Naive Bayes

Cornell CS 4/5780 — Spring 2022

Our training set consists of the set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ drawn from some unknown distribution $\mathcal{P}(X, Y)$. We assume that all pairs are sampled independently from this distribution. From this assumption, we obtain

$$\mathbf{P}_{(x_i,y_i)\sim\mathcal{P}}(\mathcal{D}) = \mathbf{P}_{(x_i,y_i)\sim\mathcal{P}}((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)) = \prod_{i=1}^{n} \mathcal{P}(x_i, y_i).$$

If we had enough data, we could estimate $\mathcal{P}(X, Y)$ similar to the coin example in the previous lecture, where we imagine a **gigantic** die that has one side for each possible value of $(\mathbf{x}, y)$. We can estimate the probability that one specific side comes up through counting:

$$\hat{\mathbf{P}}(\mathbf{x}, y) = \frac{\sum_{i=1}^{n} I(\mathbf{x}_i = \mathbf{x} \wedge y_i = y)}{n},$$

where $I(\mathbf{x}_i = \mathbf{x} \wedge y_i = y) = 1$ if $\mathbf{x}_i = \mathbf{x}$ and $y_i = y$ and 0 otherwise. This use of I generalizes our "number of heads" and "number of tails" reasoning from our coin example last lecture.

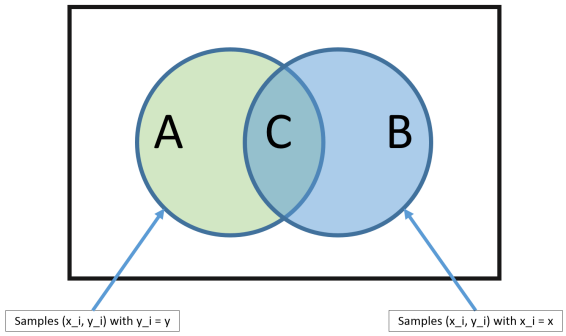
Suppose that we are using d-dimensional binary features $\mathbf{x} \in \{0, 1\}^d$ for a two-class classification problem. How many sides would this "die" have? Roughly how large would our dataset size n need to be for us to get a "good" estimate of this distribution?

Of course, if we are primarily interested in predicting the label y from the features $\mathbf{x}$, we may estimate $\mathbf{P}(Y|X)$ directly instead of $\mathbf{P}(X, Y)$. We can then use the Bayes Optimal Classifier for a specific $\hat{P}(y|\mathbf{x})$ to make predictions.

So how can we estimate $\hat{P}(y|\mathbf{x})$? Previously we have derived that $\hat{\mathbf{P}}(y) = \frac{\sum_{i=1}^{n} I(y_i=y)}{n}$. Similarly, $\hat{\mathbf{P}}(\mathbf{x}) = \frac{\sum_{i=1}^{n} I(\mathbf{x}_i=\mathbf{x})}{n}$ and $\hat{\mathbf{P}}(y, \mathbf{x}) = \frac{\sum_{i=1}^{n} I(\mathbf{x}_i=\mathbf{x} \wedge y_i=y)}{n}$. We can put these two together

$$\hat{P}(y|\mathbf{x}) = \frac{\hat{P}(y, \mathbf{x})}{\mathbf{P}(\mathbf{x})} = \frac{\sum_{i=1}^{n} I(\mathbf{x}_i = \mathbf{x} \wedge y_i = y)}{\sum_{i=1}^{n} I(\mathbf{x}_i = \mathbf{x})}$$

The Venn diagram illustrates that the MLE method estimates $\hat{P}(y|\mathbf{x})$ as $\hat{P}(y|\mathbf{x}) = \frac{|C|}{|B|}$.



Problem: But there is a big problem with this method. The MLE estimate is only good if there are many training vectors with the **same identical** features as $\mathbf{x}$! In **high dimensional spaces** (or with continuous $\mathbf{x}$), this never happens! So $|B| \to 0$ and $|C| \to 0$.

