LECTURE 8

BOGDAN ICHIM

Bagging

Decision trees suffer from *high variance*. For example, if we randomly split the data into two parts and we fitted a decision tree on both halves, we may get quite different trees.

A *low variance* method should get better results if applied repeatedly to distinct data subsets. For examples, the linear model tends to have low variance if the N to P ratio is large (N = number of samples, P = number of predictors).

Bagging, also called **Bootstrap Aggregation** is a *general* procedure for reducing the variance of our machine learning model. In particular, it is very useful in the context of decision trees.

Central Limit Theorem

Let $X_i, i = \overline{1, N}$ be a set of N *independent* random variables and each X_i have an *arbitrary* probability distribution with mean μ and finite variance σ^2. Then, the mean random variable

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$$

is **normally distributed** $\bar{X} \sim N(\mu, \frac{\sigma^2}{\sqrt{N}})$, with $\mu(\overline{X}) = \mu, \sigma(\overline{X}) = \frac{\sigma}{\sqrt{N}}$ and $var(\overline{X}) = \frac{\sigma^2}{N}$.

Corollary: Averaging the set of observations reduces variance.

Main Idea:

(1) Take many samples from the population.

(2) Build separate prediction models using each training sample.

(3) "Average" the resulting predictions.

In other words we consider B separate training sets, we compute B models $\hat{f}_i(x), i = \overline{1, B}$ and "average" them in order to obtain a single low-variance model, that is:

$$\hat{f}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}_i(x)$$

Remark: In general, we do not have access to multiple training sets!

Bootstrap = takes repeated samples from the (single) *training* data set.

Bagging = generate B different bootstrapped training data sets, train our model on the B data sets and "average" all of their predictions.

- Bagging applied to **regression** models (in particular to regression trees: construct B regression trees using the B bootstrapped data set and average the resulting predictions.
- Bagging applied to **classification** models (in particular to classification trees): same as above, but instead of averaging the prediction, we record the class predicted by each of the B trees and take a **plurality vote**: the overall prediction is the most commonly occurring class among the B predictions.

Remark: Plurality voting on categorical data is similar to computing the mean on continuous data.

Note: Plurality vote is **NOT** majority vote!

Random Forest

Averaging many highly correlated variables does not lead to an equivalently large reduction in variance as averaging many uncorrelated variables. (Note that in the Central Limit Theorem the variables are assumed to be independent!)

The aim of random forests is to **decorrelate** its trees. As in bagging, we build B decision trees on the bootstrapped training sets. However, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of P predictors.
The split is allowed to use only one of those m predictors and a fresh sample of m predictors is taken at each split..

Usual values for m: $\sqrt{P}$, $\frac{P}{2}$ etc.

If $m = P$ then random forest is equivalent to bagging in terms of generating the subsets.

Example: Suppose that there is one very strong predictor in the dataset, among other moderately strong predictors. Then, in the collection of bagged trees, most or all of the trees will attempt to use this very strong predictor in the first split. This leads to the trees having a similar structure and thus their predictions will be highly correlated. In this care, bagging will be very similar to a single tree!

Remarks:

- A common problem in modelling is that many features may be simply noise. Random forest does a good job at eliminating this noise from our data set.
- Small m is helpful when we have a large number of correlated predictors.

Boosting

Boosting does not involve bootstrapping multiple samples - each tree is fitted on a modified version of the original data set, using information from previously grown trees.

Boosting **learns slowly**! Given the current model, we fit a decision tree to the residuals from the model. In other words, we fit a tree using the current residuals, rather than the outcome Y, as response. We then add this new decision tree into the fitted function in order to update the residuals.

Usual tuning parameters:

(1) The **number of trees** B. Unlike bagging and random forest, boosting can overfit if B is too large.

(2) The **interaction depth** (the *number of splits* in each tree), which controls the interaction order of the model, since d splits can involve at most d variables. Oft as not, $d = 1$ works well.

(3) The **shrinkage parameter** $\lambda > 0$. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001. This works in conjunction with the number of trees. Small λ means large B and large λ implies small B.

The Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

$$\text{Accuracy} = \text{Rate of Correct Predictions} = \frac{TP + TN}{TP + FP + FN + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP} = 1 - \text{False Discovery Rate}$$

$$\text{False Discovery Rate} = \frac{FP}{TP + FP} = 1 - \text{Precision}$$

$$\text{Negative Predicted Value} = \frac{TN}{TN + FN} = 1 - \text{False Omission Rate}$$

$$\text{False Omission Rate} = \frac{FN}{TN + FN} = 1 - \text{Negative Predictive Value}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} = 1 - \text{Specificity } (= \textbf{Recall})$$

$$\text{Specificity} = \frac{TN}{TN + FP} = 1 - \text{Sensitivity}$$

Remark: For highly unbalanced datasets, the *Accuracy* metric does not offer enough relevant information. Instead, we use *Precision* and *Recall*. The intuition is:

- **Precision** is the fraction of **selected** samples which are **relevant**. In other words, it's the proportion of samples our model says was relevant and they were actually relevant. Precision can be thought of as a measure of a classifiers *exactness*. A low precision can also indicate a large number of False Positives.

- **Recall** is the fraction of the **relevant** samples which are **selected**. Recall can be thought of as a measure of a classifiers *completeness*. A low recall indicates a large number of False Negatives.

We also introduce the **F1 score**. This is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases by weighting the other two metrics.

We can say that the F1 score conveys the balance between the Precision and the Recall:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

For the confusion matrix of a multiclass problem, we have:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{12} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \Rightarrow \text{Accuracy} = \frac{\sum_{i=1}^{n} a_{ii}}{\sum_{i,j=1}^{n} a_{ij}}$$