CSC 446 Notes: Lecture 4

Typed by Rahman Lavaee

January 31, 2012

1 Model Selection Review

- **MLE**
$$P(X_{N+1}) \sim \theta^* = \underset{\theta}{\operatorname{argmax}}\, P(X_1^N|\theta)$$
- **Bayesian**
$$P(X_{N+1}) = \frac{1}{Z}\int P(X_{N+1}|\theta)P(\theta|X_1^N)\,\mathrm{d}\theta$$
- **MAP**
$$P(X_{N+1}) \sim \theta^* = \underset{\theta}{\operatorname{argmax}}\, P(X_1^N|\theta)P(\theta)$$

2 Minimum Description Length

Consider the classification problem. As in Naive Bayes Classifier, we still make the assumption of:

$$P(Y|\mathbf{X}) \propto P(Y)\prod_i P(\mathbf{X}_i|Y)$$

Now, suppose we let the prior probability distribution $P(\theta)$ to be a Drichlet distribution. No matter what the hyperparamaters are, this distribution does not make any preference over the number of related features. This does not conform to the Minimum Description Length intuition which tells us that smaller number of features gives us a simpler and more compressed model. Therefore we propose the following prior distribution over θ:

Suppose θ is a model parameter which is based on a subset of the features $\{\theta_1, \theta_2, ..., \theta_I\}$. Then we would like to define the prior as:

$$P(\theta) = c\exp(-I)$$

where c is a normalization factor.

So, The number of bits to encode this model is simply the negative log probability, which is:

$$\log P(\theta) = c'|\theta|$$

We will ultimately come up with the Minimum Description Length method:

2.1 MDL Estimation

Now let us apply the minimum description length priors to the Model Selection methods reviewed above:

MLE uses all the features regardless of the model and therefore the MDL priors do not make any change on its outcome.

In **Bayesian**, if we estimate the integral with a sum, we come up with:

$$P(X_{N+1}) = \sum_{\theta\subseteq\{features\}} P(X_1^N|\theta)e^{-|\theta|}$$

However, to solve this, one needs examine all 2^N possible subsets, which is impractical.

In **MAP**, the equation takes the form of:

$$\theta^* = \underset{\theta \subseteq \{features\}}{\operatorname{argmax}} \; P(X_1^N|\theta)e^{-|\theta|}$$

Here, we still have the problem of 2^N subsets during training, but we need only consider one model when classifying new points. During training, we can approximate the max operation by sorting the features according to their mutual information with the data and considering the most correlated features, for each number of features. This is based on the fact that r most correlated features is preferred over any r-size subset of features. This way, we only need to update $|Features|$ values and simply pick the biggest one.

Taking logarithm of the expression, we have:

$$\theta^* = \underset{\theta \subseteq \{features\}}{\operatorname{argmax}} \; \log P(X_1^N|\theta) - |\theta|$$

However, from experience we know that there should be a hyperparamter λ changing the expression to:

$$\theta^* = \underset{\theta \subseteq \{features\}}{\operatorname{argmax}} \; \log P(X_1^N|\theta) - \lambda|\theta|$$

Now, how can we get the best possible λ? We argue that by looking at the performance of the model, we can choose the best λ. It is actually the one that maximizes the performance of the held out data.

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{N} I(Y_i = Y_i^*)$$

Where

$$Y^* = \underset{Y}{\operatorname{argmax}} \, P(Y) \prod P(X_{N+1}|Y_i)$$

3 Decision Tree

Another method of classification is using Decision Trees. Consider the Voting problem again. The decision tree starts by querying on a feature, and then going to the left or right subtree according to the answer. The left and the right decision trees are constructed recursively. The leaves of the decision tree will be the classification over the data. (*democrat* or *republican* in this example)

Now, how can we decide on the tree structures and nodes. Suppose we have n boolean valued features and a set of data.

We know that the number of possible trees might be not only exponential, but even super-exponential. This urges us to use a heuristic method to construct the best tree.

A reasonable approach is to use the most relevant features for the higher level nodes. This way, we choose the root node as the feature with the biggest mutual information with data.

$$root = \underset{i}{\operatorname{argmax}} \, I(Y, X_i)$$

Then, the data is split over the right and left hand subtrees according to the their value for this feature. The procedure is recursively executed on both subtrees until all features are used in the tree and the value label of each single data point remains on the leaves.

