Lecture Notes

Random Forests

You are familiar with decision trees. Now it's time to learn about **random forests**, which are collections of decision trees. The great thing about random forests is that they almost always outperform decision trees in terms of accuracy.

Ensembles

An ensemble is a group of things viewed as a whole rather than individually. In ensembles, a **collection of models** is used to make predictions instead of individual models. Arguably, the most popular in the family of ensemble models is the random forest. A random forest is an ensemble made using a **combination of numerous decision trees.**

For an ensemble to work, each model of it should comply with the following conditions:

1. Each model should be **diverse**. Diversity ensures that the models serve **complementary** purposes; this means the individual models make predictions **independent of each other.**
2. Each model should be acceptable. **Acceptability** implies that each model is at least **better than a random model**.

Consider a binary classification problem where the response variable is either 0 or 1. You have an ensemble of three models, where each model has an accuracy of 0.7, i.e. it is correct 70% of the time. The following table shows all the possible cases that can occur while classifying a test data point as 1 or 0. The column to the extreme right shows the probability of each case.

Case	Result of Each Model			Result of the Ensemble	Probability
	m1	m2	m3		
1	Correct	Correct	Correct	**Correct**	0.7*0.7*0.7 = 0.343
2	Correct	Correct	Incorrect	**Correct**	0.7*0.7*0.3 = 0.147
3	Correct	Incorrect	Correct	**Correct**	0.7*0.3*0.7 = 0.147
4	Incorrect	Correct	Correct	**Correct**	0.3*0.7*0.7 = 0.147
5	Incorrect	Incorrect	Correct	**Incorrect**	0.3*0.3*0.7 = 0.063
6	Incorrect	Correct	Incorrect	**Incorrect**	0.3*0.7*0.3 = 0.063
7	Correct	Incorrect	Incorrect	**Incorrect**	0.7*0.3*0.3 = 0.063
8	Incorrect	Incorrect	Incorrect	**Incorrect**	0.3*0.3*0.3 = 0.027

Figure 1- Ensemble of three models

In the table above, there are four cases each where the decision of the final model (ensemble) is either right or wrong. Let's assume that the probability of the ensemble being right is p and the probability of the ensemble being wrong is q.
For the data in the table, p and q can be calculated as follows:
$p = 0.343 + 0.147 + 0.147 + 0.147 = 0.784$
$q = 0.027 + 0.063 + 0.063 + 0.063 = 0.216 = 1 - p$
So you can see how an ensemble of just three models provides an accuracy boost from 70% to 78.4%. In general, the higher the number of models, the higher is the accuracy of an ensemble.

Creating a Random Forest

Random forests are created using a special ensemble method called **bagging**. Bagging stands for **bootstrap aggregation. Bootstrapping** means creating bootstrap samples from a given data set. A bootstrap sample is created by **sampling** the rows of a given data set **uniformly** and **with replacements**. A bootstrap sample typically contains about 30-70% of the data from the data set. **Aggregation** implies combining the results of different models present in the ensemble.

Random forests is an ensemble of many decision trees. A random forest is created in the following way:

1. Create a **bootstrap sample** from the training set.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
1	22	54	56	21	78	82	53	67	56	0
→ 2	34	82	34	84	67	23	76	32	24	1
→ 3	12	38	13	54	74	57	86	63	89	1
4	32	93	26	21	24	34	91	34	97	0
5	62	23	64	67	68	67	24	28	52	1
→ :	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:
→ 98	23	19	86	89	57	21	53	54	44	1
99	56	12	54	90	25	86	78	34	38	0
→ 100	15	94	21	24	15	56	23	79	92	0

Figure 2 - Training set

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	Y
2	34	82	34	84	67	23	76	32	24	1
3	12	38	13	54	74	57	86	63	89	1
78	32	93	26	21	24	34	91	34	97	0
98	23	19	86	89	57	21	53	54	44	1
100	15	94	21	24	15	56	23	79	92	0

Figure 3- Bootstrap sample

2. Now construct a decision tree using the bootstrap sample. While splitting a node of the tree, only consider a **random subset of features**. Every time a node has to split, a different random subset of features will be considered.
3. Repeat steps 1 and 2 *n* times to construct *n* trees in the forest. Remember, each tree is constructed independently, so it is possible to construct them parallel to each other.
4. While predicting a test case, each tree predicts individually, and the final prediction is given by the majority vote of all the trees.

There are several advantages of a random forest:

1. A random forest is more **stable** than any single decision tree because the results get averaged out; it is not affected by the instability and bias of an individual tree.
2. A random forest is **immune to the curse of dimensionality** since only a subset of features is used to split a node.
3. You can **parallelise** the training of a forest since each tree is constructed independently.
4. You can calculate the OOB (out-of-bag) error using the training set; the OOB provides a good estimate of the performance of a forest on unseen data. Hence, there is no need to split the data into training and validation; you can use all of it to train the forest.

OOB (Out-of-Bag) Error

The OOB error is calculated by using each observation of the training set as a test observation. Since each tree is built on a bootstrap sample, each observation can be used as a test observation by the trees that did not have it in their bootstrap sample. All these trees predict on this observation, and you get an error for a single observation. The final OOB error is calculated by calculating the error on each observation and aggregating it.

It turns out that the OOB error is as good as a **cross-validation error**.

Time taken to build a forest

To construct a forest of *S* trees on a data set that has *M* features and *N* observations, the time taken will depend on the following factors:

1. The **number of trees**: The time is directly proportional to the number of trees. But, this duration can be reduced by creating the trees parallel to each other.
2. The **size of the bootstrap sample**: Generally, the size of a bootstrap sample is 30-70% of *N*. The smaller the size, the faster it takes to create a forest.
3. The **size of the subset of features** while splitting a node: Generally, this is taken as $\sqrt{M}$ in classification and M/3 in regression.

Random forests lab

To create a random forest in R, you use the 'randomForest' library.
The following R code is pretty self-explanatory.

```r
library("randomForest")

# set seed for reproducible results
set.seed(1)

# 1. Train model on train data
rf_model <- randomForest(label ~ .,          # label is the response variable
                         data = train,       # use the training data to train
                         importance = TRUE,  # track the importance of each variable to plot later
                         ntree = 1000,       # number of trees
                         mtry = 3,           # subset of features to consider while splitting a node
                         do.trace = TRUE,    # print information on console while training
                         proximity = FALSE   # should proximity measure among the rows be calculated?
                        )

# 2. Make predictions on test data
rf_predictions <- predict(rf_model, test)

# 3. Evaluate the model confusion matrix
library("caret")
confusionMatrix(test$label, rf_predictions)

# feature importance plot
varImpPlot(rf_model)
```

Figure 4- Random forest in R