

k-NN, Naive Bayes

Jindřich Libovický (reusing materials by Milan Straka)

 November 10, 2025

After this course you should...

- Be able to reason about task/problems **suitable for ML**
 - Know when to use classification, regression and clustering
 - Be able to choose from this method Linear and Logistic Regression, Multilayer Perceptron, Nearest Neighbors, Naive Bayes, Gradient Boosted Decision Trees, k -means clustering
- Think about learning as (mostly probabilistic) **optimization on training data**
 - Know how the ML methods learn including theoretical explanation
- Know how to properly **evaluate** ML
 - Think about generalization (and avoiding overfitting)
 - Be able to choose a suitable evaluation metric
 - Responsibly decide what model is better
- Be able to **implement ML algorithms** on a conceptual level
- Be able to **use Scikit-learn** to solve ML problems in Python

After this lecture you should be able to

- Implement and use k -nearest neighbors for classification and regression
- Explain the very basic principles of Bayesian thinking
- Implement and use Naive Bayes Classifier

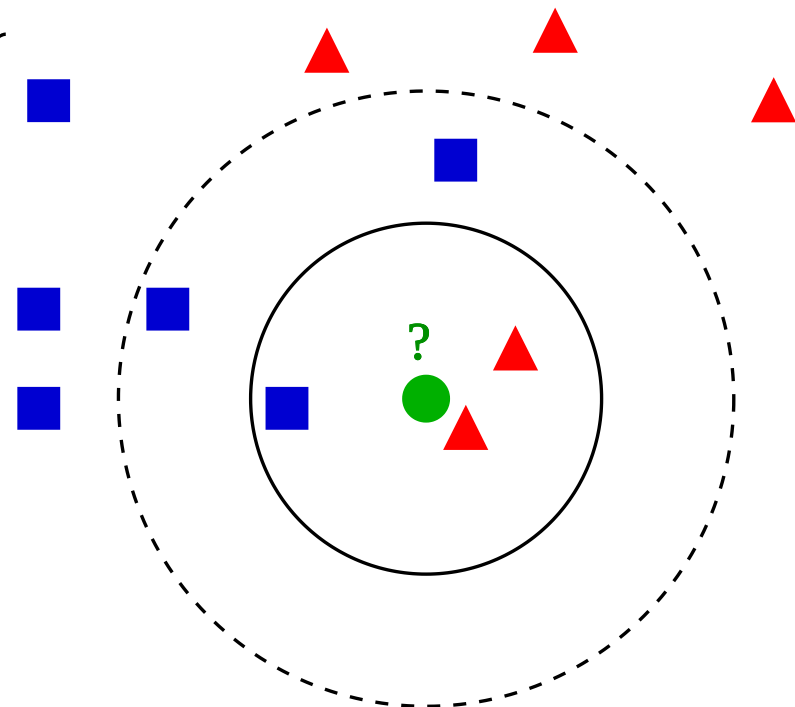
k-Nearest Neighbors

k-Nearest Neighbors

A simple but sometimes effective nonparametric method for both classification and regression is ***k*-nearest neighbors** algorithm.

The training phase of the *k*-nearest neighbors algorithm is trivial: only storing the whole train set (the so-called **lazy learning**).

For a given test example, the main idea is to use the targets of the most similar training data to perform the prediction.



<https://upload.wikimedia.org/wikipedia/commons/e/e7/KnnClassification.svg>

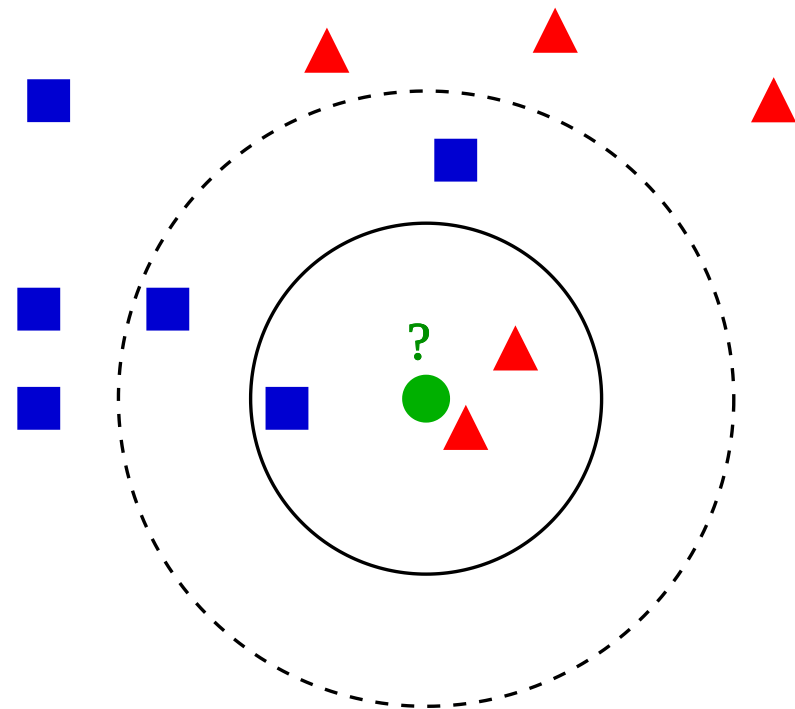
k-Nearest Neighbors: Hyperparameters

Several hyperparameters influence the behavior of the prediction phase:

- **k**: consider k most similar training examples (higher k usually decreases variance, but increases bias);
- **metric**: a function used to find the nearest neighbors; common choices are metrics based on L^p norms (usual values of p : 1, 2, 3, ∞). For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$, the distance is measured as $\|\mathbf{x} - \mathbf{y}\|_p$, where

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{1/p};$$

- **weights**: optionally, more similar examples get higher weights:
 - *uniform*: all k nearest neighbors weighted equally;
 - *inverse*: the weights are proportional to the inverse of distance;
 - *softmax*: the weights are proportional to the softmax of negative distances.



<https://upload.wikimedia.org/wikipedia/commons/e/e7/KnnClassification.svg>

Regression

To perform regression when k nearest neighbors have values t_i and weights w_i , we predict

$$t = \sum_i \frac{w_i}{\sum_j w_j} \cdot t_i.$$

Classification

For uniform weights, we can use **voting** during prediction – the most frequent class is predicted (with ties broken arbitrarily).

Otherwise, we weight the categorical distributions $\mathbf{t}_i \in \mathbb{R}^K$ (classification into K target classes represented using one-hot encoding), predicting a distribution

$$\mathbf{t} = \sum_i \frac{w_i}{\sum_j w_j} \cdot \mathbf{t}_i.$$

The predicted class is the one with the largest probability, i.e., $\arg \max_k \sum_i w_i t_{i,k}$.

A trivial implementation of the k -nearest neighbors algorithm is extremely demanding during the inference, requiring to measure distances of a given example to all training instances.

However, there exist several data structures that can speed up the k -nearest neighbor search, such as

- k - d trees, which allow both a static or dynamic construction and can perform nearest neighbor queries of uniformly random points in logarithmic time on average, but which become inefficient for high-dimensional data;
- ball trees, R-trees, ...

- Typically in combination with representation learning
- Recommendation systems (e.g., for similar videos)
- Anonymous face recognition in photo collections

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**. 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

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

1 of 10,000 products has a rare defect. We can detect it with both sensitivity and specificity of 99%. What is $P(\text{defect}|\text{positive})$?

$$P(\text{def}|\text{pos}) = \frac{\overbrace{P(\text{pos}|\text{def})P(\text{def})}^{\text{sensitivity}}}{P(\text{pos})}$$

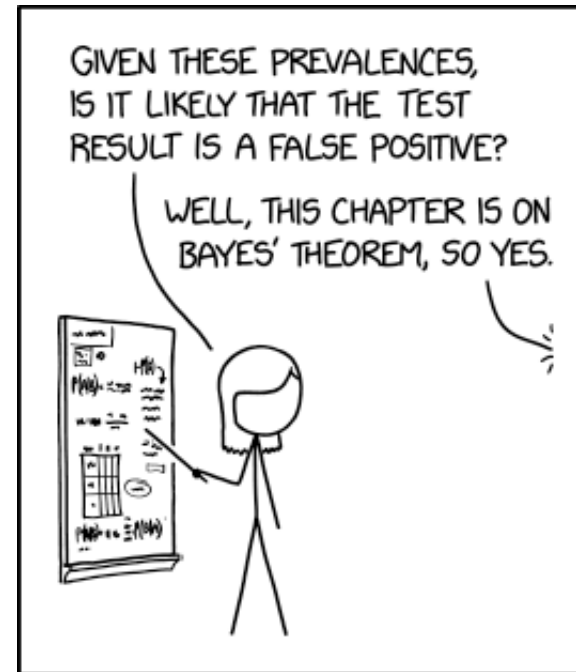
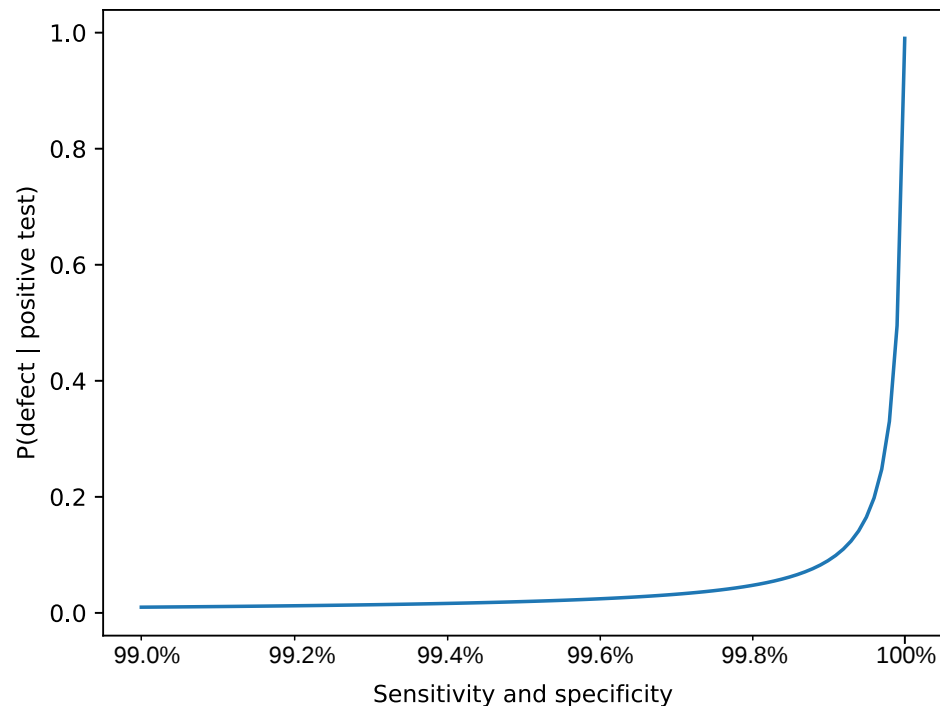
Now, we need to compute the denominator $P(\text{pos})$ by splitting into joint probabilities:

$$P(\text{pos}, \text{def}) + P(\text{pos}, \neg\text{def}) = P(\text{pos}|\text{def})P(\text{def}) + \underbrace{(1 - P(\neg\text{pos}|\neg\text{def}))}_{\text{specificity}}(1 - P(\text{def}))$$

Together

$$P(\text{def}|\text{pos}) = \frac{.99 \cdot 10^{-4}}{.99 \cdot 10^{-4} + (1 - .99)(1 - 10^{-4})} \approx 0.98\%$$

Textbook Example: How accurate do we need to be?