Naive Bayes

We can approach this dilemma with a simple trick, and an additional assumption. The trick part is to estimate $\mathbf{P}(y)$ and $\mathbf{P}(\mathbf{x}|y)$ instead, since, by Bayes rule,

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

Recall from Estimating Probabilities from Data that estimating $\mathbf{P}(y)$ and $\mathbf{P}(\mathbf{x}|y)$ is called *generative learning*. Estimating $\mathbf{P}(y)$ is easy. For example, if Y takes on discrete binary values estimating $\mathbf{P}(y)$ reduces to coin tossing. We simply need to count how many times we observe each outcome (in this case each class): $\mathbf{P}(y = c) = \frac{\sum_{i=1}^{n} I(y_i=c)}{n} = \hat{\pi}_c$.

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

Naive Bayes Assumption:

$$\mathbf{P}(\mathbf{x}|y) = \prod_{\alpha=1}^{d} \mathbf{P}(x_\alpha|y), \text{ where } x_\alpha = [\mathbf{x}]_\alpha \text{ is the value for feature } \alpha$$

i.e., feature values are **independent given the label!** This is a very **bold** assumption.

What tasks/setups can you think of where the Naive Bayes assumption could be reasonable?

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.

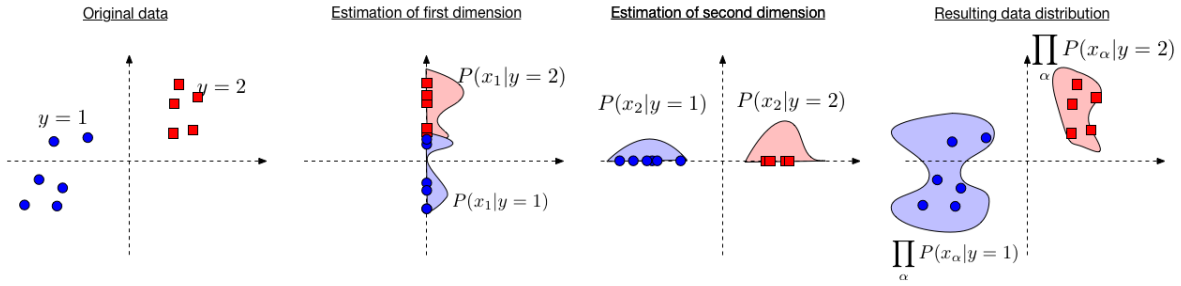


Illustration behind the Naive Bayes algorithm. We estimate $\mathbf{P}(x_\alpha|y)$ *independently in each dimension (middle two images) and then obtain an estimate of the full data distribution by assuming conditional independence* $\mathbf{P}(\mathbf{x}|y) = \prod_\alpha \mathbf{P}(x_\alpha|y)$ *(very right image).*

So, for now, let's pretend the Naive Bayes assumption holds. Then the Bayes Classifier can be defined as

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

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

Activity. Suppose that we have the following spam-classification task, where the goal is to classify an email as spam $y = 1$ *or not spam* $y = 0$*.*

1. *Given an email, we let* $x_1 = 1$ *if the email contains the word "shipping" and* 0 *otherwise.*
2. *We let* $x_2 = 1$ *if the email contains the word "perceptron" and* 0 *otherwise.*
3. *We estimate that the probability that an email contains the word "shipping" given that it is spam is* 0.8*.*
4. *We estimate that the probability that an email contains the word "shipping" given that it is not-spam is* 0.4*.*
5. *We estimate that the probability that an email contains the word "perceptron" given that it is spam is* 0.01*.*
6. *We estimate that the probability that an email contains the word "perceptron" given that it is not-spam is* 0.1*.*
7. *The prior probability that an email is spam is* 0.75*.*

How would the Naive Bayes classifier for this task classify an email that contains both the words "shipping" and "perceptron"?

How would the Naive Bayes classifier for this task classify an email that contains the word "perceptron" but not "shipping"?

Estimating $\mathbf{P}([\mathbf{x}]_\alpha|y)$

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

Case #1: Categorical features

Features:

$$[\mathbf{x}]_\alpha \in \{f_1, f_2, \cdots, f_{K_\alpha}\}$$

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 *marital status* (single / married). Model $\mathbf{P}(x_\alpha \mid y)$:

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

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 $\{1, \ldots, K_\alpha\}$. Parameter estimation:

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

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

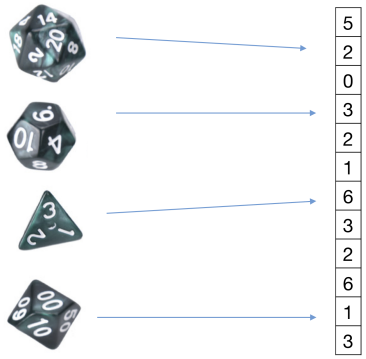


Illustration of categorical NB. For d dimensional data, there exist d independent dice for each class. Each feature has one die per class. We assume training samples were generated by rolling one die after another. The value in dimension i corresponds to the outcome that was rolled with the i^{th} die.

In words (without the l hallucinated samples) this means

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

ssentially the categorical feature model associates 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.

Prediction:

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

Case #2: Multinomial features

If feature values don't represent categories (e.g. single/married) 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 $\mathbf{x}$ the α^{th} word in my dictionary appears j times. Let us consider the example of spam filtering. Imagine the α^{th} word is indicative of being "spam". Then if $x_\alpha = 10$ means that this email is likely spam (as word α 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 occurrences) and during inference/testing (as you will obtain these monotonicities that one might expect). The multinomial distribution does exactly that.

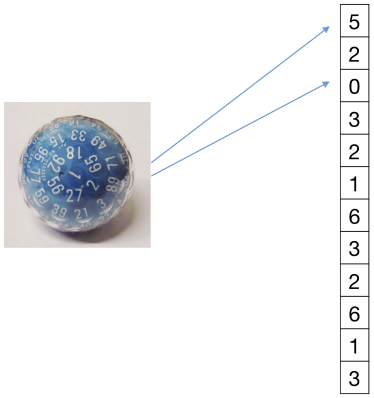


Illustration of multinomial NB. There are only as many dice as classes. Each die has d sides. The value of the i^{th} feature shows how many times this particular side was rolled.

Features:

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

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 $\mathbf{P}(\mathbf{x} \mid y)$: Use the multinomial distribution

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

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 $\mathbf{x}$ of class $y = \text{spam}$ by picking m words independently at random from the vocabulary of d words using $\mathbf{P}(\mathbf{x} \mid y = \text{spam})$. Parameter estimation:

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

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.,

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

Again, l is the smoothing parameter. Prediction:

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

Case #3: Continuous features (Gaussian Naive Bayes)

Features:

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

Model $\mathbf{P}(x_\alpha \mid y)$: Use Gaussian distribution

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

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 $\mathbf{P}(\mathbf{x}|y) \sim \mathcal{N}(\mu_y, \Sigma_y)$, where Σ_y is a diagonal covariance matrix with $[\Sigma_y]_{\alpha,\alpha} = \sigma_{\alpha,y}^2$.

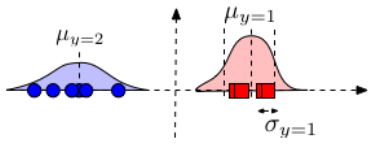


Illustration of Gaussian NB. Each class conditional feature distribution $\mathbf{P}(x_\alpha|y)$ is assumed to originate from an independent Gaussian distribution with its own mean $\mu_{\alpha,y}$ and variance $\sigma_{\alpha,y}^2$.

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 $\mu_{\alpha,y}$ 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.

