(10th March, Week 2)

The Bias-Variance Tradeoff

Start with the following idea;

$$E_{test} = (E_{test} - E_{train}) + E_{train}$$

Terms **test error**, **approximation error**, **training error**. How does this help? If E_{approx} is small, then E_{train} is a good approximation to E_{test}.

So what does E_{approx} ("amount of overfitting") depend on?

- Tends to get smaller as 'n' gets larger
- Tends to grow as model gets more "complicated" (bigger tree in case of decision tree models)

Leads to a fundamental trade-off:

- E_{train} how small you can the train error vs.
- E_{approx} how well training error approximates the test error.

Ie, there is an antagonistic relationship between the two. "Cant get both of them to zero".

Bias and Variance

So with the above commentary complete, now we can further "decompose" E_{test} into;

$$E_{test} = (E_{test} - E_{train}) + (E_{train} - E_{best}) + E_{best}$$

Terms **test error**, **variance** (sensitivity to changes in traning data), **bias** (how low can we make the training error) and general **noise** (how low can any model make the test error). Which lands us into the crux of this trade-off idea.

So concretely for a tree with high depth for example;

- fit data well, bias low
- model changes if you change the data, variance high

And ofcourse vice versa if the depth is very low...

So how deep do we build a decision tree?

70/30 (training/testing split). Then do another 70/30 *within* training to training and validation set. We use the validation set to help choose the ideal depth of the tree.

$$\sum(\hat{y}) \neq \widetilde{y}$$

It's like a pseudo representation of your over E_{test}. "Unbiased approximation of test error". We use this to choose our "hyper parameters".

1 Hyper Parameters

In the context of decision tree models, a hyperparameter is a parameter that is set before the model is trained and whose value cannot be learned from the data. Hyperparameters control the behavior of the model and can be tuned to improve its performance.

For decision trees, some common hyperparameters include:

1. Maximum depth: This controls the maximum depth of the decision tree. A deeper tree can capture more complex relationships in the data, but may also be prone to overfitting.
2. Minimum samples per leaf: This controls the minimum number of samples required to be at a leaf node. A higher value will result in a simpler tree, but may sacrifice some predictive performance.
3. Split criterion: This controls the metric used to evaluate the quality of a split at each node. The two most common split criteria are Gini impurity and entropy.

Hyperparameter tuning is an important step in the machine learning process, as it can greatly impact the performance of the model. Techniques such as grid search, random search, and Bayesian optimization can be used to find the best hyperparameter values for a given problem.

As opposed to parameters

The rules the models use for classification...

Once the "best" hyper parameters are found

We go ahead and fully train on the combined training/validation set (to get the most out of the training data set) and evaluate on the test data set to get a "final" profiled model.

Optimization Bias

We can introduce a bias via tuning these hyper parameters... It's purely another name for overfitting. But more focused on "what" parameters we come up with. (multi choice test model example given at 50min)

- grows with the number of hyper parameter tunes we try
- shrinks quickly with the number of examples

Quick Notation

Given features in y_i, $\hat{y}_i$ is a prediction based on y_i. $\overline{y_i}$ represents some sort of mean representative of y_i. $\widetilde{y_i}$ represents the actual label of y_i. This is used for the traditional presentation of the bias and variance tradeoff (where $\widetilde{y_i} = \overline{y_i} + \epsilon$). Where we assume ϵ has a mean 0, and variance of something bounded like σ^2. (see 29min for more).

Quick word on IID

scikit-learn doesn't do any sort of randomisation when split the data set, needs to be done explicitly. I believe this is a direct assignment hint...