SOMETIMES, IF YOU UNDERSTAND
BAYES' THEOREM WELL ENOUGH,
YOU DON'T NEED IT.

https://www.explainxkd.com/wiki/index.php/2545:_Bayes%27_Theorem

Moral: seemingly high-performing classifier might not be that high-performing.

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?

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*:

$$P(\text{librarian}|\text{description}) \propto P(\text{description}|\text{librarian}) \cdot P(\text{librarian}).$$

The example is taken from the Thinking, Fast and Slow by D. Kahneman

Maximum A Posteriori Estimation

We demonstrate the Bayesian probability on model fitting.

Recall the maximum likelihood estimation

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

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

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

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

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

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

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

Bayesian view on overfitting: it is just a problem of not using priors and that suitable priors would avoid it.

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 $p(\mathbf{w}) = \prod_{i=1}^D \mathcal{N}(w_i; 0, \sigma^2) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma^2 \mathbf{I})$. Then

$$\begin{aligned}\mathbf{w}_{\text{MAP}} &= \arg \max_{\mathbf{w}} p(\mathbf{X} | \mathbf{w}) p(\mathbf{w}) \\ &= \arg \max_{\mathbf{w}} \prod_{i=1}^N p(\mathbf{x}_i | \mathbf{w}) p(\mathbf{w}) \\ &= \arg \min_{\mathbf{w}} \sum_{i=1}^N \left(-\log p(\mathbf{x}_i | \mathbf{w}) - \log p(\mathbf{w}) \right).\end{aligned}$$

By substituting the probability of the Gaussian prior, we get

$$\mathbf{w}_{\text{MAP}} = \arg \min_{\mathbf{w}} \sum_{i=1}^N \left(-\log p(\mathbf{x}_i | \mathbf{w}) + \frac{D}{2} \log(2\pi\sigma^2) + \frac{\|\mathbf{w}\|^2}{2\sigma^2} \right),$$

which is in fact the L^2 -regularization.

- In Bayesian thinking, we typically think about distribution over parameters.
- After one coin toss (Bernoulli distribution), we do not believe there is a 100% probability of what happened because we had a prior belief of how a coin behaves.
- We believed the parameter p was distributed somehow and after the observation we believe in something else (by applying the Bayes theorem).
- Conjugate distribution: prior and posterior are of the same family.
- Instead of confidence intervals, credibility intervals over the parameters.

Naive Bayes Classifier

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

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

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

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

Then, classification could be performed as

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

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

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

Naive Bayes Classifier: The Naive Assumption

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

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

all x_d are independent given C_k ,

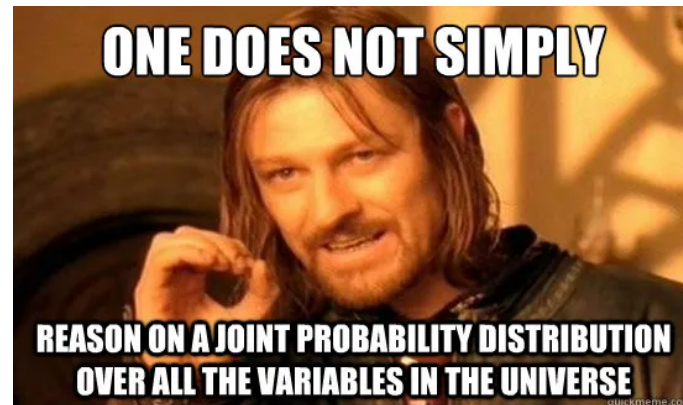
so we can rewrite

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

to

$$p(\mathbf{x}|C_k) = \prod_{d=1}^D p(x_d|C_k).$$

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



<https://medium.datadriveninvestor.com/naive-bayes-d36e57d80652>

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 $p(x_d|C_k)$ as a normal distribution $\mathcal{N}(\mu_{d,k}, \sigma_{d,k}^2)$.

