TF-IDF, Naive Bayes

Milan Straka

November 22, 2021

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Document Representation

We already know how to represent images and categorical variables (classes, letters, words, …).

Now consider a problem of representing a whole *document*.

We usually represent a document as a **bag of words** – we create a feature space with a dimension for every unique word (or for character sequences), called **term**.

However, there are many possible ways how the values of the terms might be set.

Term Frequency – Inverse Document Frequency

Commonly used ways of setting the term values:

- **binary indicators**: 1/0 depending on whether a term is present in a document or not;
- **term frequency (TF)**: relative frequency of a term in a document;

$$TF(t;d) = \frac{\text{number of occurrences of } t \text{ in the document } d}{\text{number of terms in the document } d}$$

- **inverse document frequency (IDF)**: we could also represent a term using self-information of a probability of a random document containing it (therefore, terms with lower document probability have higher weights);

$$IDF(t) = \log \frac{\text{number of documents}}{\text{number of documents containing } t \ \big(\text{optionally} + 1\big)} = I\big(P(d \ni t)\big)$$

- **TF-IDF**: empirically, product $TF \cdot IDF$ is a feature reflecting quite well how important is a term to a document in a corpus (used by 83% text-based recommender systems in 2015).

Mutual Information

Consider two random variables x and y with distributions $\mathrm{x} \sim X$ and $\mathrm{y} \sim Y$.

The conditional entropy $H(Y|X)$ can be naturally considered an expectation of a self-information of $Y|X$, so in the discrete case,

$$H(Y|X) = \mathbb{E}_{x,y}\left[I(y|x)\right] = -\sum_{x,y} P(x,y) \log P(y|x).$$

In order to assess the amount of information *shared* between the two random variables, we might consider the difference

$$H(Y) - H(Y|X) = \mathbb{E}_{x,y}\big[-\log P(y)\big] - \mathbb{E}_{x,y}\big[-\log P(y|x)\big] = \mathbb{E}_{x,y}\left[\log \frac{P(x,y)}{P(x)P(y)}\right].$$

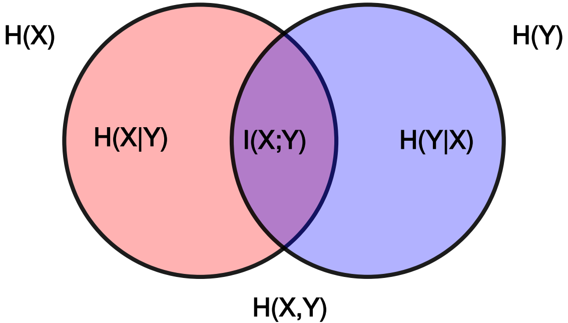
We can interpret this value as

> *How many bits of information will we learn about Y when we find out X?*

Mutual Information

Let us denote this quantity as the **mutual information** $I(X;Y)$:

$$I(X;Y) = \mathbb{E}_{x,y}\left[\log \frac{P(x,y)}{P(x)P(y)}\right].$$



https://commons.wikimedia.org/wiki/File:Entropy-mutual-information-relative-entropy-relation-diagram.svg

- The mutual information is symmetrical, so

$$I(X;Y) = I(Y;X) = H(Y) - H(Y|X) = H(X) - H(X|Y).$$

- It is easy to verify that

$$I(X;Y) = D_{\mathrm{KL}}\big(P(X,Y) \| P(X)P(Y)\big).$$

  Therefore,
  - $I(X;Y) \geq 0$,
  - $I(X;Y) = 0$ iff $P(X,Y) = P(X)P(Y)$ iff the random variables are independent.

TF-IDF as Mutual Information

Let $\mathcal{D}$ be a collection of N documents and $\mathcal{T}$ collections of terms.

Our assumption is that whenever we need to draw a document, we do it uniformly randomly. Therefore,

- $P(d) = 1/|\mathcal{D}|$ and $I(d) = H(\mathcal{D}) = \log|\mathcal{D}|$,
- $P(d|t) = 1/|\{d \in \mathcal{D} : t \in d\}|$ and $I(d|t) = H(\mathcal{D}|\mathcal{T} = t) = \log|\{d \in \mathcal{D} : t \in d\}|$,
- $I(d) - I(d|t) = H(\mathcal{D}) - H(\mathcal{D}|\mathcal{T} = t) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|} = \mathit{IDF}(t)$.

Finally, we can compute the mutual information $I(\mathcal{D};\mathcal{T})$ as

$$I(\mathcal{D};\mathcal{T}) = \sum_{d,t} P(d) \cdot P(t|d) \cdot \big(I(d) - I(d|t)\big) = \frac{1}{|\mathcal{D}|} \sum_{d,t} \mathit{TF}(t;d) \cdot \mathit{IDF}(t).$$

Therefore, summing all TF-IDF terms recovers the mutual information between $\mathcal{D}$ and $\mathcal{T}$, and we can say that each TF-IDF carries a “bit of information” attached to a document-term pair.

Bayesian Probability

Until now, we considered the so-called *frequentist probability*, where the probability of an event is considered a limit of its frequency.

In *Bayesian probability* interpretation, probability is a quantification of uncertainty instead. Bayesian probability can be considered an extension of propositional logic, where hypotheses (that must be true or false in frequentist probability) can be assigned probabilities.

Bayesian probability is the so-called *evidential* probability, where hypotheses have some initial **prior probability**, which is then updated in light of *new data* into **posterior probability**.

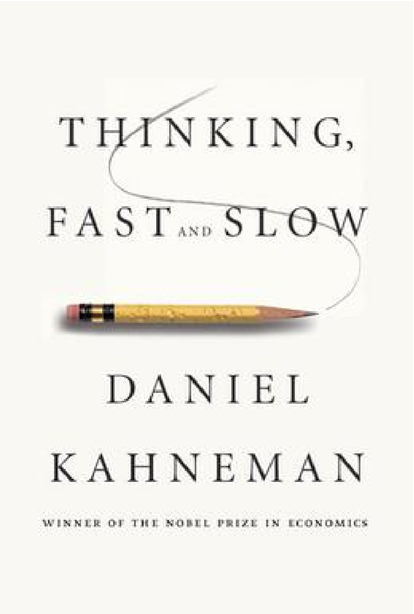
This update of prior probability into posterior probability is performed using the Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Librarian or Farmer

The following problem is from the Thinking, Fast and Slow:

> *As you consider the next question, please assume that Steve was selected at random from a representative sample. An individual has been described by a neighbor as follows: "Steve is very shy and withdrawn, invariably helpful but with little interest in people or in the world of reality. A meek and tidy soul, he has a need for order and structure, and a passion for detail." Is Steve more likely to be a librarian or a farmer?*



https://en.wikipedia.org/wiki/File:Thinking,_Fast_and_Slow.jpg

The given description corresponds more to a librarian than to a farmer.

However, there are many more farmers than librarians (for example, in 2016 there were 4.33k librarians and 130.3k regular agricultural workers in the Czech Republic, a 30:1 ratio).

The description being more fitting for a librarian is in fact a *likelihood*, while the base rates of librarians and farmers play the role of a *prior*, and the whole question asks about the *posterior*:

$$P(librarian|description) \propto P(description|librarian) \cdot P(librarian).$$

Maximum A Posteriori Estimation

We demonstrate the Bayesian probability on model fitting.

Recall the maximum likelihood estimation

$$\boldsymbol{w}_{\mathrm{MLE}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{X}; \boldsymbol{w}) = \arg\max_{\boldsymbol{w}} p(\boldsymbol{X}|\boldsymbol{w}).$$

In the Bayesian interpretation, we capture our initial assumptions about $\boldsymbol{w}$ using a prior probability $p(\boldsymbol{w})$. The effect of observing the data $\boldsymbol{X}$ can be then expressed as

$$p(\boldsymbol{w}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{X})}.$$

The quantity $p(\boldsymbol{X}|\boldsymbol{w})$ is evaluated using fixed data $\boldsymbol{X}$ and quantifies how probable the observed data is with respect to various values of the parameter $\boldsymbol{w}$. It is therefore a **likelihood**, because it is a function of $\boldsymbol{w}$.

Maximum A Posteriori Estimation

Therefore, we get that

$$\underbrace{p(\boldsymbol w|\boldsymbol X)}_{\text{posterior}} \propto \underbrace{p(\boldsymbol X|\boldsymbol w)}_{\text{likelihood}} \cdot \underbrace{p(\boldsymbol w)}_{\text{prior}},$$

where the symbol $\propto$ means "up to a multiplicative factor".

Using the above Bayesian inference formula, we can define **maximum a posteriori (MAP)** estimate as

$$\boldsymbol w_\mathrm{MAP} = \arg\max_{\boldsymbol w} p(\boldsymbol w|\boldsymbol X) = \arg\max_{\boldsymbol w} p(\boldsymbol X|\boldsymbol w)p(\boldsymbol w).$$

To utilize the MAP estimate for model training, we need to specify the parameter prior $p(\boldsymbol w)$, our *preference* among models.

Note that a possible view is that overfitting is just a problem of not using priors and that suitable priors would avoid it.

L2 Regularization as MAP

Frequently, the mean is assumed to be zero, and the variance is assumed to be σ^2. Given that we have no further information, we employ the maximum entropy principle, which provides us with $p(w_i) = \mathcal{N}(w_i; 0, \sigma^2)$, so that $p(\boldsymbol w) = \prod_i \mathcal{N}(w_i; 0, \sigma^2) = \mathcal{N}(\boldsymbol w; \boldsymbol 0, \sigma^2 \boldsymbol I)$. Then

$$\begin{aligned}\boldsymbol w_\mathrm{MAP} &= \operatorname*{arg\,max}_{\boldsymbol w} p(\boldsymbol X|\boldsymbol w)p(\boldsymbol w)\\ &= \operatorname*{arg\,max}_{\boldsymbol w} \prod\nolimits_{i=1}^N p(\boldsymbol x_i|\boldsymbol w)p(\boldsymbol w)\\ &= \operatorname*{arg\,min}_{\boldsymbol w} \sum\nolimits_{i=1}^N \Big(-\log p(\boldsymbol x_i|\boldsymbol w) - \log p(\boldsymbol w)\Big).\end{aligned}$$

By substituting the probability of the Gaussian prior, we get

$$\boldsymbol w_\mathrm{MAP} = \operatorname*{arg\,min}_{\boldsymbol w} \sum\nolimits_{i=1}^N -\log p(\boldsymbol x_i|\boldsymbol w) - \frac{c}{2}\log(2\pi\sigma^2) + \frac{\|\boldsymbol w\|^2}{2\sigma^2},$$

which is in fact the L_2-regularization.

Bernoulli and Binomial Distribution

We have already discussed the Bernoulli distribution, which is a distribution over a binary random variable with a single parameter $\varphi \in [0, 1]$, which specifies the probability of the random variable being equal to 1.

If the Bernoulli trial is repeated multiple times n, the resulting outcome is the number of successes $\in \{0, 1, \ldots, n\}$ and has a **binomial distribution** $B(n, \varphi)$.

If $\mathrm{x} \sim B(n, \varphi)$, then

$$P(\mathrm{x} = k) = \binom{n}{k} \varphi^k (1 - \varphi)^{n-k},$$

where $\binom{n}{k}$ is the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Bernoulli and Binomial Distribution

If we observe N outcomes of a Bernoulli trial (or equivalently a single outcome of a binomially-distributed N-trial random variable), we can use MLE to estimate the parameter φ.

In the context of Bayesian inference, we would start with some prior $p(\varphi)$ and then compute a posterior after any amount of observed data – be it a single trial or several trials at once. In the extreme case, we can compute a posterior after every single observed data.

This sequential nature of Bayesian inference makes it practical, if for a given prior and likelihood, the posterior comes from the same distribution family as the prior. Such a distribution is then called a **conjugate prior** of a given likelihood function.

Conjugate Prior of a Bernoulli Distribution

To derive a conjugate prior of a Bernoulli distribution, recall that for a Bernoulli-distributed random variable $P(x) = \varphi^x(1-\varphi)^{1-x}$.

Therefore, if the prior would be a product of the φ and $(1-\varphi)$ factors, it would keep the same form after being multiplied by the likelihood. Therefore, we seek for a prior of a form

$$\mathrm{Beta}(x; \alpha, \beta) \propto x^{\alpha-1}(1-x)^{\beta-1}.$$

We still need to compute a normalization constant so that the distribution will integrate to 1. Its value is called the **Beta function**

$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\,\mathrm{d}x = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

and the conjugate prior **Beta** of a Bernoulli distribution is

$$\mathrm{Beta}(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}.$$

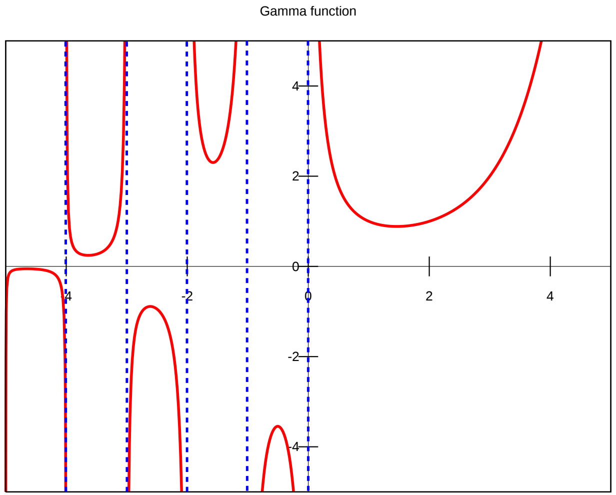
Gamma Function

The $\Gamma(x)$ used in the beta function is the **Gamma function**, which is the commonly-used extension of factorial to complex numbers fulfilling

$$\Gamma(n) = (n-1)! \text{ for any } n \in \mathbb{N}.$$

It is defined as

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \,\mathrm{d}x$$



https://commons.wikimedia.org/wiki/File:Gamma_plot.svg

and we can verify that

- $\Gamma(1) = \int_0^\infty e^{-x} \,\mathrm{d}x = \big[-e^{-x}\big]_0^\infty = \lim_{x\rightarrow\infty}\big(-e^{-x}\big) - \big(-e^0\big) = 0 + 1 = 1,$
- $\Gamma(z+1) = \int_0^\infty x^z e^{-x} \,\mathrm{d}x$ and using integration by parts, $\Gamma(z+1) = \big[-x^z e^{-x}\big]_0^\infty - \int_0^\infty -z x^{z-1} e^{-x} \,\mathrm{d}x = 0 + z\int_0^\infty x^{z-1} e^{-x} \,\mathrm{d}x = z\Gamma(z).$

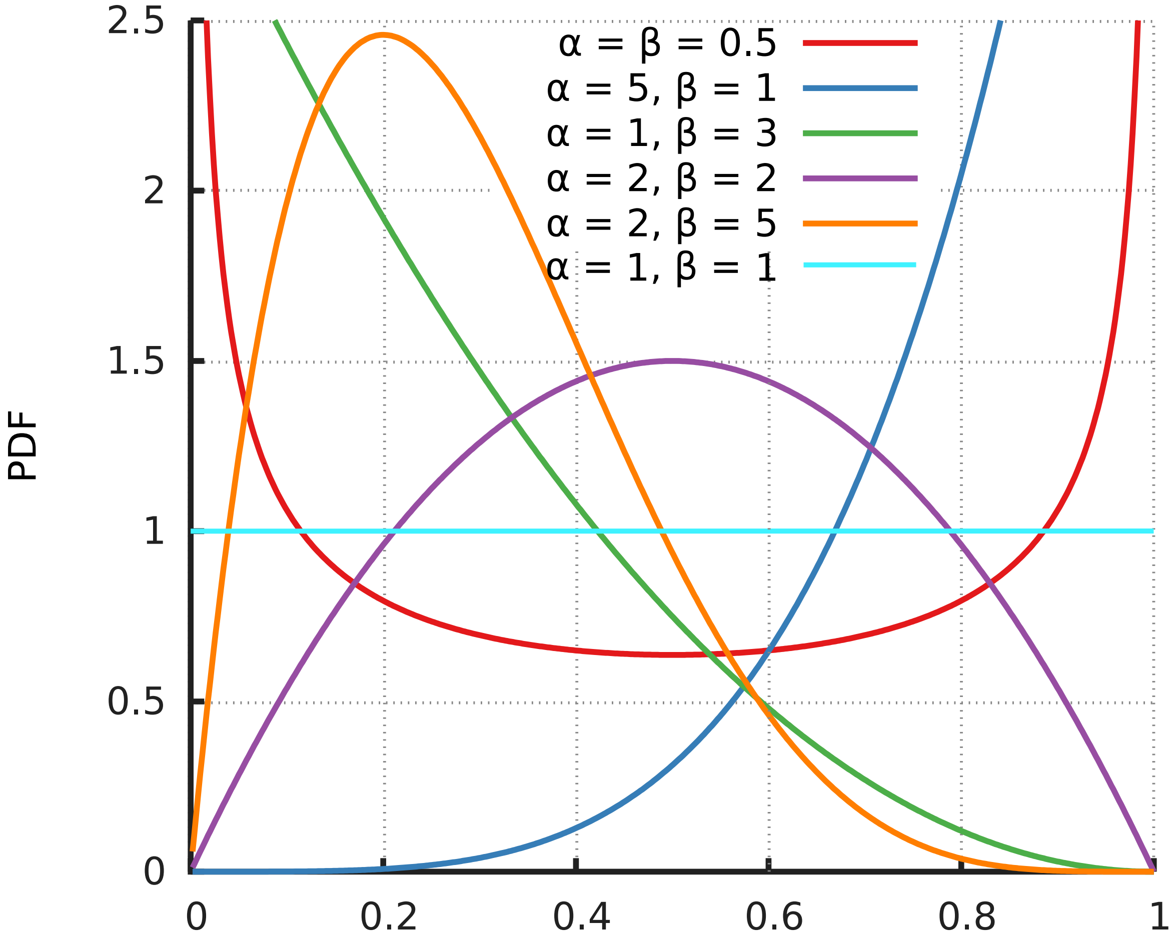
Conjugate Prior of a Bernoulli Distribution

If we have a prior $\mathrm{Beta}(\alpha, \beta)$ and we observe k successes and l failures, the posterior distribution is $\mathrm{Beta}(\alpha + k, \beta + l)$.

Therefore, the α and β parameters can be considered "counts" of successes and failures.

