So as we have seen Bias Variance trade off for LR.

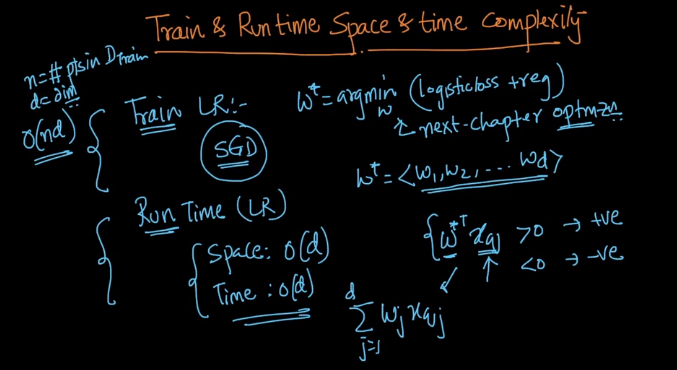
Now lets have a look at Train and Runtime complexity.

We have not seen how to train a model in LR but we will see it later.

And what actually training means in LR , nothing but finding W\* i.e. optimization.

So as we get out W\* so at run we just need to keep our new W in memory and what is size for W\*, it is d as it has d dimensions.

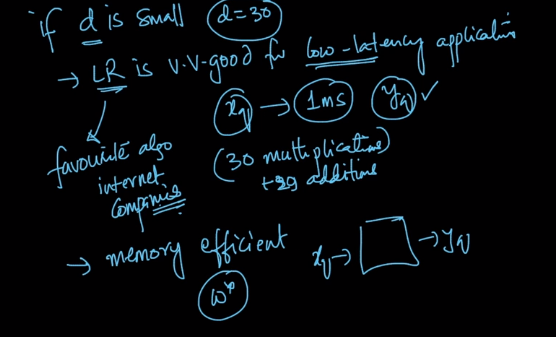
So Space complexity is O(d) and since it just need to do d multiplication with Xq so time complexity is also O(d).



So suppose if our d is small say 30 so in this case LR is very good for low latency applications like Internet companies like Amazon.

Because it just need to do 30 multiplication with Xq and 29 addition.

And it also memory efficient.



But if our d is large say 1000 or more then the multiplication we need to do is 1000 and so there is one thing called L1 reg. in LR.

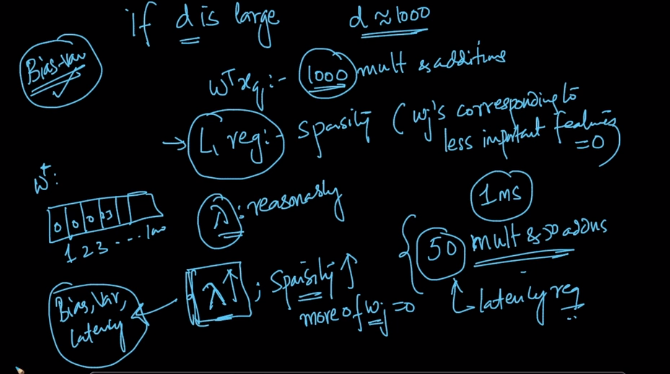
So when we perform L1 reg. so it converts less important features to 0.

And more the value of (Lambda ) more the sparsity increases and if our requirement is that we cant perform more than 50 multiplication so we need to make and equilibrium between Bias , Variance and lambda.

Of course increasing lambda after an extent will increase Bias and which results in underfitting but sometimes when there is low Latency requirement so its to have bad working model than no model.

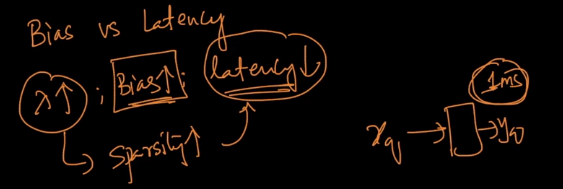
Because end of the day if requirement is very low latency somewhat 1ms and we are not able to achieve that than the model wont be accepted and hence its better to have a working model than no model.

And so in this case the lambda will be reasonably high.



SO for this we need to maintain some equilibrium state between all three i.e.

Bias, Variance and Latency.



Comments:

