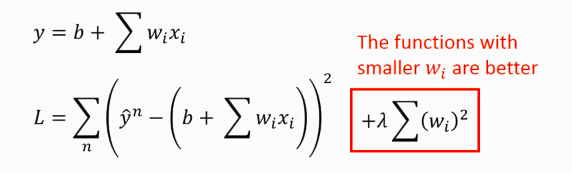
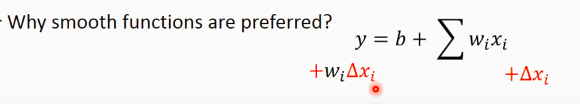
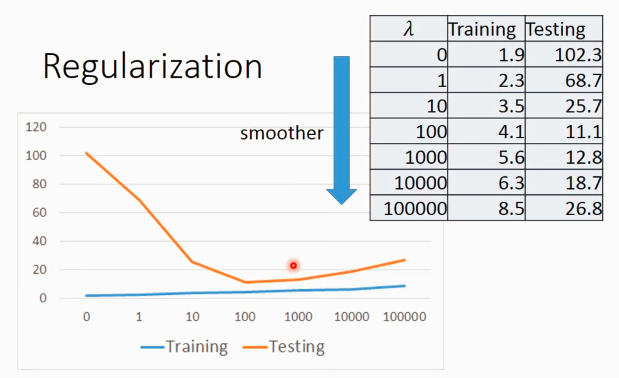
# Regularization in regression



The smaller the w, the output is more insensitive to the change of input:



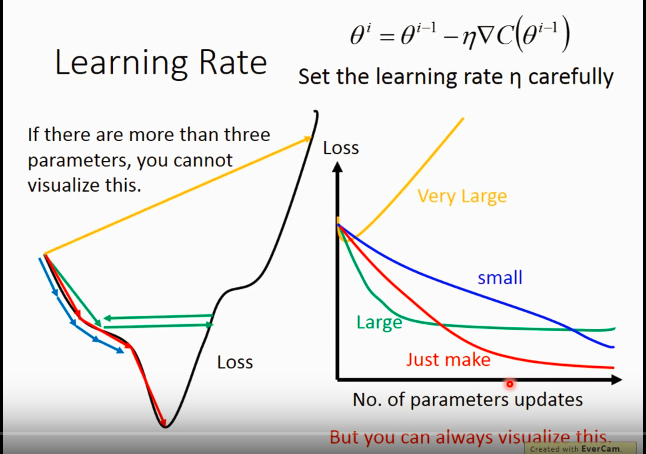
A smoother function have less influence.



However, it doesn't mean the smoother the better. Again, we need to select lamda to obtain the best model.

## Tuning learning rate:

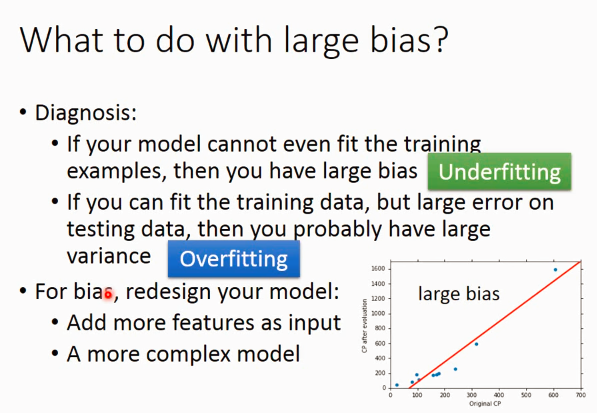
I heard that sometimes, deep learning is like “metaphysics” in parameter tuning, but it seems we can find some mathematically basis.

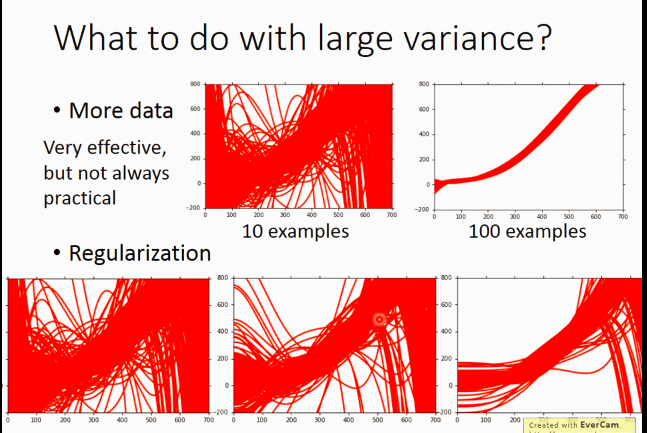


In this example, we can find that when the learning rate is too big, it will miss the minima point; when it’s too small, it would be too slow to find the minima. So, we need to do tune the parameter to make it find the minima quickly.

# Where does the error come from?

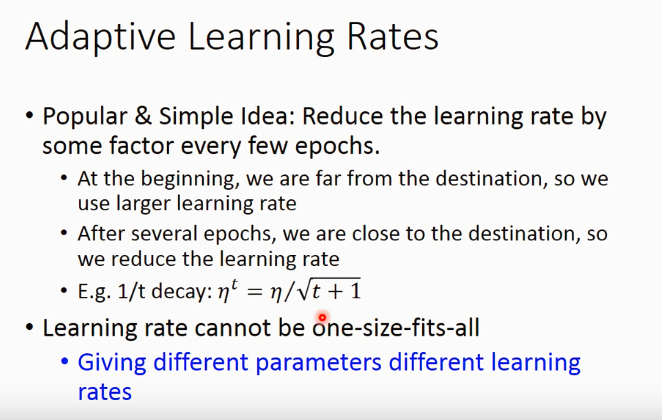
Simper model is less influenced by the sampled data.





