Lecture 8 | Cross-Validation | Stanford CS299

Date de création	@January 21, 2025 6:32 PM
Étiquettes	

Questions

- What are the key ideas?
- What terms or ideas are new to me?
- How would I define them?
- How do the ideas relate to what I already know?
- Should not assume that the profesor are always correct. Asking appropriate questions.
- What are the good ideas?
- Do the ideias have other applications?

▼ How reduce bias or variance in your ML algorithm?

- Generative models looks at each classe, one step at time.
- Descriminative models use Gradient Descent to find the best fit line.

> **Generative** models learn backwards (features to class).

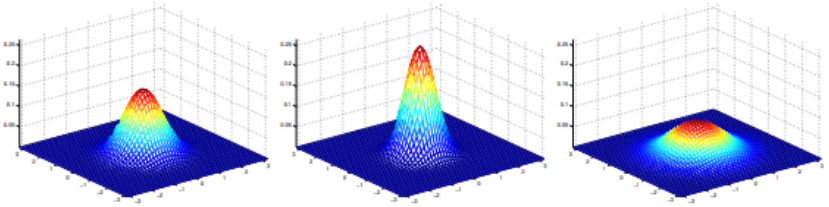
▼

This formula tell us how data behave, and we can use this to make sense of this behave and make cool stuff, like:

- predicti the future.
- analyse data.
- understand patterns.

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

Here are some examples of what the density of a Gaussian distribution looks like:



▼

Even they end up using the same Sigmoid Function GDA makes Strong Assumptions (About the Distribution Of Data) when Logistic Regression Don't.

Using GDA means you know more things (in theory) that makes a good model, since you already knows those things.

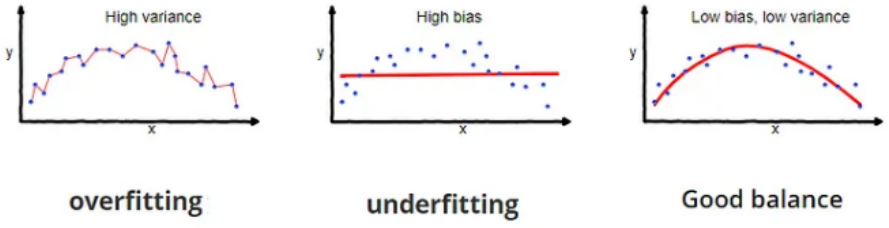
Lecture

Outline

- *Bias/Variance*
- *Regularization*
- *Train/dev/Test Split*
- *Model Selection*

Key Points

Bias and Variance is a simple concept to understand but hard to master. One should learn this one to impress their pairs. Bias and Variance at one image:



Bias in machine learning **has a very diffetente meaning**. Bias means that our model has pre conceptions about how to fit data. Variance, could be understanding as **different lines for different set of values.**

First, try the **quick and dirty approach**, to create ML algorithms. They to understand, why it does not work well and improve it.

Regularization

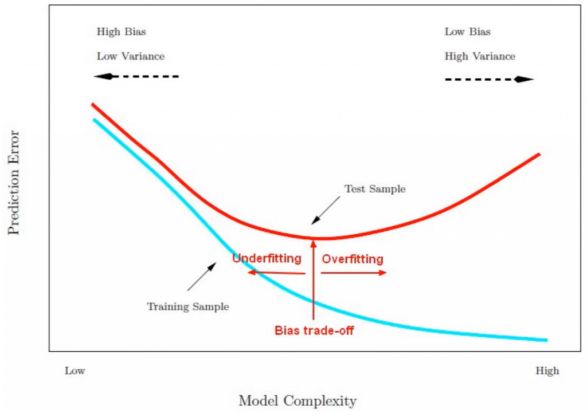
Regularization is a mathematical method that help us prevent overfitting (model learn noise).

To regularize, you just need add some therm, like: $\lambda||\Theta||^2$. We call this guy, **regularization term.**

You can say, that, SVM has his own regularization term.

Apply scaling can help when apply regularization, since the values will be in a certain range, let's say [-1, 1]

Supose we believe that our data has a Guassian distribution with parameters (a and b). The MLE is a way to find those parameters.



> Yes, more complex about model, less the error but out model lost it's hability to generalize.

Professor Andrew says that, sometimes is just better use more data in our Train set in comparison with out validation and test set.

Aha: it's ok mesure the performance of you ml model using the testset? but not okay to make decisions about it?

Q: why? we can't use the test set to make decisions?

