Our training consists of the set

$$D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

drawn from some unknown distribution $\mathbb{P}(X, Y)$.

Since all pairs are sampled iid, we obtain

$$\mathbb{P}(D) = \mathbb{P}((x_1, y_1), \ldots, (x_n, y_n)) = \prod_{\alpha=1}^{n} \mathbb{P}(x_\alpha, y_\alpha)$$

If we have enough data, we could estimate $\mathbb{P}(X, Y)$ similar to the coin example in lecture 4, where we imagine a **gigantic** die that has one side for each possible value of $(\vec{x}, y)$.

We can estimate the probability that one specific side comes up through counting:

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

where

$$I(x_i = x \wedge y_i = y) = \begin{cases} 1 & \text{if } x_i = x \text{ \& } y_i = y \\ 0 & \text{o/w} \end{cases}$$

Of course we are primarily interested in predicting the label $y$ from the features $\vec{x}$, we may estimate $\mathbb{P}(Y|X)$ directly instead of $\mathbb{P}(X, Y)$.

We can then use the Bayes Optimal Classifier for a specific $\hat{P}(y|\vec{x})$ to make predictions.

How do we estimate $\hat{P}(y|\vec{x})$?

Well, know

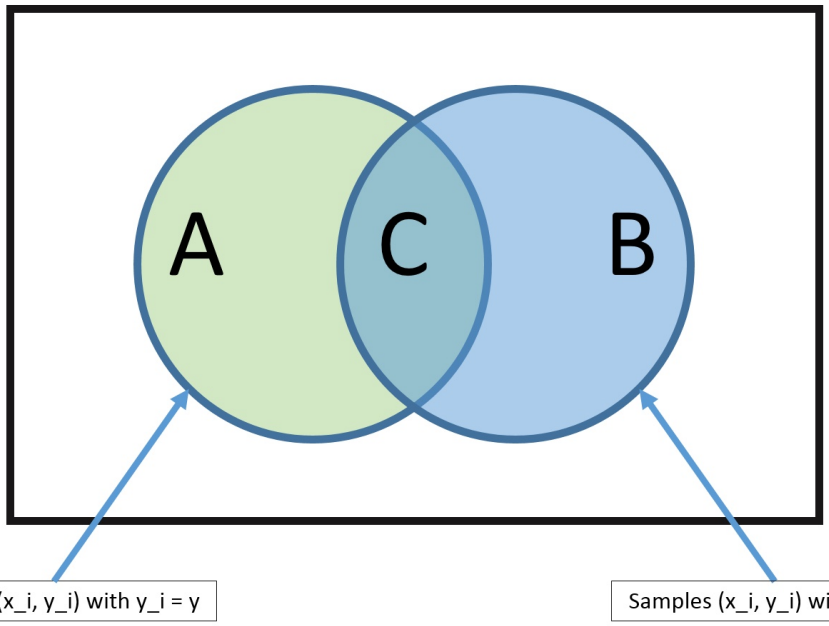
$$\hat{P}(\vec{x}) = \frac{\sum_{i=1}^{n} I(x_i = x)}{n} \quad ; \quad \hat{P}(y) = \frac{\sum_{i=1}^{n} I(y_i = y)}{n}$$

and

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

Thus,

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




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

# PROBLEM!

The MLE estimate is only good if there are many training vectors w/ the **same identical** features as $\vec{x}$! In high dimensional spaces (or w/ continuous $\vec{x}$), this **NEVER** happens! So $|B| \to 0$ and $|C| \to 0$.

## Naive Bayes

We can approach this dilemna w/ a simple trick, and an additional assumption.

The trick: estimate $P(y)$ and $P(\vec{x}|y)$ instead of $P(\vec{x}, y)$

generative learning

Thus,

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

Estimating $P(y)$ is easy.

For example, if $Y$ takes on discrete binary values estimating $P(Y)$ reduces to coin tossing. Just need to count how many times we observe each outcome

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

Estimating $P(\vec{x}|y)$ however is **NOT** easy!

