Naive Bayes

Our training consists of the set $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ drawn from some unknown distribution. We want to make an estimate for new coming data points x for its label y and we do this by estimating $P(y|x)$.

One way to tackle this is to apply Bayes' rule with a simple trick, and an additional assumption. The trick part is to estimate $P(y)$ and $P(x|y)$ instead, since, by Bayes rule,

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

Estimating $P(y)$ is easy. For example, if Y takes on discrete binary values estimating $P(Y)$ reduces to coin tossing. We simply need to count how many times we observe each outcome (in this case each class):

$$P(y = c) = \frac{\sum_{i=1}^{n} I(y_i = c)}{n} = \hat{\pi}_c$$

Estimating $P(x|y)$, however, is not easy! The additional assumption that we make is the Naive Bayes assumption.

Naive Bayes Assumption

Assumption: feature values are independent given the label!

This is a very **bold** assumption. For example, a setting where the Naive Bayes classifier is often used is spam filtering. Here, the data is emails and the label is spam or not-spam. The Naive Bayes assumption implies that the words in an email are conditionally independent, given that you know that an email is spam or not. Clearly this is not true. Neither the words of spam or not-spam emails are drawn independently at random. However, the resulting classifiers can work well in practice even if this assumption is violated.

So, for now, let's pretend the Naive Bayes assumption holds. Then the Bayes

Classifier can be defined as

$$\begin{aligned}
h(\mathbf{x}) &= \operatorname*{argmax}_y P(y|\mathbf{x}) \\
&= \operatorname*{argmax}_y \ \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\
&= \operatorname*{argmax}_y \ P(\mathbf{x}|y)P(y) && (P(\mathbf{x}) \text{ does not depend on } y) \\
&= \operatorname*{argmax}_y \ \prod_{\alpha=1}^{d} P(x_\alpha|y)P(y) && (\text{by the naive Bayes assumption}) \\
&= \operatorname*{argmax}_y \ \sum_{\alpha=1}^{d} \log(P(x_\alpha|y)) + \log(P(y)) && (\text{as log is a monotonic function})
\end{aligned}$$

Estimating $\log(P(x_\alpha|y))$ is easy as we only need to consider one dimension. And estimating $P(y)$ is not affected by the assumption.

Estimating $P(x_\alpha|y)$

Now that we know how we can use our assumption to make the estimation of $P(y|\mathbf{x})$ tractable. There are 3 notable cases in which we can use our naive Bayes classifier.

Categorical features

Features: $[x]_\alpha \in \{f_1, f_2, ..., f_{K_\alpha}\}$

Each feature α falls into one of K_α categories. (Note that the case with binary features is just a specific case of this, where $K_\alpha = 2$.) An example of such a setting may be medical data where one feature could be gender (male / female) or marital status (single / married / widowed).

Model: $P(x_\alpha|y)$

$$P(x_\alpha = j|y = c) = [\theta_{jc}]_\alpha$$
$$\text{and } \sum_{j=1}^{K_\alpha} [\theta_{jc}]_\alpha = 1$$

where $[\theta_{jc}]_\alpha$ is the probability of feature α having the value j, given that the label is c. And the constraint indicates that x_α must have one of the categories $\{1, ..., K_\alpha\}$.

For d dimensional data, there exist d independent dice for each class. Per class, each feature has one die. We assume for the training samples, they were generated by rolling one die after another. The value in dimension i corresponds to the outcome of the i^{th} die.

<u>**Parameter estimation**</u>:

$$[\hat{\theta}_{jc}]_\alpha = \frac{\sum_{i=1}^{n} I(y_i = c) I(x_{i\alpha} = j) + l}{\sum_{i=1}^{n} I(y_i = c) + lK_\alpha},$$

where $x_{i\alpha} = [x_i]_\alpha$ and l is a smoothing parameter. By setting $l = 0$ we get an MLE estimator, $l > 0$ leads to MAP. If we set $l = +1$ we get Laplace smoothing.

In words (without the l hallucinated samples) this means

$$\frac{\text{num of samples with label c that have feature } \alpha \text{ with value } j}{\text{num of samples with label } c}.$$

Essentially the categorical feature model associate a special coin with each feature and label. The generative model that we are assuming is that the data was generated by first choosing the label (e.g. "healthy person"). That label comes with a set of d "dice", for each dimension one. The generator picks each die, tosses it and fills in the feature value with the outcome of the coin toss. So if there are C possible labels and d dimensions we are estimating $d \times C$ "dice" from the data. However, per data point only d dice are tossed (one for each dimension). Die α (for any label) has K_α possible "sides". Of course this is not how the data is generated in reality - but it is a modeling assumption that we make. We then learn these models from the data and during test time see which model is more likely given the sample.

