Chris Piech
CS109

Lecture #24
Nov 16th, 2018

Naïve Bayes

Naÿe Bayes is a type of machine learning algorithm called a classifier. It is used to predict the probability of a ”label” random variable Y based on the state of ”feature” random variables $\mathbf{X}$. We are going to learn all necessary parameters for the probabilistic relationship between $\mathbf{X}$ and Y from data. Specifically, Naÿe Bayes learns parameters from **training data** with feature/label pairs ($\mathbf{x}$, y) in order to estimate a function $\hat{y} = g(\mathbf{x})$. This function can then be used to make predictions.

1 Machine Learning: Classification

In classification tasks, your job is to use training data with feature/label pairs ($\mathbf{x}$, y) in order to estimate a label predicting function $\hat{y} = g(\mathbf{x})$. This function can then be used to make future predictions. In classification y takes on one of a **discrete** number of values. As such: $g(\mathbf{x}) = \underset{y}{\text{argmax}}\ \hat{P}(Y = y|\mathbf{X} = \mathbf{x})$.

To learn $\hat{P}(Y = y|\mathbf{X} = \mathbf{x})$ you are given N different ($\mathbf{x}$, y) pairs as **training data** : $(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})$ Where $\mathbf{x}^{(i)}$ is a vector of m discrete **features** for the ith training example and $y^{(i)}$ is the discrete label for the ith training example. A “feature” is a technical term for an assignment to an input random variable. This symbolic description of classification hides the fact that prediction is applied to interesting real life problems:

1. Predicting heart disease Y, based on a set of m observations from a heart scan $\mathbf{X}$.
2. Predicting ancestry Y, based on m DNA states $\mathbf{X}$.
3. Predicting if a user will like a movie Y given whether or not they like a set of m movies $\mathbf{X}$.

In this chapter we are going to assume that all random variables are binary. While this is not a necessary assumption (naive bayes can work for non-binary data), it makes it much easier to learn the core concepts. Specifically we assume that all labels are binary $y \in \{0, 1\}$ and all features are binary $x_j \in \{0, 1\}\ \forall j$.

2 Naïve Bayes Algorithm

Here is the Naïve Bayes algorithm. After presenting the algorithm I am going to show the theory behind it.

Prediction

For an set of features $\mathbf{x} = [x_1, x_2, \dots, x_m]$, we can make a corresponding prediction for y. We use hats (eg $\hat{P}$) to symbolize estimation.

$$\begin{aligned}
\hat{y} = g(\mathbf{x}) &= \underset{y\in\{0,1\}}{\text{argmax}}\ \hat{P}(Y = y|\mathbf{X} = \mathbf{x}) && \text{Our goal in classification} \\
&= \underset{y\in\{0,1\}}{\text{argmax}}\ \hat{P}(Y = y)\prod_{i=1}^{m}\hat{P}(X_i = x_i|Y = y) && \text{After applying Bayes and making assumptions} \\
&= \underset{y\in\{0,1\}}{\text{argmax}}\ \log\hat{P}(Y = y) + \sum_{i=1}^{m}\log\hat{P}(X_i = x_i|Y = y) && \text{Log version for numerical stability}
\end{aligned}$$

In order to calculate this expression we are going to need to learn the estimates $\hat{P}(Y = y)$ and $\hat{P}(X_i = x_i|Y = y)$ from data using a process called “training”.

Training

The objective in training is to estimate the probabilities $P(Y=k)$ and $P(X_i = x_i|Y=y)$ for all $1 \le i \le m$ features. 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:

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

Estimating $P(Y=y)$ is also straight forward. Using MLE estimation:

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

3 Theory

Now that you have the algorithm spelled out, lets go over the theory of how we got there. To do so, we will first explore an algorithm which doesn't work called "Brute Force Bayes". Then we introduce the Naive Bayes Assumption which will make our calculations possible.

Brute Force Bayes

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

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

Using our training data we could learn the conditional probability of $\mathbf{X}$ given Y as one giant conditional probability table 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 binary, this is a totally reasonable approach. We could estimate the parameters using MLE or MAP estimators and then calculate argmax with just a single lookup into our table.