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

Now let feature d and class k be fixed and let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_k}$ 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 $\mathcal{N}(\mu_{d,k}, \sigma_{d,k}^2)$ amounts to

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

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

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

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

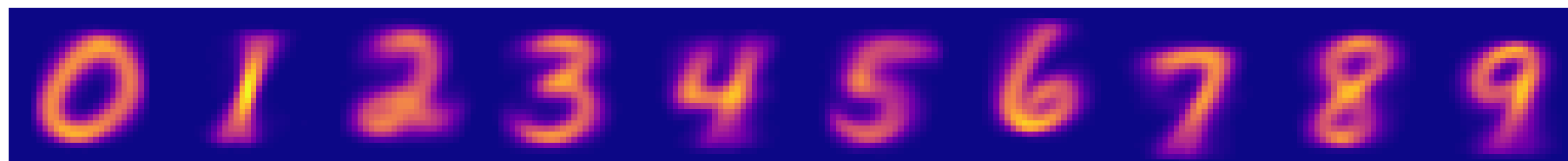
Similarly, zeroing out the derivative with respect to $\sigma_{d,k}^2$ gives

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

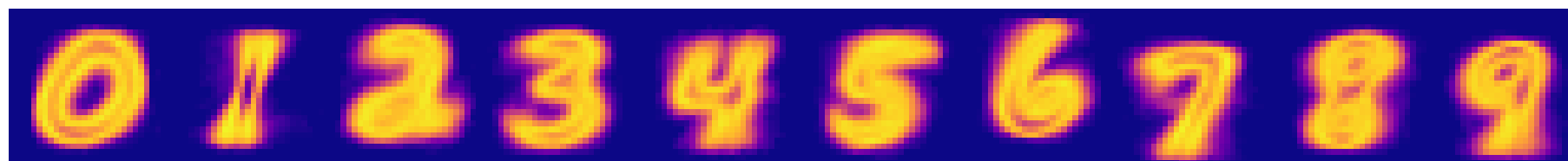
from which we obtain $\sigma_{d,k}^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} (x_{i,d} - \mu_{d,k})^2$.

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).

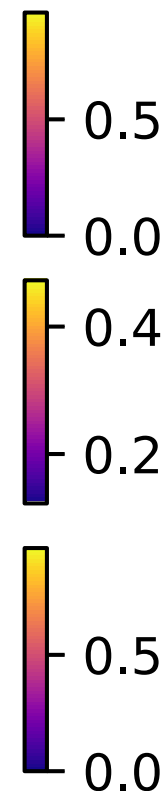
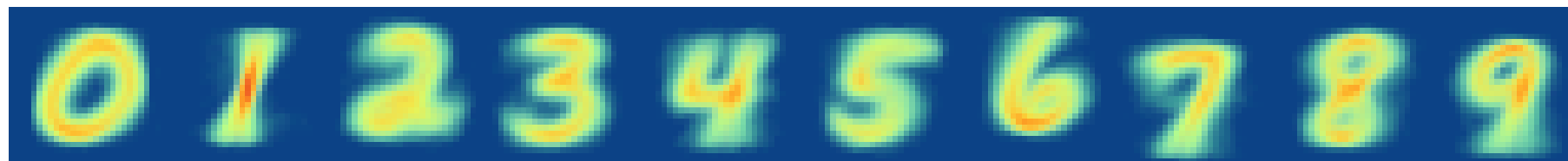
Estimated means



Estimated standard deviations



Estimated means (R+B) and stds (G)



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

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

$$p(C_k|\mathbf{x}) \propto \left(\prod_{d=1}^D p_{d,k}^{x_d} \cdot (1 - p_{d,k})^{(1-x_d)} \right) p(C_k),$$

and by computing a logarithm we get

$$\log p(C_k|\mathbf{x}) + c = \log p(C_k) + \sum_d (x_d \log \frac{p_{d,k}}{1-p_{d,k}} + \log(1 - p_{d,k})) = b_k + \mathbf{x}^T \mathbf{w}_k,$$

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

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

Bernoulli Naive Bayes Estimation: Derivation

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

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

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

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

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

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.