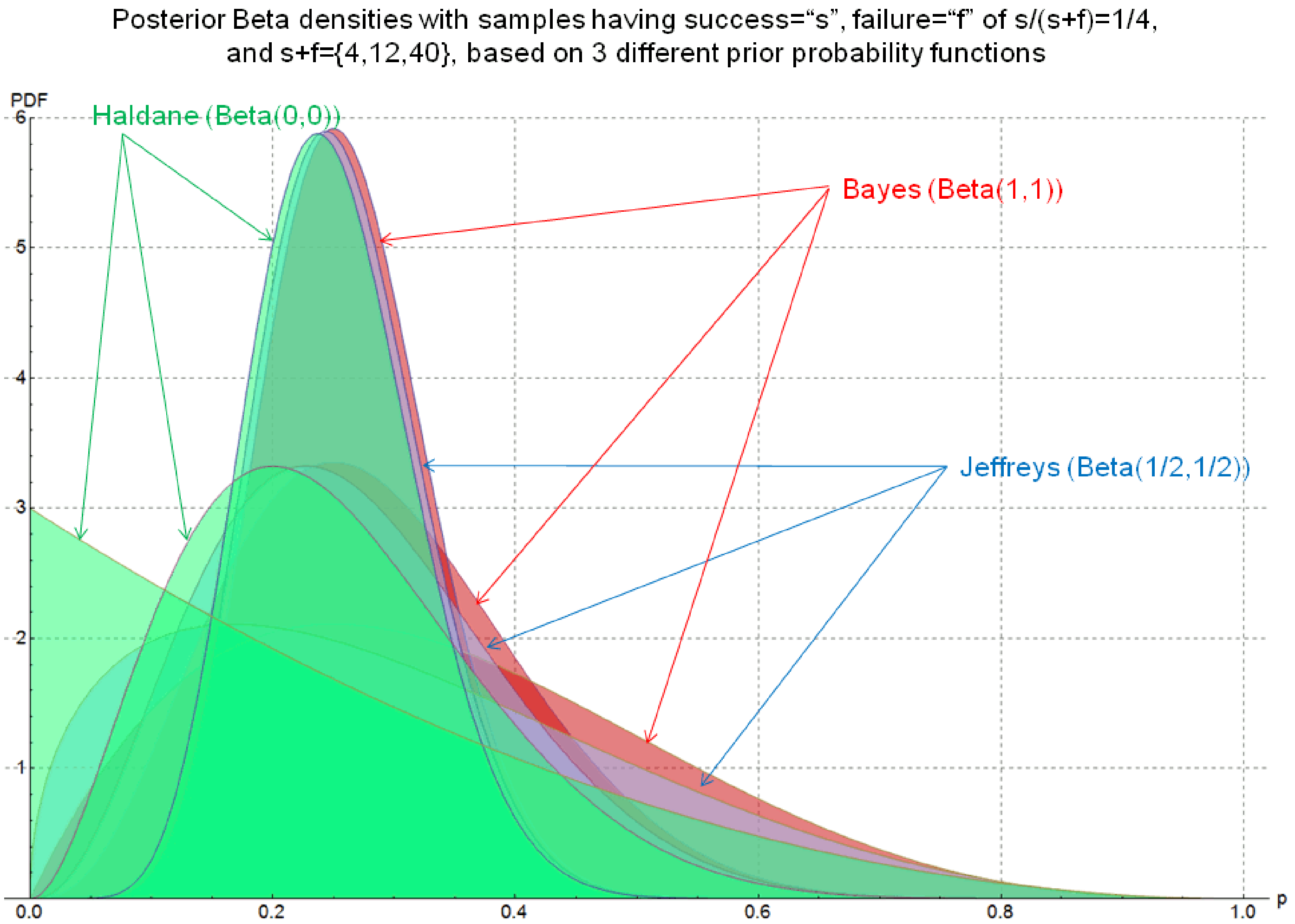
Therefore, the prior corresponds to adding some number of "pseudo-observations".

Note that $\mathrm{Beta}(1, 1)$ is uniform, $\mathrm{Beta}(\alpha, \beta)$ corresponds to $\alpha - 1$ successes and $\beta - 1$ failures and the mode ($\arg\max$) of $\mathrm{Beta}(\alpha, \beta)$ for $\alpha, \beta > 1$ is $(\alpha - 1)/(\alpha + \beta - 2)$.



https://commons.wikimedia.org/wiki/File:Beta_distribution_pdf.svg

Conjugate Prior of a Bernoulli Distribution



https://commons.wikimedia.org/wiki/File:Beta_distribution_for_3_different_prior_probability_functions,_skewed_case_sample_size_=_(4,12,40)_-_J._Rodal.png

Multinomial and Dirichlet Distribution

Similarly to how the binomial distribution models outcomes of n independent Bernoulli trials, the **multinomial distribution** generalizes the categorical distribution by considering n trials.

The multinomial distribution is again parametrized with a probability distribution $\boldsymbol{p} \in [0, 1]^K$ and a number of trials $n \in \mathbb{N}$, and the probability of x_k outcomes of category k is

$$P(\boldsymbol{x}) = \binom{n}{x_1\ x_2\ \ldots\ x_K} p_1^{x_1} p_2^{x_2} \cdots p_K^{x_K},$$

where $\binom{n}{x_1\ x_2\ \ldots\ x_K}$ is the multinomial coefficient $\binom{n}{x_1\ x_2\ \ldots\ x_K} = \frac{n!}{x_1! x_2! \cdots x_K!}$.

The conjugate prior of the categorical distribution is a generalization of the beta distribution – the **Dirichlet distribution**

$$\operatorname{Dir}(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{\Gamma(\Sigma_i\, \alpha_i)}{\Pi_i\, \Gamma(\alpha_i)} \cdot \prod\nolimits_i x_i^{\alpha_i - 1},$$

where the $\boldsymbol{\alpha}$ play again the role of the (pseudo-)counts of the individual classes.

Naive Bayes Classifier

So far, our classifiers were so-called **discriminative** and had a form

$$p(C_k|\boldsymbol{x}) = p(C_k|x_1, x_2, \ldots, x_D).$$

Instead, we might use the Bayes' theorem and rewrite to

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})}.$$

Then, classification could be performed as

$$\arg\max_k p(C_k|\boldsymbol{x}) = \arg\max_k \frac{p(\boldsymbol{x}|C_k)p(C_k)}{p(\boldsymbol{x})} = \arg\max_k p(\boldsymbol{x}|C_k)p(C_k).$$

Therefore, instead of modeling $p(C_k|\boldsymbol{x})$, we model

- the prior $p(C_k)$ according to the distribution of classes in the data, and
- the distribution $p(\boldsymbol{x}|C_k)$.

Naive Bayes Classifier

Modeling the distribution $p(\boldsymbol{x}|C_k)$ is however difficult – $\boldsymbol{x}$ can be high-dimensional high-structured data.