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

Naïve Bayes Assumption

The Naïve Bayes Assumption is that each feature of $\mathbf{x}$ is independent of one another given y. That 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: $\mathcal{O}(m)$ if $|\mathbf{x}|=m$. 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 theory can be simplified.

$$\begin{aligned}
\hat{y} = g(\mathbf{x}) &= \underset{y \in \{0,1\}}{\operatorname{argmax}} \ \hat{P}(\mathbf{X} = \mathbf{x} | Y = y)\hat{P}(Y = y) && \text{As we last left off} \\
&= \underset{y}{\operatorname{argmax}} \ \Big(\prod_{i=1}^{n} \hat{P}(X_i = x_i | Y = y)\Big)\hat{P}(Y = y) && \text{Using the naïve bayes assumption} \\
&= \underset{y}{\operatorname{argmax}} \ \log \hat{P}(Y = y) + \sum_{i=1}^{m} \log \hat{P}(X_i = x_i | Y = y) && \text{Log version for numerical stability}
\end{aligned}$$

This algorithm is both fast and stable when training and making predictions. Let's think about a particular feature: i. How should we represent $P(X_i = x_i | Y = y)$? For a particular value of y that we are conditioning on, X_i can take on one of a few discrete values. Thus for each particular y we can model the likelihood of X_i taking on values as a multinomial. Then we can find MLE and MAP estimations for the parameters of those multinomial. Recall that MLE to estimate parameter p_i for a multinomial is just counting, and that MAP (with Laplace prior) to estimate parameter p_i imagines one extra example of each outcome:

$$\hat{p}_{iMLE} = \frac{n_i}{n} \qquad\qquad \hat{p}_{iMAP} = \frac{n_i + 1}{n + k}$$

Where: n is the number of observations. n_i is the number of observations with outcome i. k is the total number of outcome types.

Note that in the version of classification we are using in CS109, X_i is binary (technically binary is multionmial). Which means that $k = 2$. I used the multinomial derivation to help you understand how one would handle features (X_is) that take on more values.

Naïve Bayes is a simple form of a Bayesian Network where the label Y is the only variable which directly influences the likelihood of each feature variable X_i.

4 Example

Say we have a training dataset of thirty examples of people's preferences (like or not) for Star Wars, Harry Potter and Lord of the Rings. Each training example has x_1, x_2 and y where x_1 is whether or not the user liked Star Wars, x_2 is whether or not the user liked Harry Potter and y is whether or not the user liked Lord of the Rings. For the 30 training examples the MLE estimates are as follows:

X_1 / Y	0	1	MLE estimates	
0	3	10	0.23	0.77
1	4	13	0.24	0.76

X_2 / Y	0	1	MLE estimates	
0	5	8	0.38	0.62
1	7	10	0.41	0.59

Y	#	MLE est.
0	13	0.43
1	17	0.57

X_1 / Y	0	1	MAP estimates	
0	3	10	0.27	0.73
1	4	13	0.26	0.74

X_2 / Y	0	1	MAP estimates	
0	5	8	0.4	0.6
1	7	10	0.42	0.58

Y	#	MAP est.
0	13	0.45
1	17	0.55

For a new user who likes Star Wars ($X_1 = 1$) but not Harry Potter ($X_2 = 0$) do you predict that they will like Lord of the Rings? Yes! $Y = 1$ lead to a larger value in the argmax term:

$$\begin{aligned}
&\text{if } Y = 0: && \hat{P}(X_1 = 1 | Y = 0)\hat{P}(X_2 = 0 | Y = 0)\hat{P}(Y = 0) = (0.77)(0.38)(0.43) = 0.126 \\
&\text{if } Y = 1: && \hat{P}(X_1 = 1 | Y = 1)\hat{P}(X_2 = 0 | Y = 1)\hat{P}(Y = 1) = (0.76)(0.41)(0.57) = 0.178
\end{aligned}$$