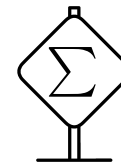
Therefore, **Laplace** or **additive smoothing** is used, and the probability $p_{d,k}$ estimated as

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

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 $\text{Beta}(\alpha + 1, \alpha + 1)$ as a prior distribution.

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

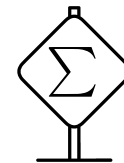


Similarly to the Bernoulli NB case, we can write the log-likelihood as

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

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



$$\sum_{i=1}^{N_k} \log \left(\prod_d p_{d,k}^{x_{i,d}} \right) = \sum_{i,d} x_{i,d} \log p_{d,k}.$$

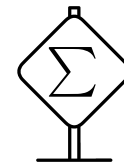
To maximize this quantity with respect to a probability distribution $\sum_d p_{d,k} = 1$, we need to form a *Lagrangian*

$$\mathcal{L} = \sum_{i,d} x_{i,d} \log p_{d,k} + \lambda \left(1 - \sum_d p_{d,k} \right).$$

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

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

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



$$p_{d,k} = \frac{n_{d,k}}{\sum_{d'=1}^D n_{d',k}}.$$

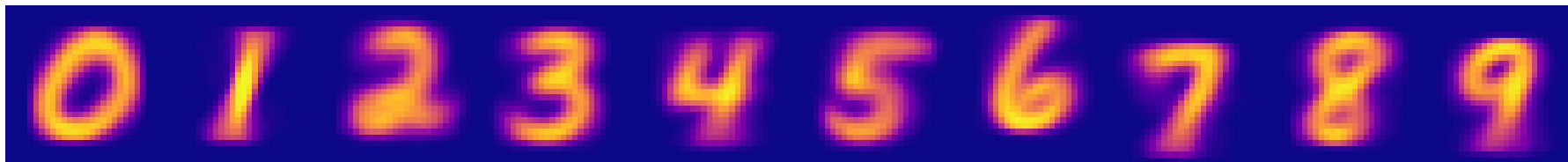
However, for the same reasons as in the Bernoulli NB case, we also use the Laplace smoothing, i.e., utilize a Dirichlet prior $\text{Dir}(\alpha + 1)$, and instead use

$$p_{d,k} = \frac{n_{d,k} + \alpha}{\sum_{d'=1}^D (n_{d',k} + \alpha)} = \frac{n_{d,k} + \alpha}{\left(\sum_{d'=1}^D n_{d',k} \right) + \alpha D}$$

with pseudo-count $\alpha > 0$.