Therefore, the so-called **Naive Bayes classifier** assumes that

> *all x_d are independent given C_k,*

so we can rewrite

$$p(\boldsymbol{x}|C_k) = p(x_1|C_k)p(x_2|C_k, x_1)p(x_3|C_k, x_1, x_2)\cdots p(x_D|C_k, x_1, x_2, \ldots)$$

to

$$p(\boldsymbol{x}|C_k) = \prod\nolimits_{d=1}^{D} p(x_d|C_k).$$

Notice that modeling $p(x_d|C_k)$ is substantially easier because it is a distribution over a single-dimensional quantity.

Naive Bayes Classifier

There are in fact several naive Bayes classifiers, depending on the distribution $p(x_d|C_k)$.

Gaussian Naive Bayes

In Gaussian naive Bayes, we expect a continuous feature to have normal distribution for a given C_k, and model $p(x_d|C_k)$ is modeled as a normal distribution $\mathcal{N}(\mu_{d,k}, \sigma^2_{d,k})$.

Assuming we have the training data $\boldsymbol{X}$ together with K-class classification targets $\boldsymbol{t}$, the "training" phase consists of estimating the parameters $\mu_{d,k}$ and $\sigma^2_{d,k}$ of the distributions $\mathcal{N}(\mu_{d,k}, \sigma^2_{d,k})$ for $1 \le d \le D$, $1 \le k \le K$, employing the maximum likelihood estimation.

Now let feature d and class k be fixed and let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_k}$ be the training data *corresponding to the class* k. We already know that maximum likelihood estimation using N_k samples drawn from a Gaussian distribution $\mathcal{N}(\mu_{d,k}, \sigma^2_{d,k})$ amounts to

$$\mathop{\arg\min}_{\mu_{d,k}, \sigma_{d,k}} \frac{N_k}{2} \log(2\pi\sigma^2_{d,k}) + \sum_{i=1}^{N_k} \frac{(x_{i,d} - \mu_{d,k})^2}{2\sigma^2_{d,k}}.$$

Gaussian Naive Bayes

Setting the derivative with respect to $\mu_{d,k}$ to zero results in

$$0 = \sum\nolimits_{i=1}^{N_k} \frac{-2(x_{i,d} - \mu_{d,k})}{2\sigma_{d,k}^2},$$

which we can rewrite to $\mu_{d,k} = \frac{1}{N_k}\sum\nolimits_{i=1}^{N_k} x_{i,d}$.

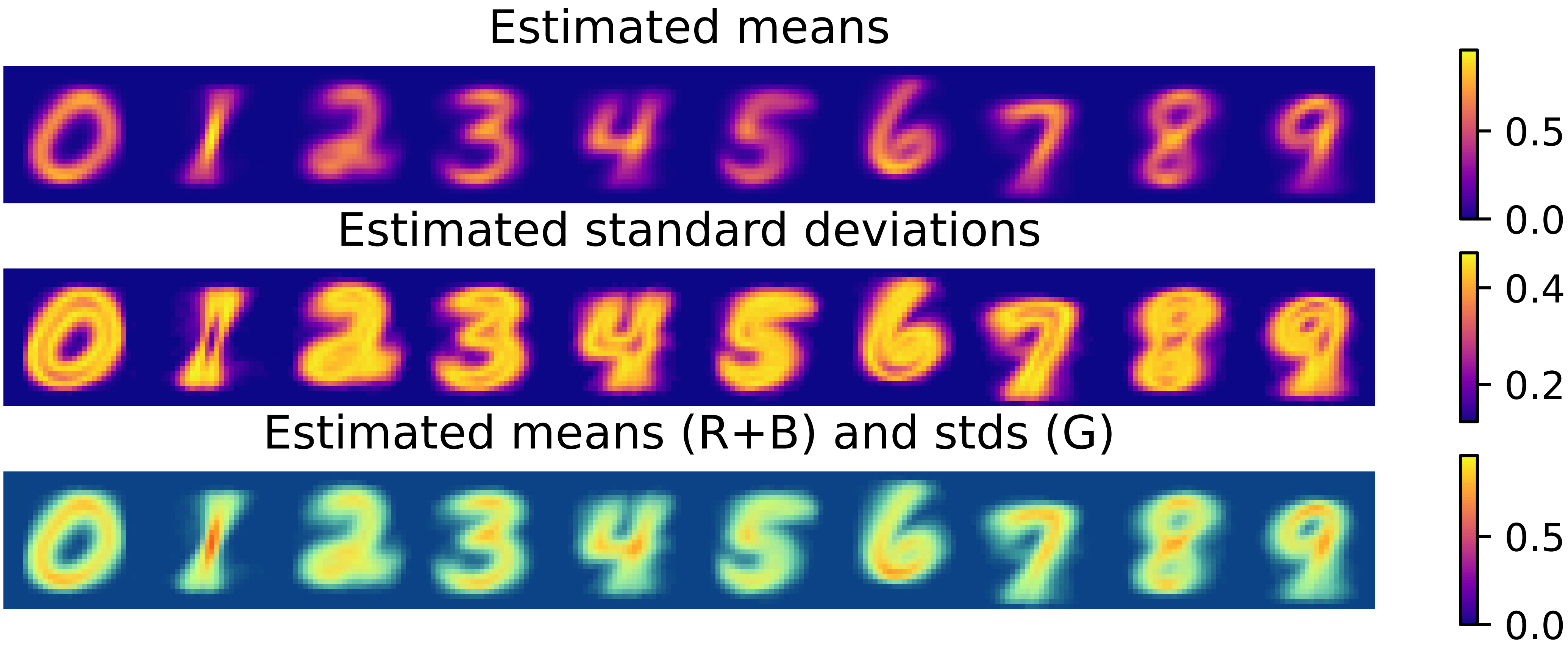
Similarly, zeroing out the derivative with respect to $\sigma_{d,k}^2$ gives

$$0 = \frac{N_k}{2\sigma_{d,k}^2} - \frac{1}{2(\sigma_{d,k}^2)^2}\sum\nolimits_{i=1}^{N_k}(x_{i,d} - \mu_{d,k})^2,$$

from which we obtain $\sigma_{d,k}^2 = \frac{1}{N_k}\sum\nolimits_{i=1}^{N_k}(x_{i,d} - \mu_{d,k})^2$.