Need to make an additional assumption, the Naive Bayes assumption

## Naive Bayes Assumption:

$$\mathbb{P}(\vec{x}|y) = \prod_{\alpha=1}^{d} \mathbb{P}(x_\alpha|y)$$

where $x_\alpha = [\vec{x}]_\alpha$ is the value for feature $\alpha$

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

For example, a setting where Naive Bayes 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 whether an email is spam or not

Clearly, this is **NOT** true Sameer!

Neither the words of spam or not spam email are drawn independently at random.

However, the resulting classifiers can work well in practice even if this assumption is violated.

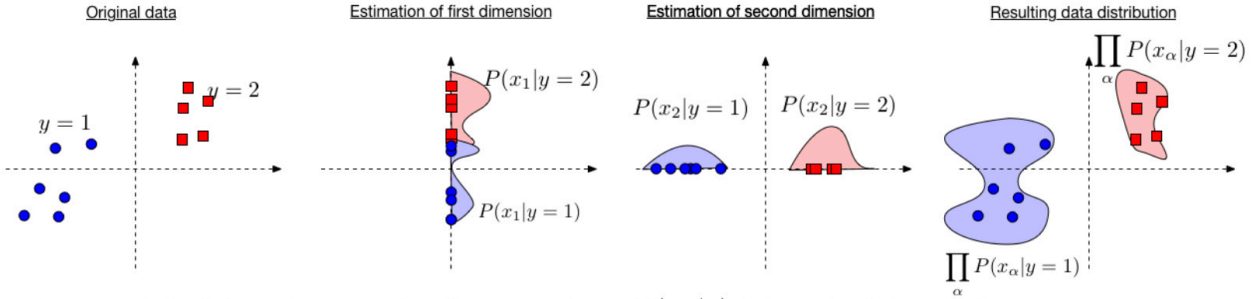



*Illustration behind the Naive Bayes algorithm. We estimate $P(x_\alpha|y)$ independently in each dimension (middle two images) and then obtain an estimate of the full data distribution by assuming conditional independence $P(\mathbf{x}|y) = \prod_\alpha P(x_\alpha|y)$ (very right image).*

So, for now assume the Naive Bayes assumption holds. Then the Bayes classifier can be defined as

$$
\begin{aligned}
h(\vec{x}) &= \underset{y}{\operatorname{argmax}}\ \mathbb{P}(y \mid \vec{x}) \\
&= \underset{y}{\operatorname{argmax}}\ \frac{\mathbb{P}(\vec{x} \mid y)\,\mathbb{P}(y)}{\mathbb{P}(\vec{x})} \\
&= \underset{y}{\operatorname{argmax}}\ \mathbb{P}(\vec{x} \mid y)\,\mathbb{P}(y) && \text{P(x) has no y dependence} \\
&= \underset{y}{\operatorname{argmax}}\ \prod_{\alpha=1}^{d} \mathbb{P}(x_\alpha \mid y)\,\mathbb{P}(y) && \text{by Naive Bayes Assumption} \\
&= \underset{y}{\operatorname{argmax}}\ \sum_{\alpha=1}^{d} \log(\mathbb{P}(x_\alpha \mid y)) + \log(\mathbb{P}(y)) && \text{log is monotonic}
\end{aligned}
$$

Estimating $\mathbb{P}(x_\alpha \mid y)$ is easy as we only need to consider one dimension. And estimating $\mathbb{P}(y)$ is **NOT** impacted by assumption.

## Estimating $\mathbb{P}([\vec{x}]_\alpha \mid y)$

Now that we know how we can use our assumption to make the definition of $\mathbb{P}(y \mid \vec{x})$ tractable.

There are **3** notable cases we can use our Naive Bayes classifier.