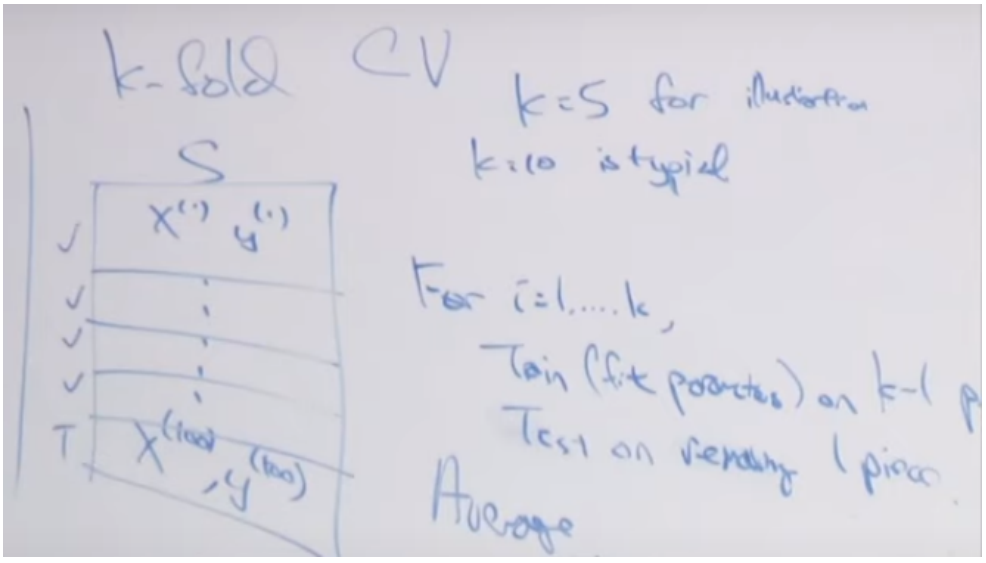
Aha: Because we want data the model never saw, like, on kaggle we have data to submit our precitions.

Aha: If you have a small dataset, and only in this case, you can use a strategy called k-fold cross validation to model selection.

Aha: k-fold consists in split out data in k pieces, train our data in k-1 and test with the that piece.

Q: Why k-1?

Aha: We use k-1 becanse, for k = 5, we loop trought the dataset five time, train in four and test in one. Each time, we change the test set, and in the end we take the avarange error.

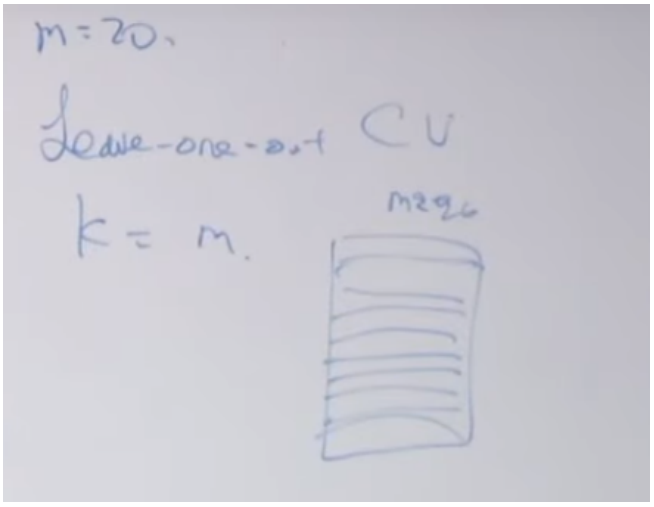


Huh: Doesn't understand that part about, "after choose your model", retrain the data in 100% of it. It means, wo holdout?

A good number for k is 10.

Aha: Using k-fold is very expensive, imagine you apply this to five models! so that's why we need to use in small data.

One extreme version of k-fold is leave-one-out. For this case, supose we have 20 exemples, we train in 19 and use 1 as test.



We can use thos for cases where n<100.