# Gradient Descend

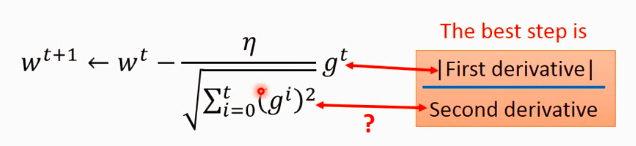
At first, the initial value of weight and bias might be far away from the real value, so the learning rate should be big, after some iterations, the rate should be smaller to avoid skip/miss the minima point.

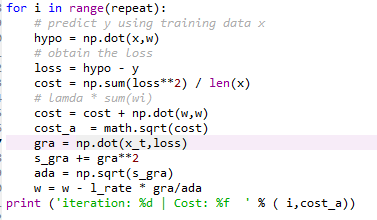


## Two method to do the optimization

1. **Adagrad gradient descend:**

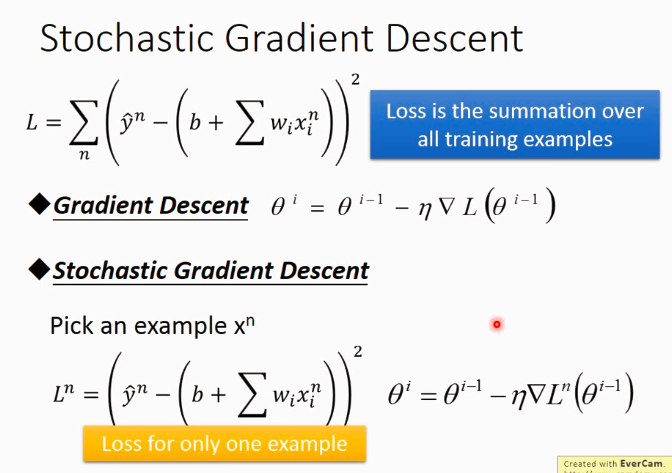
When we have multiple parameters, the bigger gradient doesn’t necessarily mean bigger learning rate. So we can not just rely on gt.





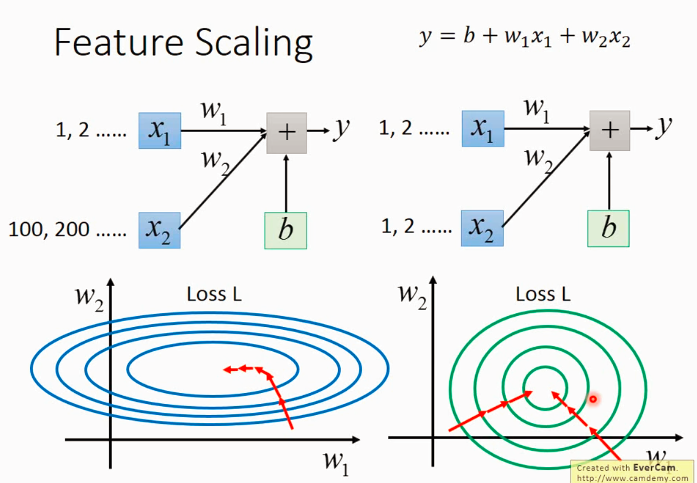
1. **stochastic gradient descent**

adagrad use all the samples to find the loss while sgd update the parameters once when it is feed with a new sample.



## Feature Scaling

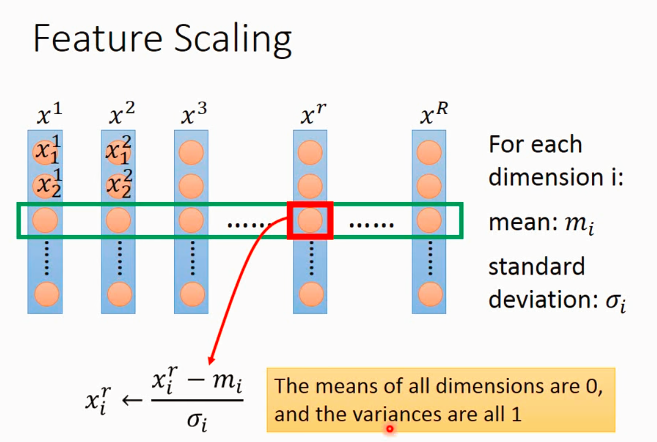
It makes parameter update more efficient.

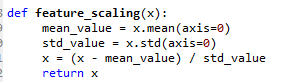


The update is always goes along the gradient direction, not the minima direction(shown as red arrows above). So for the ellipse, the gradient direction is not always the same as the minima direction, so it’s not the “shortest path”. But the Loss on the right, the gradient direction is always the gradient direction, so it’s very efficient.

Ps: learning rate for w1 and w2 on the left side should be very different to get a good model.(I’m not sure if this statement is totally correct, but you might get what I mean, hopefully?)

**One simple way to do the scaling**





Some problems I met in HW1:

1. tensorflow:
2. optimization function

I tried to use tensorflow to do the prediction, but when I used “GradientDescentOptimizer”, weight and biases are all 0, but when I changed to “AdagradOptimizer”, then it’s ok. Maybe I need to study more to figure it out.

1. I deal with a batch of data, not single ones, it might be quicker.

There’s a loooooooooot to improve even for this simple example…

Problems remained:

1. Coding the non-numeric feature (rainy, cloudy, snow)
2. …