DSC 255: Machine learning

Homework 9

Mathematical and conceptual exercises

1. Decision trees have great *expressive power*: they can represent *any* decision boundary, by carving up the space to a fine enough granularity. Which of the other methods we've studied are similarly expressive? For each, give a brief explanation.

   (a) Linear classifiers.

   (b) Support vector machines with a quadratic kernel.

   (c) Nearest neighbor classifiers.

   (d) Classifiers based on Gaussian generative models.

2. Let's say our training set consists of n data points in d-dimensional space. At the top node of the tree, we have to pick a split, which involves choosing (i) a feature and (ii) a split value along that feature. How many possibilities do we need to try out, roughly? *Hint:* How many ways to choose a feature? And once we have a feature, what is the effective number of splits?

3. In order to grow a decision tree, we need a measure of how *pure* or *impure* (in terms of labels) a node is. A popular measure of this is the *Gini impurity index.* If there are two labels, and a node has p fraction of one label and $1-p$ fraction of the other, the Gini impurity index is $2p(1-p)$. Notice that this is maximized when $p = 1/2$.

   What is the Gini impurity index for a node in which 20% of the points have one label while 80% have the other label?

4. *Working with weighted data.* Boosting requires the weak learning algorithm to work with training data in which each point $(x^{(i)}, y^{(i)})$ has a positive *weight* $\lambda_i > 0$. Intuitively, a weight of two would be equivalent to having two copies of that data point.

   How would you incorporate weights into the following learning algorithms without explicitly making copies of data points?

   (a) Decision trees.

   Hint: Previously, we measured the uncertainty at any node by looking at the fraction of points with each label (call these $p_1, \ldots, p_k$ if there are k labels). Now we need to compute these fractions differently, taking weights λ_i into account.

   (b) Gaussian generative models.

   Hint: For each class j, we need the weight of that class (π_j), the mean (μ_j), and the covariance matrix (Σ_j). Now we need to compute them differently, taking weights λ_i into account.

   (c) Support vector machines.

   Hint: Can you incorporate the weights λ_i into the objective function for soft-margin SVM?

   Give a brief explanation in each case.

5. *Convergence behavior of boosting.* Suppose we run boosting using weak classifiers from some class H (such as decision stumps), and suppose that on each round, the weak learner always returns a weak classifier whose error rate on the weighted data (for that round) is at most $1/2 - \epsilon$, for some $\epsilon > 0$. Which of the following is true? Give a brief explanation in each case.

   (a) Boosting converges to a final classifier with zero test error.

   (b) Boosting converges to a final classifier with zero training error.

   (c) Boosting's final classifier belongs to class H.

6. *Random forests versus boosted decision trees.* Which of the following is a benefit of random forests over boosted decision trees? Explain briefly in each case.

   (a) The trees can be trained in parallel.

   (b) Each individual tree is more highly optimized.

   (c) Each individual tree has better accuracy.

Programming exercises

Before beginning this week's lab, download the data set `mini-data.txt` from the course website.

7. *A toy 2-d data set for decision trees and boosting.* Obtain the data set `mini-data.txt` from the course webpage. Each line has a two-d data point followed by a label (0 or 1).

   (a) Plot this data to see what it looks like. Show this plot in your writeup.

   (b) Use `sklearn.tree.DecisionTreeClassifier` to fit a decision tree to the data. What stopping criterion did you use?

   (c) Display the tree using `sklearn.tree.plot_tree`.

   (d) Fit boosted decision stumps to this data using `sklearn.ensemble.AdaBoostClassifier`. Use a relatively small number of stumps, and display each of them.

   (e) Give a table showing how accuracy on the training data improves as each successive stump is added.

8. *Credit card fraud data.* Download the data set at

   `https://www.kaggle.com/mlg-ulb/creditcardfraud`.

   This data set has details of 284,807 credit card transactions, some of which are fraudulent. Each transaction is represented by 28 features (scrambled using PCA as a primitive kind of anonymization), and has a corresponding label (1 is fraudulent and 0 is legitimate).

   (a) How many of the transactions are fraudulent? Why might this be problematic when learning a classifier?

   (b) Downsample the legitimate transactions to make the data set more balanced. For instance, you might try to make sure that each class (legitimate or fraudulent) makes up at least 25% of the data set.

   (c) Fit three kinds of classifier to the data:

   - decision tree
   - boosted decision stumps

- random forest

In each case, use cross-validation to estimate the confusion matrix. For doing the cross-validation, you might find it helpful to use `sklearn.model_selection.cross_val_predict`. For computing the confusion matrix, you can use `sklearn.metrics.confusion_matrix`. If you want a slick display of the confusion matrix, you might try `sklearn.metrics.ConfusionMatrixDisplay`.