However, the variances are usually smoothed (increased) by a given constant α to avoid too sharp distributions (in Scikit-learn, the default value of α is 10^{-9} times the largest variance of all features).

Gaussian Naive Bayes Example



Means and standard deviations estimated by Gaussian NB on a subset of the MNIST dataset.

Bernoulli Naive Bayes

When the input features are binary, the $p(x_d|C_k)$ might be modeled using a Bernoulli distribution

$$p(x_d|C_k) = p_{d,k}^{x_d} \cdot (1 - p_{d,k})^{(1-x_d)}.$$

We can therefore write

$$p(C_k|\boldsymbol{x}) \propto \Big(\prod\nolimits_{d=1}^{D} p_{d,k}^{x_d} \cdot (1 - p_{d,k})^{(1-x_d)}\Big) p(C_k),$$

and by computing a logarithm we get

$$\log p(C_k|\boldsymbol{x}) + c = \log p(C_k) + \sum\nolimits_d \left(x_d \log \tfrac{p_{d,k}}{1-p_{d,k}} + \log(1 - p_{d,k})\right) = b_k + \boldsymbol{x}^T \boldsymbol{w}_k,$$

where the constant c does not depend on C_k and is therefore not needed for prediction

$$\arg\max\nolimits_k \log p(C_k|\boldsymbol{x}) = \arg\max\nolimits_k b_k + \boldsymbol{x}^T \boldsymbol{w}_k.$$

Bernoulli Naive Bayes Estimation

To estimate the probabilities $p_{d,k}$, we turn again to the maximum likelihood estimation. The log-likelihood of N_k samples drawn from Bernoulli distribution with parameter $p_{d,k}$ is

$$\sum\nolimits_{i=1}^{N_k} \log\left(p_{d,k}^{x_{i,d}}(1-p_{d,k})^{1-x_{i,d}}\right) = \sum\nolimits_{i=1}^{N_k} \left(x_{i,d}\log p_{d,k} + (1-x_{i,d})\log(1-p_{d,k})\right).$$

Setting the derivative with respect to $p_{d,k}$ to zero, we obtain

$$0 = \sum\nolimits_{i=1}^{N_k} \left(\frac{x_{i,d}}{p_{d,k}} - \frac{1-x_{i,d}}{1-p_{d,k}}\right) = \frac{1}{p_{d,k}(1-p_{d,k})}\sum\nolimits_{i=1}^{N_k} \Big((1-p_{d,k})x_{i,d} - p_{d,k}(1-x_{i,d})\Big),$$

giving us $p_{d,k} = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{i,d}$.

Bernoulli Naive Bayes Estimation

We could therefore estimate the probabilities $p_{d,k}$ as

$$p_{d,k} = \frac{\text{number of documents of class } k \text{ with nonzero feature } d}{\text{number of documents of class } k}.$$

However, if a feature d is always set to one (or zero) for a given class k, then $p_{d,k} = 1$ (or 0). That is impractical because the resulting classifier would give probability zero to inputs with the opposite value of such a feature.

Therefore, **Laplace** or **additive smoothing** is used, and the probability $p_{d,k}$ estimated as

$$p_{d,k} = \frac{\text{number of documents of class } k \text{ with nonzero feature } d + \alpha}{\text{number of documents of class } k + 2\alpha}$$

for some pseudo-count $\alpha > 0$.

Note that even if this technique has a special name, it corresponds to using a *maximum a posteriori* estimate, using $\operatorname{Beta}(\alpha+1, \alpha+1)$ as a prior distribution.

Multinomial Naive Bayes

The last variant of naive Bayes we will describe is the **multinomial naive Bayes**, where $p(\boldsymbol{x}|C_k)$ is modeled to be multinomial distribution, $p(\boldsymbol{x}|C_k) \propto \prod_d p_{d,k}^{x_d}$.

Similarly to the Bernoulli NB case, we can write the log-likelihood as

$$\log p(C_k|\boldsymbol{x}) + c = \log p(C_k) + \sum\nolimits_d x_d \log p_{d,k} = b_k + \boldsymbol{x}^T \boldsymbol{w}_k.$$

Multinomial Naive Bayes Estimation

As in the previous cases, we turn to the maximum likelihood estimation in order to find out the values of $p_{d,k}$. We start with the log-likelihood

$$\sum_{i=1}^{N_k} \log\Big(\prod\nolimits_d p_{d,k}^{x_{i,d}}\Big) = \sum\nolimits_{i,d} x_{i,d} \log p_{d,k}.$$

To maximize this quantity with respect to a probability distribution $\sum_d p_{d,k} = 1$, we need to form a *Lagrangian*

$$\mathcal{L} = \sum\nolimits_{i,d} x_{i,d} \log p_{d,k} + \lambda\Big(1 - \sum\nolimits_d p_{d,k}\Big).$$

Setting the derivative with respect to $p_{d,k}$ to zero results in $0 = \sum_{i=1}^{N_k} \frac{x_{i,d}}{p_{d,k}} - \lambda$, so

$$p_{d,k} = \frac{1}{\lambda} \sum_{i=1}^{N_k} x_{i,d} = \frac{\sum_{i=1}^{N_k} x_{i,d}}{\sum_{i=1}^{N_k} \sum_d x_{i,d}}, \text{ where } \lambda \text{ is set to fulfill } \sum\nolimits_d p_{d,k} = 1.$$

Multinomial Naive Bayes Estimation

Denoting $n_{d,k}$ as the sum of features x_d for a class C_k, the probabilities $p_{d,k}$ could be therefore estimated as

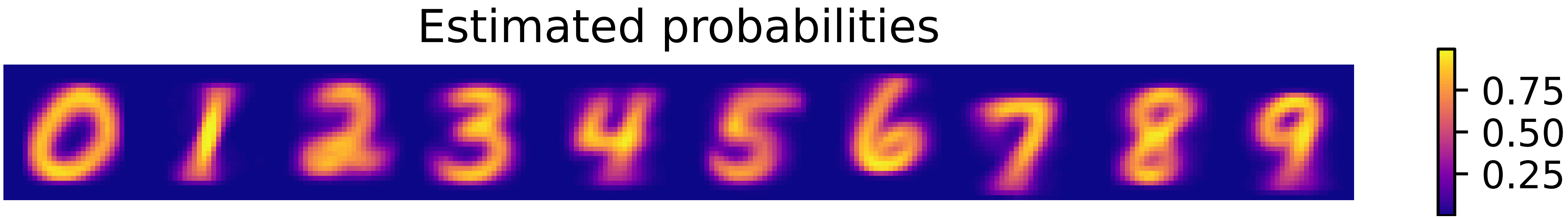
$$p_{d,k} = \frac{n_{d,k}}{\sum_j n_{j,k}}.$$