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

Naive Bayes is a linear classifier

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

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

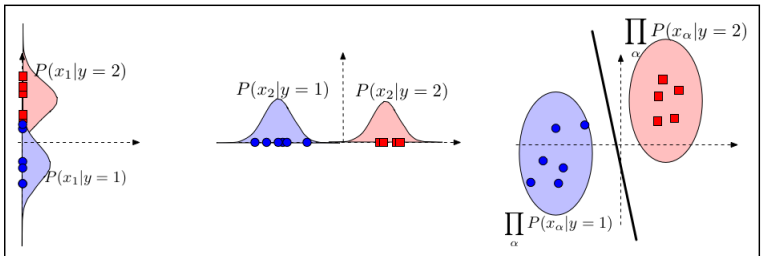
That is,

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

As before, we define $\mathbf{P}(x_\alpha|y=+1) \propto \theta_{\alpha+}^{x_\alpha}$ and $\mathbf{P}(y=+1) = \pi_+$:

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

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



Naive Bayes leads to a linear decision boundary in many common cases. Illustrated here is the case where $\mathbf{P}(x_\alpha|y)$ is Gaussian and where $\sigma_{\alpha,c}$ is identical for all c (but can differ across dimensions α). The boundary of the ellipsoids indicate regions of equal probabilities $\mathbf{P}(\mathbf{x}|y)$. The red decision line indicates the decision boundary where $\mathbf{P}(y=1|\mathbf{x}) = \mathbf{P}(y=2|\mathbf{x})$.

Simplifying this further leads to

$$\begin{aligned}
\mathbf{w}^\top \mathbf{x} + b > 0 &\iff \sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha \overbrace{(\log(\theta_{\alpha+}) - \log(\theta_{\alpha-}))}^{[\mathbf{w}]_\alpha} + \overbrace{\log(\pi_+) - \log(\pi_-)}^{b} > 0 && \text{(Plugging in definition of } \mathbf{w}, b.\text{)} \\
&\iff \exp\left(\sum_{\alpha=1}^{d} [\mathbf{x}]_\alpha (\log(\theta_{\alpha+}) - \log(\theta_{\alpha-})) + \log(\pi_+) - \log(\pi_-)\right) > 1 && \text{(exponentiating both sides)} \\
&\iff \prod_{\alpha=1}^{d} \frac{\exp\left(\log \theta_{\alpha+}^{[\mathbf{x}]_\alpha} + \log(\pi_+)\right)}{\exp\left(\log \theta_{\alpha-}^{[\mathbf{x}]_\alpha} + \log(\pi_-)\right)} > 1 && \text{Because } a\log(b) = \log(b^a) \text{ and } \exp(a-b) = \frac{e^a}{e^b} \text{ operations} \\
&\iff \prod_{\alpha=1}^{d} \frac{\theta_{\alpha+}^{[\mathbf{x}]_\alpha} \pi_+}{\theta_{\alpha-}^{[\mathbf{x}]_\alpha} \pi_-} > 1 && \text{Because } \exp(\log(a)) = a \text{ and } e^{a+b} = e^a e^b \\
&\iff \frac{\prod_{\alpha=1}^{d} \mathbf{P}([\mathbf{x}]_\alpha|Y=+1)\pi_+}{\prod_{\alpha=1}^{d} \mathbf{P}([\mathbf{x}]_\alpha|Y=-1)\pi_-} > 1 && \text{Because } \mathbf{P}([\mathbf{x}]_\alpha|Y=-1) = \theta_{\alpha-}^{[\mathbf{x}]_\alpha} \\
&\iff \frac{\mathbf{P}(\mathbf{x}|Y=+1)\pi_+}{\mathbf{P}(\mathbf{x}|Y=-1)\pi_-} > 1 && \text{By the naive Bayes assumption.} \\
&\iff \frac{\mathbf{P}(Y=+1|\mathbf{x})}{\mathbf{P}(Y=-1|\mathbf{x})} > 1 && \text{By Bayes rule (the denominator } \mathbf{P}(\mathbf{x}) \text{ cancels out, and } \pi_+ = \mathbf{P}(Y=+1).\text{)} \\
&\iff \mathbf{P}(Y=+1|\mathbf{x}) > \mathbf{P}(Y=-1|\mathbf{x}) \\
&\iff \underset{y}{\text{argmax}}\ \mathbf{P}(Y=y|\mathbf{x}) = +1 && \text{i.e. the point } \mathbf{x} \text{ lies on the positive side of the hyperplane iff Naive Bayes predicts } +1
\end{aligned}$$

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

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

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