Now to classify another point, one needs to take a path in the tree to reach to a label. But now, we just used all the features to distinguish between different data points. This is literally using all the information (whether really relevant or not) and there is the problem of **overfitting**. To solve this problem, we can prune the tree from below, iteratively, look at the performance of the system, and pick the best pruning. This way, we can again think of the tree as a model and try to pick the model which gives us the minimum description length. So, we are motivated to solve:

$$\underset{tree}{\operatorname{argmax}} \prod_n P(X_n|tree)2^{-|tree|}$$

3.1 Decision Trees vs. Naive Bayes

Recall that we made this assumption for Naive Bayes :

$$P(X_1, X_2|Y) = P(X_1|Y)P(X_2|Y)$$

Now, suppose this is not the case. Consider the following data for *voting problem*.

X_1	X_2	Y
N	N	D
N	Y	R
Y	N	R
Y	Y	D

In this example, we have: $I(X_1, Y) = 0$ and $I(X_2, Y) = 0$ and $P(X_1, X_2|Y) \neq P(X_1|Y)P(X_2|Y)$. Now suppose we want to use Naive Bayes method. We have:

$$P(Y|X_1^N) = \frac{1}{Z}P(Y)\prod_n P(X_n|Y)$$

Where Z is a normalization factor. Consider the decision surface where $P(Y|X_1^N) = \frac{1}{2}$. This surface has the equation:

$$\frac{1}{2} = \log Z + \log P(Y) + \sum_n \log P(X_N|Y)$$

The first two terms on the righthand side are constants with respect to X_1^N. Now consider the last term. Now:

$$c = \sum_n \log P(X_n|Y = 0)$$

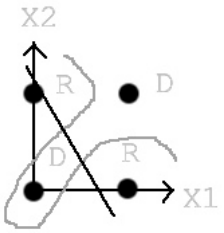
(On the decision surface $P(X_i|Y = 0) = P(X_i|Y = 1)$. So, we can simply let Y be zero.)

Now we propose a linear model for every term.

$$c = \sum_n c_n + X_n\theta_n$$

By choosing $c_n = \log P(X_n = 0|Y = 0)$ and $\theta_n = \log \frac{P(X_n=1|Y=0)}{P(X_n=0|Y=0)}$ this gives us the same value of: $\log P(X_n|Y = 0)$ for both $X_n = 1$ and $X_n = 0$.

Thus, this Naive Bayes Model will specify a Linear Decision Surface like the black line in the graph below. Therefore, it does not make the right decision for two data points.



But the optimal classification surface must look like the gray curve.

Now suppose we use decision tree method for this dataset. Since the mutual information for both features is equal to zero, there is no preference over the feature to be chosen as root. The complete decision tree is a complete binary tree with height two. The classification will consist of four square regions, one for every data point. Now suppose we prune one of the subtrees. Then we would come up with assigning one data to two adjacent squares, which is still not desired, but better than the linear decision surface.

The good point about pruned trees is that even if there has been no training data point with the features equal to those of the test data, we can still classify that based on its most informative features. In Naive Bayes, however, we had the problem of division by zero:

$$P(Y|X_1^N) = \frac{C(Y, X_1^N)}{C(X_1^N)}$$

4 Brief on Bayes Networks

Generally any feature can effect any other feature. However, in some problem domains we can limit the influences. The notion of *Bayes Networks*, *Belief Networks* or *Directed Graphical Models* is based on limiting these effects. Suppose we have a directed acyclic graph with its set of vertices equal to set of features in our model.

In general, we have:

$$P(X_1, ..., X_N) = \prod_{n=1}^{N} P(X_n|par(X_n))$$

where X_i's are any number of vertices in the graph and $par(X)$ is the set of all parents of vertex X.(those vertices pointing directly to X via a single edge).

Let us look at some simple examples of the Bayes Networks:

- $A \rightarrow B$

  $$P(A, B) = P(A)P(B|A) = P(B)P(A|B)$$

- $A\ B$

  $$P(A, B) = P(A)P(B)$$

  A and B are independent ($A \perp\!\!\!\perp B$).

- $A \rightarrow B \rightarrow C$

  $$P(A, B, C) = P(A)P(B|A)P(C|B)$$
  $$P(C, A|B) = P(C|B)P(A|B)$$

  C and A are independent given B ($A \perp\!\!\!\perp C|B$).

- $A \rightarrow B \leftarrow C$

  $$P(A, B, C) = P(A)P(B|A, C)P(C)$$
  $$P(A, C) = P(A)P(C)$$

  C and A are independent ($A \perp\!\!\!\perp C$).