2\right]=\underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}(\mathbf{x})}+\underbrace{\left[\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2\right]}_{\text{bias}(\mathbf{x})}$$

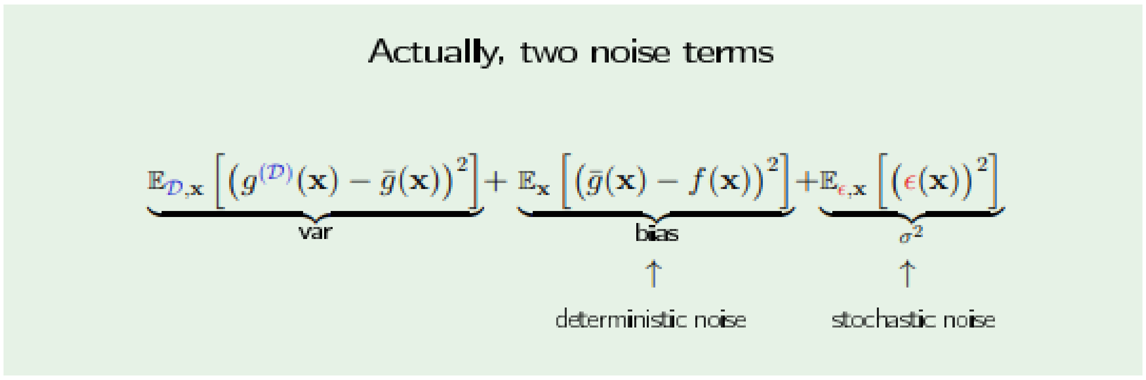
What if f is a noisy target?

$$y=f(\mathbf{x})+\epsilon(\mathbf{x}) \qquad \mathbb{E}\left[\epsilon(\mathbf{x})\right]=0$$



A noise term

$$\begin{aligned}\mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\ y\ \right)^2\right]&=\mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-f(\mathbf{x})-\epsilon(\mathbf{x})\right)^2\right]\\&=\mathbb{E}_{\mathcal{D},\epsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})+\bar{g}(\mathbf{x})-f(\mathbf{x})-\epsilon(\mathbf{x})\right)^2\right]\\&=\mathbb{E}_{\mathcal{D},\epsilon}\Big[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2+\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2+\left(\epsilon(\mathbf{x})\right)^2\\&\qquad+\ \text{cross terms}\Big]\end{aligned}$$



Actually, two noise terms

$$\underbrace{\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x})-\bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}}+\underbrace{\mathbb{E}_{\mathbf{x}}\left[\left(\bar{g}(\mathbf{x})-f(\mathbf{x})\right)^2\right]}_{\text{bias}}+\underbrace{\mathbb{E}_{\epsilon,\mathbf{x}}\left[\left(\epsilon(\mathbf{x})\right)^2\right]}_{\sigma^2}$$

$\uparrow$ deterministic noise $\qquad \uparrow$ stochastic noise

The bias-variance decomposition, which we discussed earlier is a useful tool for understanding how noise affects performance:

$$\mathbb{E}_{\mathcal{D}}[E_{\text{out}}] = \sigma^2 + \text{bias} + \text{var}.$$

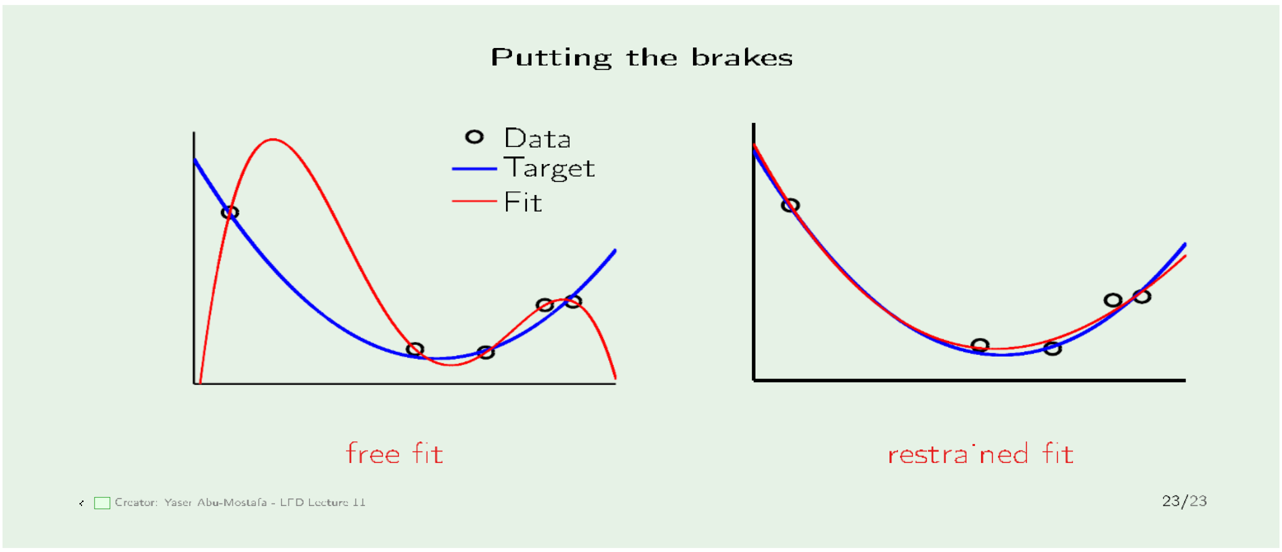
The first two terms reflect the direct impact of the stochastic and deterministic noise. The variance of the stochastic noise is sigma2 and the bias is directly related to the deterministic noise in that it captures the model's inability to approximate f. The var term is indirectly impacted by both types of noise, capturing a model's susceptibility to being led astray by the noise.

6. Dealing with Noise

To deal with noise we have two cures that are

1.Regularisation: Putting the brakes.

2.**Validation**: Checking the bottom line.



Regularization is our first weapon to combat overfitting. It constrains the learning algorithm to improve out-of-sample error, especially when noise is present.

Why regularization is needed. The linear model is too sophisticated for the amount of data we have, since a line can perfectly fit any 2 points. This need would persist even if we changed the target function, as long as we have either stochastic or deterministic noise. The need for regularization depends on the quantity and quality of the data. Given our meager data set, our choices were either to take a simpler model, such as the model with constant functions, or to constrain the linear model. It turns out that using the complex model but constraining the algorithm toward simpler hypotheses gives us more flexibility, and ends up giving the best Eout. In practice, this is the rule not the exception.

Note: Scribe is based on lecture taught by Prof. Aritra Hazra on 26 Feb'21 under course O Machine Learning (CS60050). All the figures are taken from slides and handout uploaded on course website.