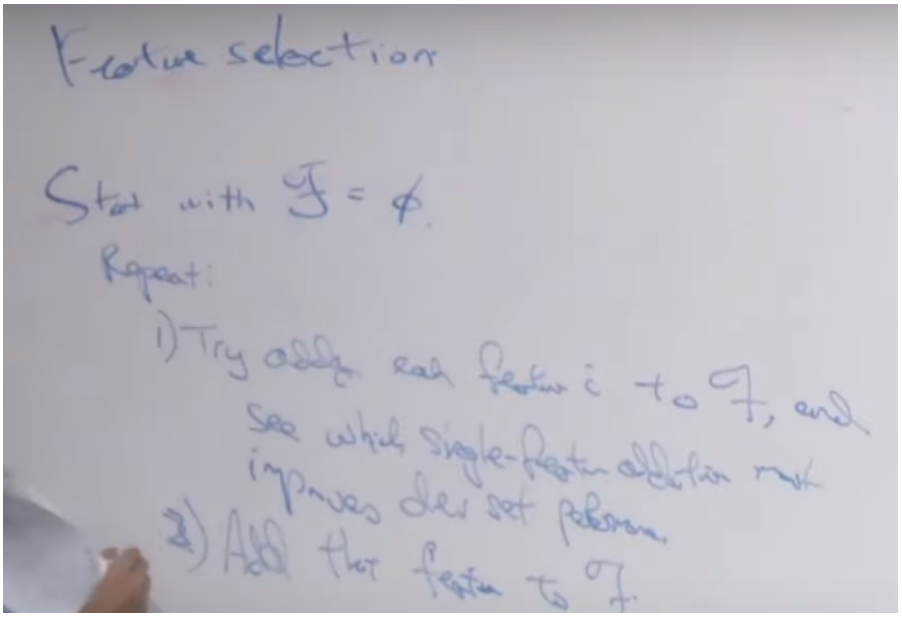
We don't use CV in Deep Learning because is not simple train a neural net, say 20 times!

Aha: We need fecture selection simply because not always all your features are important.

Aha: Another great way to prevent overfitting is selection a couple of important features

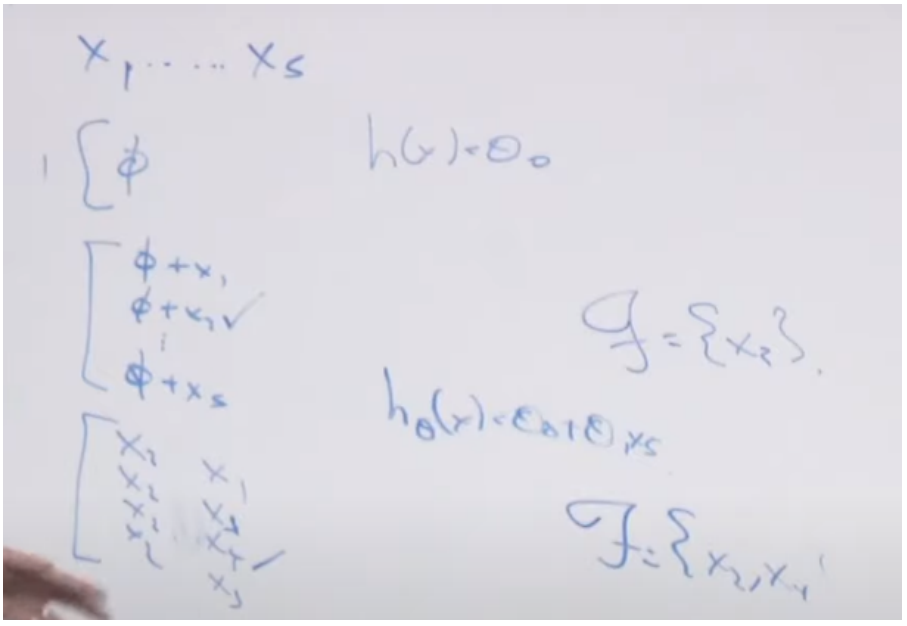
Not always is a great tool (feature selection), say in computer vision, where each pixel matters.

Intution: Supose that you want to predict if a car will broken, the data have all parts of the car, but in reallity you need just some parts of it, that's why we use feature selection.



A simple algorithm able to compute feature selection:

- start empty scrit
- loop trough each feature
- verify what feature improve the test set/dev set
- add that feature to our empy script



Q: why $\Phi + x_i$?

Aha: We combine features with the feacture we select before and check if the performance improves.

Vocab

- **Frequentist Stats**: Don't care about prior knoledge just about the present data
- **Basyan**: Look at the past and assume certain things.
- **MLE**: Way to find the best parameters that make our data most likely to happen.
- **Hold-Out Cross-Validation**: When you separe data in train/valid/test sets
-
-
-
-
-

More

- https://www.youtube.com/watch?v=-8s9KuNo5SA
- https://www.youtube.com/watch?v=glLNo1ZnmPA
- https://www.youtube.com/watch?v=wjILv3-UGM8
- https://nessie.ilab.sztaki.hu/~kornai/2020/AdvancedMachineLearning/Ng_MachineLearningYearning.pdf