<u>**Prediction**</u>:

$\text{argmax}_y \; P(y = c \mid \mathbf{x}) \propto \text{argmax}_y \; \hat{\pi}_c \prod_{\alpha=1}^{d} [\hat{\theta}_{jc}]_\alpha$

Multinomial features

If feature values don't represent categories (e.g. male/female) but counts we need to use a different model. E.g. in the text document categorization, feature value $x_\alpha = j$ means that in this particular document x the αth word in my dictionary appears j times. Let us consider the example of spam filtering. Imagine the αth word is indicative towards "spam". Then if $x_\alpha = 10$ means that this email is likely spam (as word appears 10 times in it). And another email with $x'_\alpha = 20$ should be even more likely to be spam (as the spammy word appears twice as often). With categorical features this is not guaranteed. It could be that the training set does not contain any email that contain word α exactly 20 times. In this case you would simply get the hallucinated smoothing values for both spam and not-spam - and the signal is lost. We need a model that incorporates our knowledge that features are counts - this will help us during estimation (you don't have to see a training email with exactly the same number of word occurances) and during inference/testing (as you will obtain these monotonicities that one might expect). The multinomial distribution does exactly that.

There are only as many dice as classes. Each die has d sides. The value of the ith feature shows how many times this particular side was rolled.

Features:

$$x_\alpha \in \{0, 1, 2, \dots, m\} \text{ and } m = \sum_{\alpha=1}^{d} x_\alpha$$

Each feature α represents a count and m is the length of the sequence. An example of this could be the count of a specific word α in a document of length m and d is the size of the vocabulary.

Model $P(x|y)$:

Use the multinomial distribution

$$P(\mathbf{x} \mid m, y = c) = \frac{m!}{x_1! \cdot x_2! \cdot \dots \cdot x_d!} \prod_{\alpha=1}^{d} (\theta_{\alpha c})^{x_\alpha}$$

where $\theta_{\alpha c}$ is the probability of selecting x_α and $\sum_{\alpha=1}^{d} \theta_{\alpha c} = 1$. So, we can use this to generate a spam email, i.e., a document x of class y=spam by picking m words independently at random from the vocabulary of d words using $P(x|y = spam)$.

Parameter Estimate:

$$\hat{\theta}_{\alpha c} = \frac{\sum_{i=1}^{n} I(y_i = c) x_{i\alpha} + l}{\sum_{i=1}^{n} I(y_i = c) m_i + l \cdot d}$$

where $m_i = \sum_{\beta=1}^{d} x_{i\beta}$ denotes the number of words in document i. The numerator sums up all counts for feature x_α and the denominator sums up all counts of all features across all data points. E.g.,

$$\frac{\text{num of times word } \alpha \text{ appears in all spam emails}}{\text{num of words in all spam emails combined}}.$$

Again, l is the smoothing parameter.

Prediction:

$$\underset{c}{\operatorname{argmax}}\ P(y=c \mid \mathbf{x}) \propto \underset{c}{\operatorname{argmax}}\ \hat{\pi}_c \prod_{\alpha=1}^{d} \hat{\theta}_{\alpha c}^{x_\alpha}$$

Continuous features (Gaussian Naive Bayes)

Features:

$$x_\alpha \in \mathbb{R} \qquad\qquad \text{(each feature takes on a real value)}$$

Model $P(x_\alpha|y)$: Use Gaussian distribution

$$P(x_\alpha \mid y=c) = \mathcal{N}\left(\mu_{\alpha c}, \sigma_{\alpha c}^2\right) = \frac{1}{\sqrt{2\pi}\sigma_{\alpha c}} e^{-\frac{1}{2}\left(\frac{x_\alpha - \mu_{\alpha c}}{\sigma_{\alpha c}}\right)^2}$$

Note that the model specified above is based on our assumption about the data - that each feature α comes from a class-conditional Gaussian distribution. The full distribution $P(x|y) \sim N(\mu_y, \Sigma_y)$, where Σ_y is a diagonal covariance matrix with $[\Sigma_y]_{\alpha,\alpha} = \sigma_{\alpha,y}^2$.

