Random Forest Classifier Summary

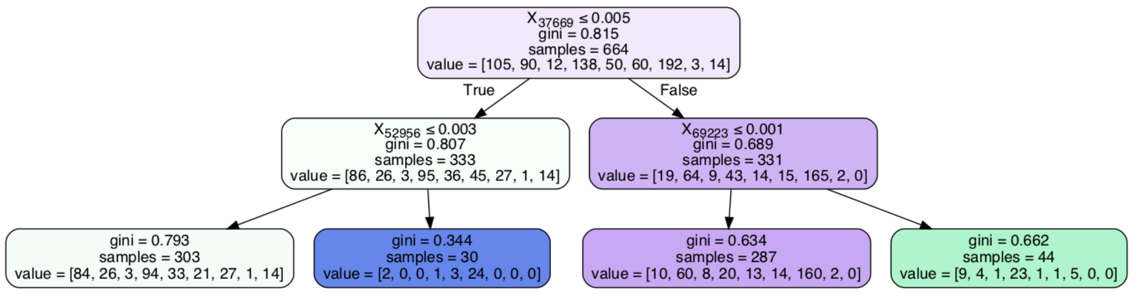
Cory Weinfeld

In this document we provide a discussion of a random forest classifier. The motivation behind this is to elaborate on the behind the scenes process of the accompanying Jupyter notebook consisting of a solution to the ”Personalized Medicine: Redefining Cancer Treatment” competition on Kaggle.

To utilize a random forest classifier we first need a matrix with features to classify. To obtain such a matrix we use a bag of words approach on the text documents. What a bag of words does is assigns a fixed integer value to every word across all documents. We then obtain a dictionary of words. Now for a document with ID i we take the frequency of each word and store it in the ijth position of a matrix, where j represents the fixed integer value of the word in the dictionary [1]. It is important to note that this will be a sparse matrix since each text document will contain some amount of unique words. At this point we have a matrix which can tell us about the feature of a text document consisting of total word counts for any given word, but to be more useful we would like to determine the density of a given word in a document as well as its uniqueness to that document.

We now transform our matrix using a term frequency – inverse document frequency (tf-idf) method to account for these desired features. To compute the tf-idf of a word in a document we take tf(w) to be the amount of times w appears in a document and $idf(w, d) = \log\left(\frac{n}{df(w,d)}\right)$ where n is the total number of documents and $df(w, d)$ represents the amount of documents that contain w, and finally we take tf-idf to be $tf(w) \times idf(w, d)$ [1]. In a sense when we create the tf-idf matrix we take our matrix obtained from a bag of words and multiply the nonzero values by their respective idf(w,d) value. Once we obtain a matrix consisting of tf-idf values we will have a useful feature that to consider in a random forest classifier.

Before we discuss a random forest it is first useful to introduce the notion of a decision tree. A decision tree is essentially a process which separates data based on some quality. In our case a decision tree will separate data based on the tf-idf score of the various words in the dictionary described above. The decision tree will look for the ”best” split of the data which attempts to preserve a purity of the classes. Below in figure 1 we demonstrate an example of a decision tree constructed using a subset of our data [3].



$X_{37669} \le 0.005$
gini = 0.815
samples = 664
value = [105, 90, 12, 138, 50, 60, 192, 3, 14]
True
False
$X_{52956} \le 0.003$
gini = 0.807
samples = 333
value = [86, 26, 3, 95, 36, 45, 27, 1, 14]
$X_{69223} \le 0.001$
gini = 0.689
samples = 331
value = [19, 64, 9, 43, 14, 15, 165, 2, 0]
gini = 0.793
samples = 303
value = [84, 26, 3, 94, 33, 21, 27, 1, 14]
gini = 0.344
samples = 30
value = [2, 0, 0, 1, 3, 24, 0, 0, 0]
gini = 0.634
samples = 287
value = [10, 60, 8, 20, 13, 14, 160, 2, 0]
gini = 0.662
samples = 44
value = [9, 4, 1, 23, 1, 1, 5, 0, 0]

Figure 1: Decision Tree Process

Figure 1 is an example of a decision tree of depth 2, meaning it splits the data twice resulting in four final nodes. We see that the classifier determined that the first split should occur where word 37669 occurs in the dictionary and it determines in each document if that word has a tf-idf score less than or equal to 0.005. In this case it split the data nearly in half. We also see that it measures the data by assigning it a gini coefficient which is computed by taking

$$\text{gini} = 1 - \sum_{i=1}^{9} \left(\frac{x_i}{N}\right)^2$$

, where x_i is the amount of samples in class i and N is the total sample size [4]. We can think of the gini coefficient as a measure of how "pure" each class is, that is a high gini coefficient means the data is more distributed throughout the classes at that node and a low gini coefficient means one class contains much of the data. Clearly a decision tree is successful when each node has a low gini coefficient. In theory a decision tree will continue to split the data until each node has a gini coefficient of 0, but in practice we tend to stop expanding the tree well before this occurs or else we risk over-fitting the data.

A major drawback of creating only a single decision tree is that they often tend to over-fit the data. One can observe that what makes for a good split in the training data of a classifier might not make for a good split of the testing data. To remedy this we introduce what's called a random forest.

With a random forest classifier what occurs in the classification process is we take our training data, which in this case is the tf-idf matrix where each row of corresponds to a text document, and repeatedly take a random subset of it (with replacement) and create a decision tree for each random subset. Moreover each decision tree, at each split, selects a random subset of the features to further split. What we end up with is a large amount of decision trees (a so-called "forest") where each classifies the data in a slightly (or perhaps not so slightly) different way. [2]

When it comes to classifying new data what will occur is that each tree in the random forest will vote on what class it thinks the data belongs too. The classifier then takes a majority rules approach and predicts the class in that way. In theory this approach should avoid over-fitting the data, and that makes it one of the more prominent classifiers currently in use.

References

[1] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[2] Wikipedia contributors. *Random forest — Wikipedia, The Free Encyclopedia.* [Online; accessed 26-November-2018]. 2018. URL: `https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=869972790`.

[3] R.N. Brown. *Creating and Visualizing Decision Trees with Python.* URL: `https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176` (visited on 11/26/2018).

[4] Madhu Sanjeevi. *Chapter 4: Decision Trees Algorithms.* URL: `https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1` (visited on 11/26/2018).