# Case 1: Categorical Features

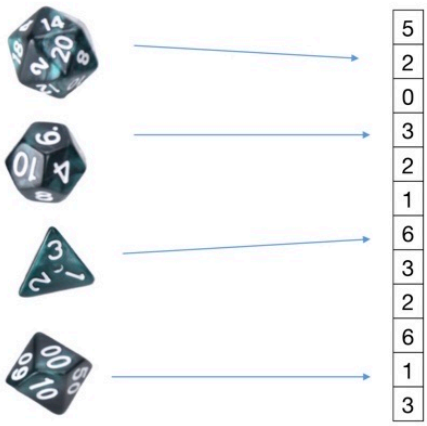



*Illustration of categorical NB. For $d$ dimensional data, there exist $d$ independent dice for each class. Each feature has one die per class. We assume training samples were generated by rolling one die after another. The value in dimension $i$ corresponds to the outcome that was rolled with the $i^{th}$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

## Features:

$$[\vec{x}]_\alpha \in \{f_1, f_2, \ldots, f_{K_\alpha}\}$$

Each feature $\alpha$ falls into one of $K_\alpha$ categories. (Binary, $K_\alpha = 2$). An example of such a setting could be medical data where one feature could be gender or marital status.

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

$$P(x_\alpha = j \mid y = c) = [\theta_{jc}]_\alpha$$

and

$$\sum_{j=1}^{K_\alpha} [\theta_{jc}]_\alpha = 1$$

$P(x_\alpha = j \mid y = c)$

where $[\theta_{jc}]_\alpha$ is the probability of feature $\alpha$ having value $j$, given that the label is $c$. And the constraint indicates that $x_\alpha$ 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 $x_{i\alpha} = [\vec{x}_i]_\alpha$ and $l$ = smoothing parameter.

$l = 0 \rightarrow$ MLE estimator

$l > 0 \Rightarrow$ MAP

$l = +1 \rightarrow$ Laplace smoothing

In words (w/o hallucinated 1 samples) this means

$$\frac{\#\text{ of sample w/ label } c \text{ that have feature } \alpha \text{ w/ value } j}{\#\text{ of samples w/ label } c}$$

Stay w/ me Sameer, this next part gets a little confusing

- Essentially the categorical feature model associates a special coin w/ each feature and label.
- The generative model we are assuming is that the data was generated by first choosing the label.
  - that label comes w/ a set of $d$ dice; one for each dimension
- The generator picks each die, tosses it, and fills in the feature value w/ the outcome of the coin toss
- So for $C$ possible labels and $d$ dimensions we are estimating $d \times C$ "dice" from the data.
- **HOWEVER**, per data pt only $d$ dice are tossed (1 per dimension)
- Die $\alpha$ (for any label) has $K_\alpha$ possible "sides"
- Of course, this is **NOT** how data is generated in reality - but it is a modeling assumption we make
- We then learn these models from the data and during test time see which model is more likely given the sample

Prediction:

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

# Case #2: Multinomial Features

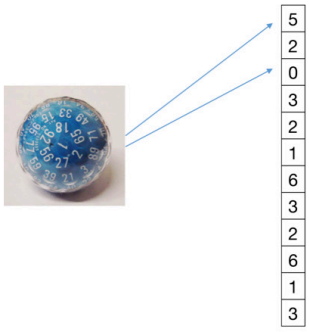



*Illustration of multinomial NB. There are only as many dice as classes. Each die has $d$ sides. The value of the $i^{th}$ feature shows how many times this particular side was rolled.*

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 $\mathbf{x}$ the $\alpha^{th}$ word in my dictionary appears $j$ times. Let us consider the example of spam filtering. Imagine the $\alpha^{th}$ word is indicative towards ``spam''. Then if $x_\alpha = 10$ means that this email is likely spam (as word $\alpha$ 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 $\alpha$ 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.

