CS 189: Introduction to Machine Learning - Discussion 9

1. Maximum Entropy Distribution

Suppose we have a discrete random variable that has a Categorical distribution described by the parameters $p_1, p_2, \ldots, p_d$. Recall that the definition of entropy of a discrete random variable is

$$H(X) = E[-\log p(X)] = -\sum_{i=1}^{d} p_i \log p_i$$

Find the distribution (values of the p_i) that maximizes entropy. (Hint: remember that $\sum_{i=1}^{d} p_i = 1$. Don't forget to include that in the optimization as a constraint!)

Solution:

For simplicity, assume that log has base e, i.e. log = ln (the solution is the same no matter what base we assume). The optimization problem we are trying to solve is:

$$\begin{aligned} \underset{\vec{p}}{\text{argmin}} \quad & \sum_{i=1}^{d} p_i \log p_i \\ \text{s.t.} \quad & \sum_{i=1}^{d} p_i = 1 \end{aligned}$$

Formulating the Lagrangian, we get

$$\mathcal{L}(\vec{p}, \lambda) = \sum_{i=1}^{d} p_i \log p_i + \lambda \left(1 - \sum_{i=1}^{d} p_i\right)$$

Taking the derivative w.r.t. p_i and λ:

$$\begin{aligned} \frac{\partial}{\partial p_i} \mathcal{L}(\vec{p}, \lambda) &= \log p_i + \frac{p_i}{p_i} - \lambda \implies \lambda - 1 = \log p_i \\ \frac{\partial}{\partial \lambda} \mathcal{L}(\vec{p}, \lambda) &= \sum_{i=1}^{d} p_i = 1 \end{aligned}$$

This says that $\log p_i = \log p_j, \forall i, j$, which implies that $p_i = p_j, \forall i, j$. Combining this with the constraint, we get that $p_i = \frac{1}{d}$, which is the uniform distribution.

2. Decision Trees

Recall that training a decision tree requires looking at every feature to find the best split, where the best split greedily maximizes the information gain. The information gain is defined as

$$H - \left[\frac{n_1 H_1 + n_2 H_2}{n_1 + n_2}\right]$$

where H is the entropy at the current node, H_1 is the entropy at the "left" split, and H_2 is the entropy at the "right" split. n_1 and n_2 are the number of data points at the "left" and "right" splits.

(a) What are good values to choose to test the splits?

Solution:
Imagine we are deciding to split on a specific real-valued feature. Obviously, we should choose every value of the data to be the threshold for the split. Choosing a split value between two values in our data will give the same splitting of the data. For instance, imagine that we have data:

$$\begin{bmatrix} 5 & 0 \\ 6 & 0 \\ 7 & 1 \end{bmatrix}$$

where the first column is the features, and the second column is the labels (we have three data points). Note that splitting on 6.5 and 6.3 will result in the same splits. Thus, we want to choose each feature value for a split.

(b) What is the running time for the naive approach to finding the best split (just finding the split, not training the entire tree)?

Solution:
We must search through every possible split to find the best one, and there are dn of them, where d is the dimensionality of our data and n is the number of data points. For every split value, we must walk through all n data points and classify them accordingly. We can calculate the information gain from these two new subsets of our data in linear time. Thus, we have $\mathcal{O}(dn^2)$.

(c) What is a smarter way to search for the best split, and what is the running time of this?

Solution:
Before we start scanning through a feature, we can sort the data with respect to that feature value. Then, every time we choose a new split value, only one data point will be classified differently. Calculating the new entropies can be done in constant time. Assuming we use a comparison based sorting algorithm, our new running time is $\mathcal{O}(dn \log n)$.

3. Random Forests

(a) In terms of the bias-variance tradeoff, where does a single deep decision tree fall?

Solution:
A decision tree that is grown very deep will have low bias, but high variance. This commonly leads to overfitting on the training data.

(b) How does bootstrap aggregating, or bagging, help? How are predictions made using bagging?

Solution:
Bagging for decision trees trains several decision trees (often smaller trees, called decision stumps) on a training set that is generated by sampling with replacement from the original training set. Predictions for new data points are done by querying the entire ensemble and predicting by the majority (or plurality, for multiclass) vote. The variance of the resulting ensemble is lower than a single decision tree.

(c) How do random forests extend bagging? What problem do random forests help with?

Solution:
Random forests, at each node, only allow a random subset of features to split on. This prevents all of the trees in the ensemble from picking the same very good splits, and becoming highly correlated.