

SVR, Kernel Approximation, Naive Bayes

Milan Straka

 November 23, 2020



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



unless otherwise stated

SVM For Regression

The idea of SVM for regression is to use an ε -insensitive error function

$$\mathcal{L}_\varepsilon(t, y(\mathbf{x})) = \max(0, |y(\mathbf{x}) - t| - \varepsilon).$$

The primary formulation of the loss is then

$$C \sum_i \mathcal{L}_\varepsilon(t_i, y(\mathbf{x}_i)) + \frac{1}{2} \|\mathbf{w}\|^2.$$

In the dual formulation, we require every training example to be within ε of its target, but introduce two slack variables ξ^- , ξ^+ to allow outliers. We therefore minimize the loss

$$C \sum_i (\xi_i^- + \xi_i^+) + \frac{1}{2} \|\mathbf{w}\|^2$$

while requiring for every example $t_i - \varepsilon - \xi_i^- \leq y(\mathbf{x}_i) \leq t_i + \varepsilon + \xi_i^+$ for $\xi_i^- \geq 0, \xi_i^+ \geq 0$.

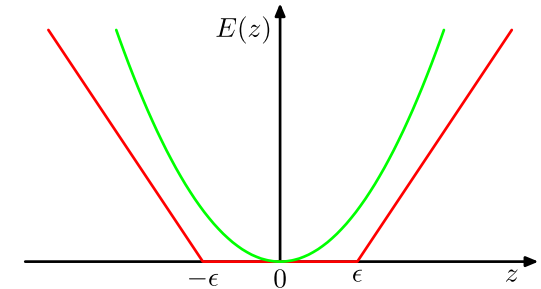


Figure 7.6 of Pattern Recognition and Machine Learning.

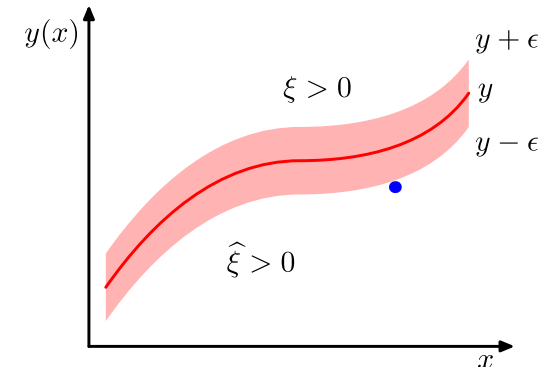


Figure 7.7 of Pattern Recognition and Machine Learning.

The Lagrangian after substituting for \mathbf{w} , b , ξ^- and ξ^+ is

$$L = \sum_i (a_i^+ - a_i^-) t_i - \varepsilon \sum_i (a_i^+ + a_i^-) - \frac{1}{2} \sum_i \sum_j (a_i^+ - a_i^-) (a_j^+ - a_j^-) K(\mathbf{x}_i, \mathbf{x}_j)$$

subject to

$$0 \leq a_i^+, a_i^- \leq C,$$

$$\sum_i (a_i^+ - a_i^-) = 0.$$

The prediction is then given by

$$y(\mathbf{z}) = \sum_i (a_i^+ - a_i^-) K(\mathbf{z}, \mathbf{x}_i) + b.$$

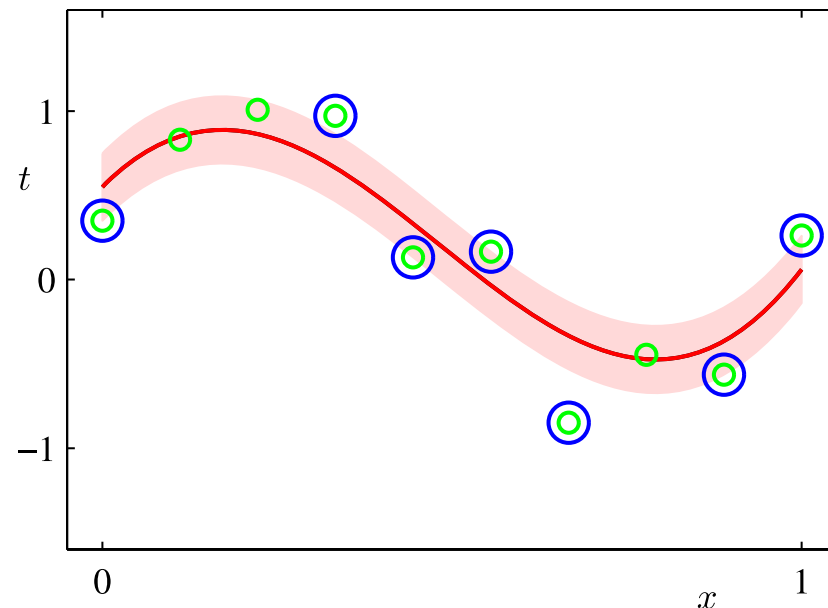


Figure 7.8 of Pattern Recognition and Machine Learning.