However, for the same reasons as in the Bernoulli NB case, we also use the Laplace smoothing, i.e., utilize a Dirichlet prior $\operatorname{Dir}(\alpha+1)$, and instead use

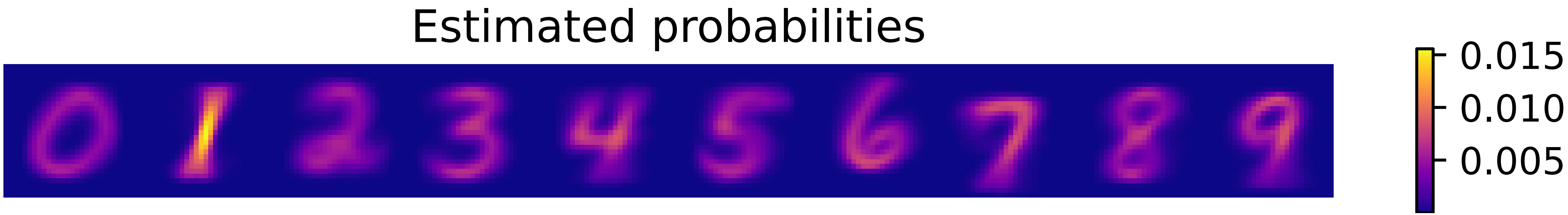
$$p_{d,k} = \frac{n_{d,k}+\alpha}{\sum_j (n_{j,k}+\alpha)} = \frac{n_{d,k}+\alpha}{\left(\sum_j n_{j,k}\right)+\alpha D}$$

with pseudo-count $\alpha > 0$.

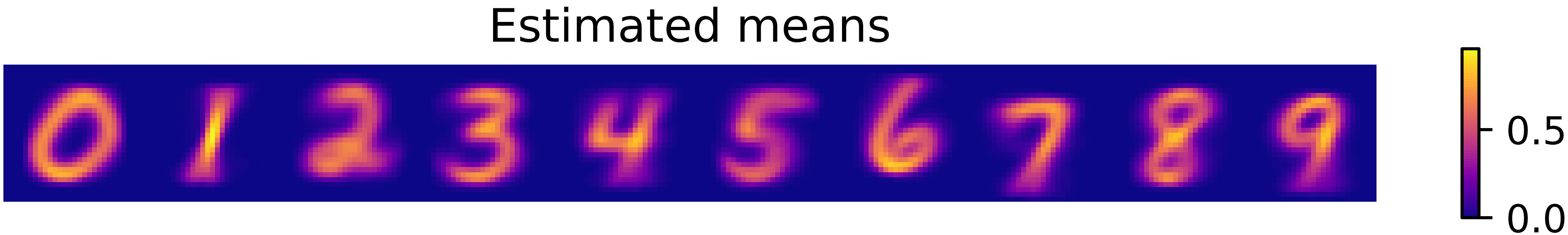
Naive Bayes Example



Probabilities estimated by Bernoulli NB on a subset of the MNIST dataset.



Probabilities estimated by multinomial NB on a subset of the MNIST dataset.



Means estimated by Gaussian NB on a subset of the MNIST dataset.

Naive Bayes Conclusions

The choice among the Gaussian, Bernoulli and multinomial naive Bayes depends on the feature values.

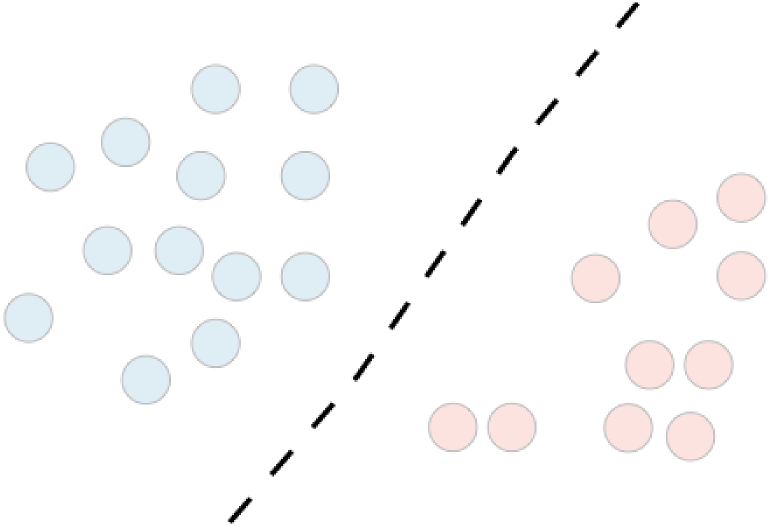
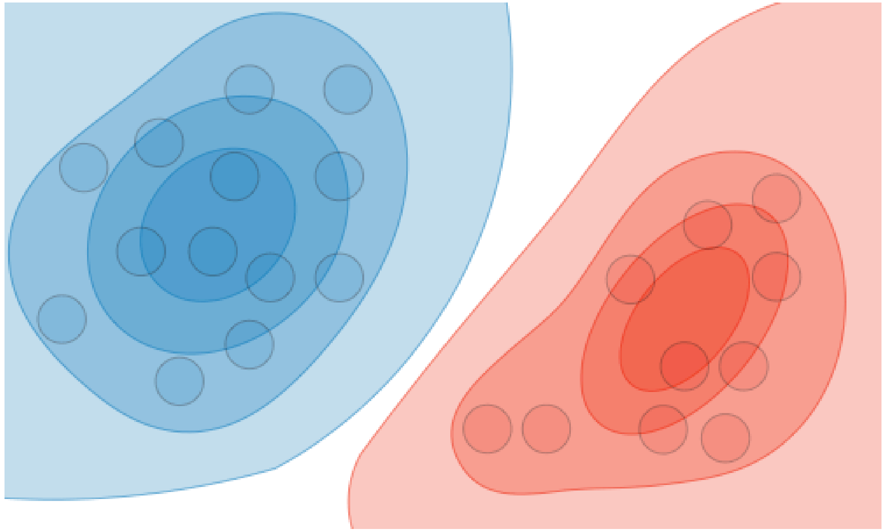
- If we expect the individual feature values to be roughly normally distributed, Gaussian NB is an obvious choice.
- To use multinomial NB, the features should roughly follow the multinomial distribution – i.e., they must be non-negative, be interpretable as “counts” and “compete” with each other.
    - Note that the feature can be real-valued (the multinomial distribution can be extended to real-value observations using the Γ function).
- In order to use Bernoulli NB, the features *must* be binary. However, an important difference is that contrary to the multinomial NB, the **absence of features** is also modeled by the $(1 - p_{d,k})$ term; the multinomial NB uses $p_{d,k}^0 = 1$ in such a case.