## Features:

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

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

## Model $P(\vec{x} \mid y)$: Use multinomial distribution

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

ways to reshuffle

probability of getting $x_\alpha | y = c$

amount of times observed

where $\theta_{\alpha c}$ is the probability of selecting $x_\alpha$ and $\sum_{\alpha=1}^{d} \theta_{\alpha c} = 1$

So, we can use this to generate a spam email!

i.e a document $\vec{x}$ of class $y$ = spam by picking $m$ words independently at random from the vocabulary of $d$ words using $P(\vec{x} \mid y = \text{spam})$

## Parameter Estimation:

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

all counts for features $x_\alpha$

sums up all counts of all features across ALL data points

where

$$m_i = \sum_{\beta=1}^{d} x_{i\beta}$$

denotes the number of words in document $i$.

Once again, $l$ is smoothing parameter.

## Prediction:

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

$$\mathbb{P}(Y = 1 \mid X = x_{test}) = \frac{\mathbb{P}(X = x_{test} \mid Y = 1)\,\mathbb{P}(Y = 1)}{\mathbb{P}(X = x_{test})}$$

$$\mathbb{P}(Y = -1 \mid X = x_{test}) = \frac{\mathbb{P}(X = x_{test} \mid Y = -1)\,\mathbb{P}(Y = -1)}{\mathbb{P}(X = x_{test})}$$

$$= \frac{\mathbb{P}(X = x_{test} \mid Y = 1)\,\mathbb{P}(Y = 1)}{\mathbb{P}(X = x_{test} \mid Y = -1)\,\mathbb{P}(Y = -1)}$$

# Case 3: Continuous Features (Gaussian Naive Bayes)

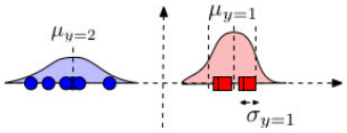



*Illustration of Gaussian NB. Each class conditional feature distribution $P(x_\alpha|y)$ is assumed to originate from an independent Gaussian distribution with its own mean $\mu_{\alpha,y}$ and variance $\sigma^2_{\alpha,y}$.*

## Features:

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

## Model:

$$P(x_\alpha | y = c) = \mathcal{N}(\mu_{ac}, \sigma^2_{ac}) = \frac{1}{\sqrt{2\pi}\,\sigma_{ac}} e^{-\frac{(x-\mu_{ac})^2}{2\sigma^2_{ac}}}$$

Note that the model is based on our assumption about the data - that each feature $\alpha$ comes from a class-conditional Gaussian distribution.

The full distribution

$$P(\vec{x}|y) \sim \mathcal{N}(\mu_y, \Sigma_y)$$ ← diagonal covariance matrix w/ $|\Sigma_y|_{\alpha,y} = \sigma^2_{\alpha,y}$

## Parameter Estimation:

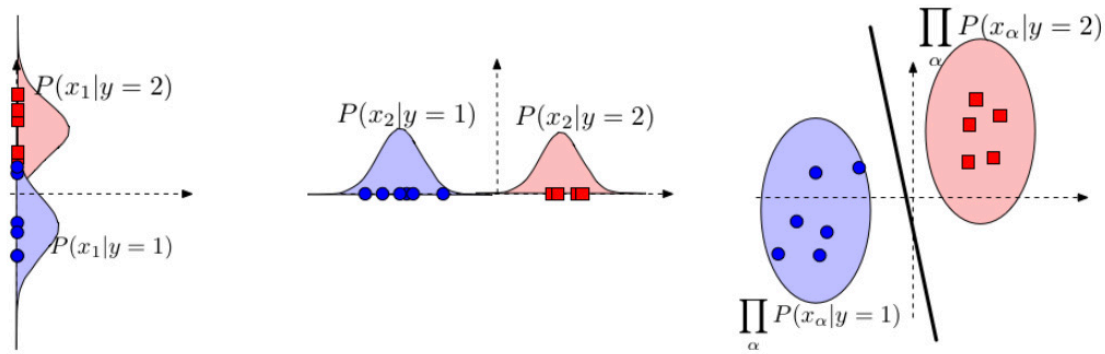
We estimate the parameters of the distributions for each dimension and class independently. Gaussian distributions only have 2 parameters.