Using RBF Kernel in Parametric Methods

The RBF kernel empirically works well, but can be used only in the kernel methods (i.e., in the dual formulation, which is a non-parametric model), which have time complexity superlinear with the size of the training data.

Therefore, several methods have been developed to allow using an approximation of the RBF kernel in parametric models like logistic regression or MLP.

Generally, these methods define a mapping $\psi : \mathbb{R}^D \rightarrow \mathbb{R}^M$, generating M features from a given input example, such that

$$K(\mathbf{x}, \mathbf{z}) \approx \psi(\mathbf{x})^T \psi(\mathbf{z}) = \sum_m \psi_m(\mathbf{x}) \psi_m(\mathbf{z}).$$

For a given example \mathbf{x} , the features $\psi(\mathbf{x})$ are then used as input to a parametric classifier (or appended to other features we construct).

The hyperparameter M affects the quality of the approximation and is usually on the order of hundreds.

One way to approximate RBF kernel is Monte Carlo approximation of its Fourier transform.

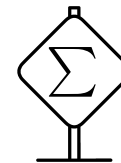
The Fourier transform of a real-valued integrable function f is

$$\hat{f}(w) \stackrel{\text{def}}{=} \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-ixw} dx,$$

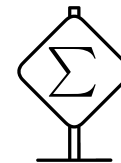
where the $\hat{f}(w)$ can be considered its *frequency spectrum*.

The transformation is invertible, and we can recover the original function as

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(w) e^{ixw} dw.$$



Now consider a shift-invariant kernel $K(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$. If we knew its frequency spectrum p , we could write it as



$$k(\mathbf{x} - \mathbf{y}) = \int_{R^D} p(\mathbf{w}) e^{i\mathbf{w}^T(\mathbf{x} - \mathbf{y})} d\mathbf{w},$$

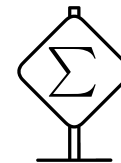
which we can rewrite using $\xi(\mathbf{x}; \mathbf{w}) = e^{i\mathbf{w}^T \mathbf{x}}$ as

$$k(\mathbf{x} - \mathbf{y}) = \int_{R^D} p(\mathbf{w}) e^{i\mathbf{w}^T(\mathbf{x} - \mathbf{y})} d\mathbf{w} = \mathbb{E}_{\mathbf{w} \sim p} [\xi(\mathbf{x}; \mathbf{w}) \xi(\mathbf{y}; \mathbf{w})^*].$$

Therefore, $\xi(\mathbf{x}; \mathbf{w}) \xi(\mathbf{y}; \mathbf{w})^*$ is an unbiased estimate of the kernel.

Random Fourier Features

However, working with the complex numbers $\xi(\mathbf{x}; \mathbf{w}) = e^{i\mathbf{w}^T \mathbf{x}}$. Nevertheless, considering that the kernel and the frequency spectrum is real-valued, it is enough just to consider the real part of $\xi(\mathbf{x}; \mathbf{w})\xi(\mathbf{y}; \mathbf{w})^*$.



Recalling Euler formula stating that $e^{i\theta} = \cos \theta + i \sin \theta$, the real part of the ξ product is $\cos(\mathbf{w}^T (\mathbf{x} - \mathbf{y}))$, and we would like to compute it from $\cos(\mathbf{w}^T \mathbf{x})$ and $\cos(\mathbf{w}^T \mathbf{y})$, which are the real parts of $\xi(\mathbf{x}; \mathbf{w})$ and $\xi(\mathbf{y}; \mathbf{w})$, respectively.

Remembering that $\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y$, we can rewrite $\cos x \cos y$ as

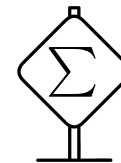
$$\cos x \cos y = \frac{1}{2} (\cos(x - y) + \cos(x + y)).$$

In order to get rid of the last term, we introduce a bias b sampled from uniform distribution $U[0, 2\pi]$ and consider mappings

$$\psi(\mathbf{x}; \mathbf{w}, b) \stackrel{\text{def}}{=} \sqrt{2} \cos(\mathbf{w}^T \mathbf{x} + b).$$

Combining the last two equations leads to

$$\begin{aligned} & \mathbb{E}_{b \sim U[0, 2\pi]} \left[\sqrt{2} \cos(\mathbf{w}^T \mathbf{x} + b) \sqrt{2} \cos(\mathbf{w}^T \mathbf{y} + b) \right] \\ &= \mathbb{E}_{b \sim U[0, 2\pi]} \left[\cos(\mathbf{w}^T \mathbf{x} - \mathbf{w}^T \mathbf{y}) + \cos(\mathbf{w}^T \mathbf{x} + \mathbf{w}^T \mathbf{y} + 2b) \right] \\ &= \cos(\mathbf{w}^T (\mathbf{x} - \mathbf{y})), \end{aligned}$$



where the last equation holds because the \cos integrate to zero with respect to b (actually, range $[0, \pi]$ would be sufficient).