Generative and Discriminative Models

So far, all our classification models (but naive Bayes) have been **discriminative**, modeling a *conditional distribution* $p(t|\boldsymbol{x})$ (predicting some output distribution).

On the other hand, the **generative models** estimate *joint distribution* $p(t, \boldsymbol{x})$, often by employing Bayes' theorem and estimating $p(\boldsymbol{x}|t) \cdot p(t)$. They therefore model the probability of the data being generated by an outcome, and only transform it to $p(t|\boldsymbol{x})$ during prediction.

Generative and Discriminative Models

	Discriminative Model	Generative Model
Goal	Estimate $P(t\|\boldsymbol{x})$	Estimate $P(t, \boldsymbol{x}) = P(\boldsymbol{x}\|t)P(t)$
What's learned	Decision boundary	Probability distribution of the data
Illustration	 *https://stanford.edu/~shervine/teaching/cs-229/illustrations/discriminative-model.png*	 *https://stanford.edu/~shervine/teaching/cs-229/illustrations/generative-model.png*

Generative and Discriminative Models

- Empirically, discriminative models perform better in classification tasks, because modeling the decision boundary is often much easier than modeling the data distribution.
- On the other hand, generative models can recognize anomalies/outliers/out-of-distribution data (when the input example has low probability under the data distribution).
- The term *generative* comes from a (theoretical) possibility of "generating" random instances of $\boldsymbol{x}$ and t. However, just being able to evaluate $p(\boldsymbol{x}|\boldsymbol{t})$ does not necessarily mean there is an *efficient* procedure of actually sampling (generating) $\boldsymbol{x}$.
  - In recent years, generative modeling combined with deep neural networks created a new family of *deep generative models* like VAE or GAN, which can in fact efficiently generate samples from $p(\boldsymbol{x})$.

Figure 1 from paper "Large Scale GAN Training for High Fidelity Natural Image Synthesis", https://arxiv.org/abs/1809.11096.

LogReg/NaiveBayes Generative-Discriminative Pair

Given that

- multinomial/Bernoulli naive Bayes fits $\log p(C_k, \boldsymbol{x})$ as a linear model, and
- a logistic regression also fits $\log p(C_k | \boldsymbol{x})$ as a linear model,

multinomial/Bernoulli NB and logistic regression form a so-called **generative-discriminative** pair.

Several theorems are known about this generative-discriminative pair (for proofs see the 2002 paper *On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes* by NG and Jordan):

- If the assumed model in naive Bayes is correct, then both logistic regression and naive Bayes converge to the same performance.
- Asymptotically, logistic regression is always better or equal to the naive Bayes.
- Let $\varepsilon > 0$ be given and let the model contain D features.
  - Logistic regression can reach the optimal error up to ε with $\Omega(D)$ training examples.
  - naive Bayes can reach the optimal error up to ε with $\Omega(\log(D))$ examples.

LogReg/NaiveBayes Generative-Discriminative Pair

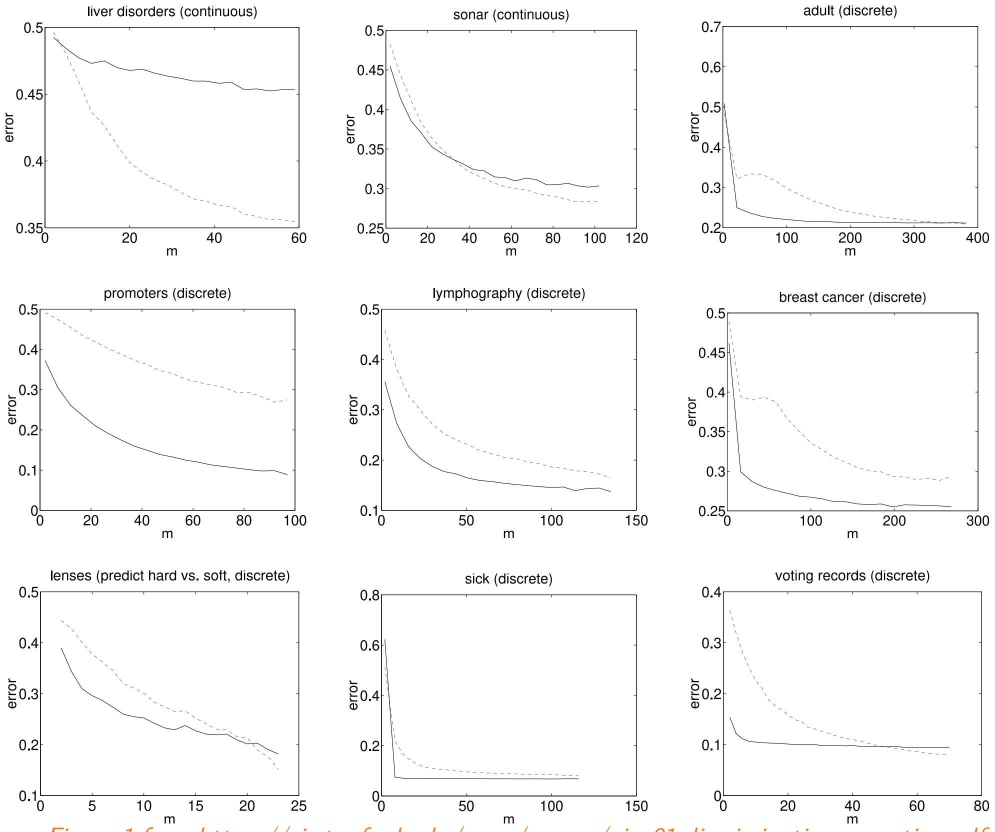


Figure 1 from https://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf.

The results of experiments from the 2002 paper *On Discriminative vs. Generative Classifiers* by NG and Jordan. The generalization error of logistic regression (dashed lines) and naive Bayes (solid lines) are plotted with respect to the number of training examples m.