$$\mu_{ac} \leftarrow \frac{1}{n_c} \sum_{i=1}^{n} I(y_i = c) x_{i\alpha}$$ ← avg feature value of dimension $\alpha$ from all samples w/ label $y$

where $n_c = \sum_{i=1}^{n} I(y_i = c)$

$$\sigma^2_{ac} \leftarrow \frac{1}{n_c} \sum_{i=1}^{n} I(y_i = c)(x_{i\alpha} - \mu_{ac})^2$$ ← variance of estimate $\mu_{ac}$

# Naive Bayes is a Linear Classifier




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

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

We can show that

$$h(\vec{x}) = \underset{y}{\operatorname{argmax}}\ P(y) \prod_{\alpha=1}^{d} P(x_\alpha | y) = \operatorname{sign}(\vec{w}^T \vec{x} + b)$$

That is,

$$\vec{w}^T \vec{x} + b > 0 \iff h(\vec{x}) = +1$$

As before, we define

$$P(x_\alpha | y = +1) \propto \theta_{\alpha+}^{x_\alpha} \quad \text{and} \quad P(Y = +1) = \pi_+ :$$

$$|\vec{w}|_\alpha = \log(\theta_{\alpha+}) - \log(\theta_{\alpha-})$$

$$b = \log(\pi_+) - \log(\pi_-)$$

If we use the above to do classification, we can compute $\vec{w}^T \vec{x} + b$

Simplifying further leads to

$$\vec{w}^T\vec{x} + b > 0 \iff \sum_{\alpha=1}^{d} [x]_\alpha \left(\log(\theta_{\alpha+}) - \log(\theta_{\alpha-})\right) + \log(\pi_+) - \log(\pi_-) > 0$$

exponentiate both sides

$$\iff \exp\left(\sum_{\alpha=1}^{d} [x]_\alpha \left(\log(\theta_{\alpha+}) - \log(\theta_{\alpha-})\right) + \log(\pi_+) - \log(\pi_-)\right) > 1$$

$a \log b = \log b^a$

$\exp(a-b) = \frac{\exp(a)}{\exp(b)}$

$$\iff \prod_{\alpha=1}^{d} \frac{\exp\left(\log\left(\theta_{\alpha+}^{[x]_\alpha}\right) + \log(\pi_+)\right)}{\exp\left(\log\left(\theta_{\alpha-}^{[x]_\alpha}\right) + \log(\pi_-)\right)} > 1$$

$e^{\log(a)} = a$

$e^{a+b} = e^a e^b$

$$\iff \prod_{\alpha=1}^{d} \frac{\theta_{\alpha+}^{[x]_\alpha}\pi_+}{\theta_{\alpha-}^{[x]_\alpha}\pi_-} > 1$$

$\theta_{\alpha-}^{[x]_\alpha} = P([x_\alpha] | Y=-1)$

$$\iff \frac{\prod_{\alpha=1}^{d} P([\vec{x}]_\alpha | Y=+1)\pi_+}{\prod_{\alpha=1}^{d} P([\vec{x}]_\alpha | Y=-1)\pi_-} > 1$$

Naive Bayes Assumption

$$\iff \frac{P(\vec{x} | Y=+1)\pi_+}{P(\vec{x} | Y=-1)\pi_-} > 1$$

Bayes Rule

$$\iff \frac{P(Y=+1 | \vec{x})}{P(Y=-1 | \vec{x})} > 1$$

$$\iff P(Y=+1 | \vec{x}) > P(Y=-1 | \vec{x})$$

$$\iff \operatorname{argmax}_y P(Y=y | \vec{x}) = +1$$

i.e the point $\vec{x}$ lies on positive side of hyperplane iff Naive Bayes predicts 1

2. In the case of Gaussian continuous features we can show that

$$P(y | \vec{x}) = \frac{1}{1 + e^{-y(\vec{w}^T\vec{x} + b)}}$$

← also called logistic regression

NB & LR produce asymptotically the same model if naive Bayes assumption holds.