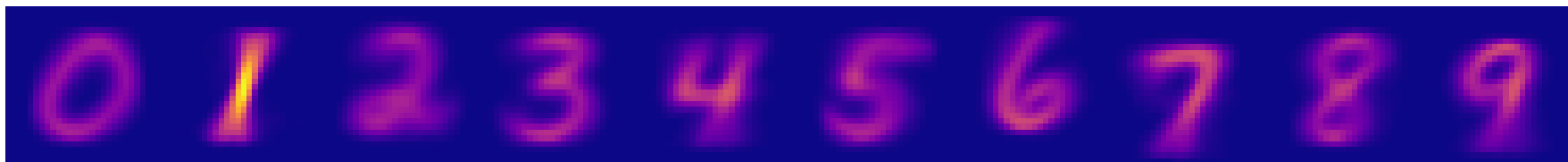
Naive Bayes Example

Estimated probabilities



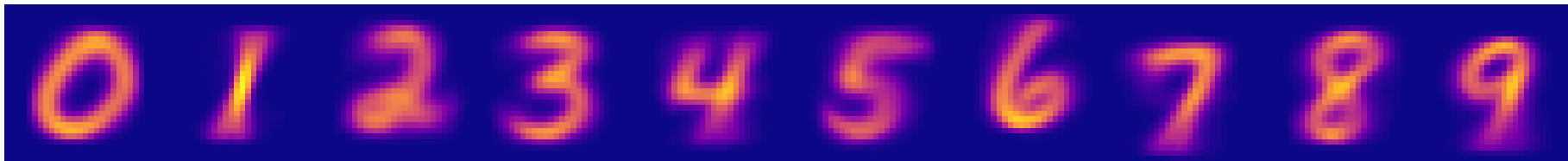
Probabilities estimated by Bernoulli NB on a subset of the MNIST dataset.

Estimated probabilities



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

Estimated means



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

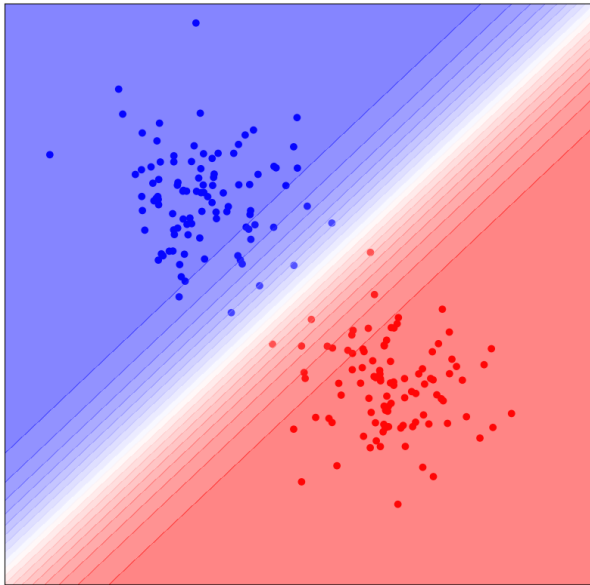
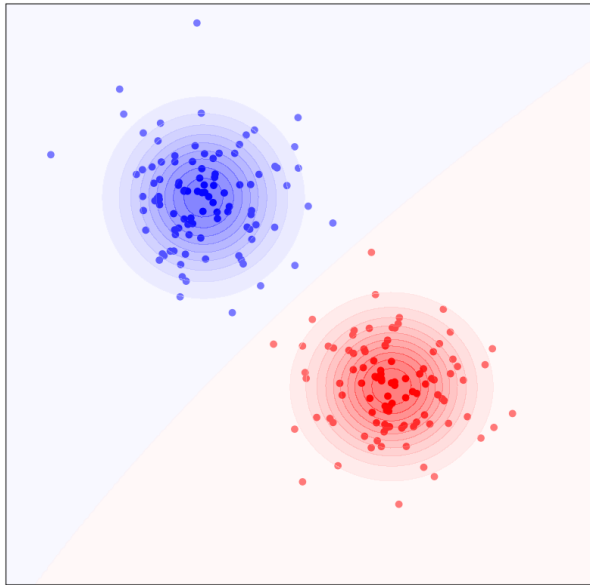
The choice among the Gaussian, Bernoulli and multinomial naive Bayes depends on the feature values.

- 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 – they must be nonnegative, be interpretable as “counts”, and “compete” with each other.
- 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 (except for the naive Bayes) have been **discriminative**, modeling a *conditional distribution* $p(t|\mathbf{x})$.

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

	Discriminative Model	Generative Model
Goal	Estimate $P(t \mathbf{x})$	Estimate $P(t, \mathbf{x}) = P(\mathbf{x} t)P(t)$
What's learned	Decision boundary	Probability distribution of the data
Illustration		

- Big topic in 2000's: Generative models are better with very little data, with enough data discriminative models are always better.
- What is now called generative (LLMs, Diffusion models) is disputable

After this lecture you should be able to

- Implement and use k -nearest neighbors for classification and regression
- Explain the very basic principles of Bayesian thinking
- Implement and use Naive Bayes Classifier