Parameter Estimation As always, we estimate the parameters of the distributions for each dimension and class independently. Gaussian distributions only have two parameters, the mean and variance. The mean μ_α,y is estimated by the average feature value of dimension α from all samples with label y. The (squared) standard deviation is simply the variance of this estimate.

$$\mu_{\alpha c} \leftarrow \frac{1}{n_c} \sum_{i=1}^{n} I(y_i = c) x_{i\alpha} \qquad\qquad \text{where } n_c = \sum_{i=1}^{n} I(y_i = c)$$

$$\sigma_{\alpha c}^2 \leftarrow \frac{1}{n_c} \sum_{i=1}^{n} I(y_i = c)(x_{i\alpha} - \mu_{\alpha c})^2$$

Naive Bayes is a linear classifier

1. Suppose that $y_i \in \{-1, +1\}$ and features are multinomial We can show that

$$h(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\ P(y) \prod_{\alpha-1}^{d} P(x_\alpha \mid y) = \operatorname{sign}(\mathbf{w}^\top \mathbf{x} + b)$$

that is, $\mathbf{w}^\top \mathbf{x} + b > 0 \iff h(\mathbf{x}) = +1$.

If we define $P(x_\alpha|y=+1) \propto \theta_{\alpha+}^{x_\alpha}$ and $P(Y=+1) = \pi_+$, then

$$[\mathbf{w}]_\alpha = \log(\theta_{\alpha+}) - \log(\theta_{\alpha-})$$
$$b = \log(\pi_+) - \log(\pi_-)$$

If we use the above to do classification, we can compute for $\mathbf{w}^\top \cdot \mathbf{x} + b$. Simplifying this further leads to

$$
\begin{aligned}
&\mathbf{w}^\top \mathbf{x} + b > 0 \\
&\iff \sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha \overbrace{(\log(\theta_{\alpha+}) - \log(\theta_{\alpha-}))}^{[\mathbf{w}]_\alpha} + \overbrace{\log(\pi_+) - \log(\pi_-)}^{b} > 0 && \textcircled{1} \\
&\iff \exp\left(\sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha (\log(\theta_{\alpha+}) - \log(\theta_{\alpha-})) + \log(\pi_+) - \log(\pi_-)\right) > 1 && \textcircled{2} \\
&\iff \prod_{\alpha=1}^{d} \frac{\exp\left(\log \theta_{\alpha+}^{[\mathbf{x}]_\alpha} + \log(\pi_+)\right)}{\exp\left(\log \theta_{\alpha-}^{[\mathbf{x}]_\alpha} + \log(\pi_-)\right)} > 1 && \textcircled{3} \\
&\iff \prod_{\alpha=1}^{d} \frac{\theta_{\alpha+}^{[\mathbf{x}]_\alpha} \pi_+}{\theta_{\alpha-}^{[\mathbf{x}]_\alpha} \pi_-} > 1 && \textcircled{4} \\
&\iff \frac{\prod_{\alpha=1}^{d} P([\mathbf{x}]_\alpha | Y = +1)\pi_+}{\prod_{\alpha=1}^{d} P([\mathbf{x}]_\alpha | Y = -1)\pi_-} > 1 && \textcircled{5} \\
&\iff \frac{P(\mathbf{x}|Y = +1)\pi_+}{P(\mathbf{x}|Y = -1)\pi_-} > 1 && \textcircled{6} \\
&\iff \frac{P(Y = +1|\mathbf{x})}{P(Y = -1|\mathbf{x})} > 1 && \textcircled{7} \\
&\iff P(Y = +1|\mathbf{x}) > P(Y = -1|\mathbf{x}) \\
&\iff \underset{y}{\operatorname{argmax}} P(Y = y|\mathbf{x}) = +1 && \textcircled{8}
\end{aligned}
$$

- ① (Plugging in definition of $\mathbf{w}, b$.)
- ② (exponentiating both sides)
- ③ Because $a \log(b) = \log(b^a)$ and $\exp{(a - b)} = \frac{e^a}{e^b}$ operations
- ④ Because $\exp(\log(a)) = a$ and $e^{a+b} = e^a e^b$
- ⑤ Because $P([\mathbf{x}]_\alpha | Y = -1) = \theta_{\alpha-}^{\mathbf{x}]_\alpha}$
- ⑥ By the naive Bayes assumption.
- ⑦ By Bayes rule (the denominator $P(\mathbf{x})$ cancels out, and $\pi_+ = P(Y = +1)$.)
- ⑧ i.e. the point $\mathbf{x}$ lies on the positive side of the hyperplane iff Naive Bayes predicts +1

2. In the case of continuous features (Gaussian Naive Bayes), we can show that

$$P(y \mid \mathbf{x}) = \frac{1}{1 + e^{-y(\mathbf{w}^\top \mathbf{x} + b)}}$$

This model is also known as <u>**logistic regression**</u>. NB and LR produce asymptotically the same model if the Naive Bayes assumption holds.