Naïve Bayes

Naive Bayes is a Machine Learning algorithm for the ``classification task". It make the substantial assumption (called the Naive Bayes assumption) that all features are independent of one another, given the classification label. This assumption is wrong, but allows for a fast and quick algorithm that is often useful. In order to implement Naive Bayes you will need to learn how to train your model and how to use it to make predictions, once trained.

Training (aka Parameter Estimation)

The objective in training is to estimate the probabilities $P(Y)$ and $P(X_i|Y)$ for all $0 < i \leq m$ features. We use the symbol $\hat{p}$ to make it clear that the probability is an estimate.

Using an MLE estimate:

$$\hat{p}(X_i = x_i|Y = y) = \frac{(\text{\# training examples where } X_i = x_i \text{ and } Y = y)}{(\text{\# training examples where } Y = y)}$$

Using a Laplace MAP estimate:

$$\hat{p}(X_i = x_i|Y = y) = \frac{(\text{\# training examples where } X_i = x_i \text{ and } Y = y) + 1}{(\text{\# training examples where } Y = y) + 2}$$

The prior probability of Y trained using an MLE estimate:

$$\hat{p}(Y = y) = \frac{(\text{\# training examples where } Y = y)}{(\text{\# training examples})}$$

Prediction

For an example with $\mathbf{x} = [x_1, x_2, \dots, x_m]$, estimate the value of y as:

$$\hat{y} = \underset{y=\{0,1\}}{\arg\max}\ \log \hat{p}(Y = y) + \sum_{i=1}^{m} \log \hat{p}(X_i = x_i|Y = y)$$

Note that for small enough datasets you may not need to use the log version of the argmax.

Theory

In the world of classification when we make a prediction we want to chose the value of y that maximizes $P(Y = y|\mathbf{X})$.

$$\begin{aligned}
\hat{y} &= \underset{y=\{0,1\}}{\arg\max} P(Y = y|\mathbf{X} = \mathbf{X}) && \text{Our objective} \\
&= \underset{y=\{0,1\}}{\arg\max} \frac{P(Y = y)P(\mathbf{X} = \mathbf{x}|Y = y)}{P(\mathbf{X} = \mathbf{x})} && \text{By bayes theorem} \\
&= \underset{y=\{0,1\}}{\arg\max} P(Y = y)P(\mathbf{X} = \mathbf{x}|Y = y)) && \text{Since } P(\mathbf{X} = \mathbf{x}) \text{ is constant with respect to } Y
\end{aligned}$$

Using our training data we could interpret the joint distribution of $\mathbf{X}$ and Y as one giant multinomial with a different parameter for every combination of $\mathbf{X} = \mathbf{x}$ and $Y = y$. If for example, the input vectors are only length one. In other words $|\mathbf{x}| = 1$ and the number of values that x and y can take on are small, say binary, this is a totally reasonable approach. We could estimate the multinomial using MLE or MAP estimators and then calculate argmax over a few lookups in our table.

The bad times hit when the number of features becomes large. Recall that our multinomial needs to estimate a parameter for every unique combination of assignments to the vector $\mathbf{x}$ and the value y. If there are $|\mathbf{x}| = n$ binary features then this strategy is going to take order $\mathcal{O}(2^n)$ space and there will likely be many parameters that are estimated without any training data that matches the corresponding assignment.

Naive Bayes Assumption
The Naïve Bayes Assumption is that each feature of $\mathbf{x}$ is independent of one another given y.

The Naïve Bayes Assumption is wrong, but useful. This assumption allows us to make predictions using space and data which is linear with respect to the size of the features: $\mathcal{O}(n)$ if $|\mathbf{x}| = n$. That allows us to train and make predictions for huge feature spaces such as one which has an indicator for every word on the internet. Using this assumption the prediction algorithm can be simplified.

$$\begin{aligned}
\hat{y} &= \underset{y=\{0,1\}}{\arg\max}\ P(Y = y)P(\mathbf{X} = \mathbf{x}|Y = y) && \text{As we last left off} \\
&= \underset{y=\{0,1\}}{\arg\max}\ P(Y = y)\prod_i P(X_i = x_i|Y = y) && \text{Naïve bayes assumption} \\
&= \underset{y=\{0,1\}}{\arg\max}\ \log P(Y = y) + \sum_i \log P(X_i = x_i|Y = y) && \text{For numerical stability}
\end{aligned}$$

In the last step we leverage the fact that the argmax of a function is equal to the argmax of the log of a function. This algorithm is both fast and stable both when training and making predictions.