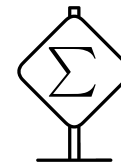
In order to decrease the variance of the estimator, we sample M values of \mathbf{w} and b and define

$$\psi_i(\mathbf{x}; \mathbf{w}_i, b_i) \stackrel{\text{def}}{=} \sqrt{2/M} \cos(\mathbf{w}_i^T \mathbf{x} + b_i).$$

Random Fourier Features

Lastly, we need the frequency spectrum of an RBF kernel.

It can be shown that for an RBF kernel with $\gamma = \frac{1}{2}$, the frequency spectrum is the density function of the standard normal distribution,



$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}; 0, 1).$$

To handle different values of γ , it is sufficient to suitably scale the input features, which we can implement by scaling the sampled \mathbf{w} . It is therefore straightforward to verify that using

$$\mathbf{w} \sim \sqrt{2\gamma} \mathcal{N}(0, 1)$$

results in an approximation of an RBF kernel with scale parameter γ .

The disadvantage of this approach is that we sample completely randomly, not taking any data into consideration.

Nyström Approximation

A different approach to approximate an RBF kernel is to use a subset of data as basis. Assume that we have a sample of M data $\mathbf{x}_1, \dots, \mathbf{x}_M$, denoting $\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$.

Our goal is to represent $K(\mathbf{x}, \mathbf{y})$ as

$$K(\mathbf{x}, \mathbf{y}) \approx \sum_{m=1}^M \psi_m(\mathbf{x}; \mathbf{v}_m) \psi_m(\mathbf{y}; \mathbf{v}_m)$$

by using linear mappings $\psi_m(\mathbf{y}; \mathbf{v}_m) = \sum_i \mathbf{v}_{m,i} K(\mathbf{y}, \mathbf{x}_i)$ with parameters $\mathbf{v}_m \in \mathbb{R}^M$.

If we denote the matrix with columns $\mathbf{v}_1, \dots, \mathbf{v}_M$ as $\mathbf{V} \in \mathbb{R}^{M \times M}$, we can write

$$\psi(\mathbf{y}; \mathbf{V}) = \mathbf{V}^T K(\mathbf{y}, \mathbf{x}_*),$$

where $K(\mathbf{y}, \mathbf{x}_*)$ is a vector of $K(\mathbf{y}, \mathbf{x}_1), \dots, K(\mathbf{y}, \mathbf{x}_M)$.

Nystrom Approximation

In order to construct our approximation, we choose \mathbf{V} such that our approximation is exact for all data $\mathbf{x}_1, \dots, \mathbf{x}_M$. Therefore, it must hold that

$$\mathbf{K}_{i,j} = (\mathbf{V}^T \mathbf{K}_i)^T \mathbf{V}^T \mathbf{K}_j.$$

We can rewrite the condition for all indices as $\mathbf{K} = \mathbf{K}^T \mathbf{V} \mathbf{V}^T \mathbf{K}$.

Therefore, we would like \mathbf{V} to be something like $\mathbf{K}^{-1/2}$.

Nyström Approximation

Because the kernel is a real symmetric matrix, it has an **eigenvalue decomposition**

$$\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{U}^T,$$

where \mathbf{U} is an orthogonal matrix and \mathbf{D} is a diagonal one.

Therefore, we can take $\mathbf{V} = \mathbf{U}\mathbf{D}^{-1/2}\mathbf{U}^T$, where

$$D_{i,i}^{-1/2} = \begin{cases} 0 & \text{if } D_{i,i} = 0, \\ D_{i,i}^{-1/2} & \text{otherwise.} \end{cases}$$

It is then straightforward to show that the required equation holds.

The overall kernel is therefore approximated by computing the kernel values of a given point and the chosen data subset, multiplied by \mathbf{V} . Empirically, the approximation works usually better than random Fourier features, because it concentrates more on the part of the space populated by the data.

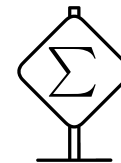
Nyström Approximation

On the previous page, we computed square roots of the diagonal matrix \mathbf{D} , which exist only if the values on the diagonal are non-negative.

However, the matrix \mathbf{K} is positive semi-definite for any kernel, and a matrix is positive semi-definite iff all its eigenvalues are non-negative.

To show that the matrix \mathbf{K} is positive semi-definite, we use the fact that the corresponding kernel is defined as a scalar product of the feature map φ :

$$\begin{aligned} \mathbf{z}^T \mathbf{K} \mathbf{z} &= \sum_{i,j} z_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) z_j \\ &= \left(\sum_i z_i \varphi(\mathbf{x}_i) \right)^T \left(\sum_j z_j \varphi(\mathbf{x}_j) \right) \\ &= \left\| \sum_i z_i \varphi(\mathbf{x}_i) \right\|^2 \\ &\geq 0. \end{aligned}$$



To represent a document, we might consider it a **bag of words**, and create a feature space with a dimension for every unique word. We can represent a word in a document as:

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

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

- **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);

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

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

Naive Bayes Classifier

Consider a discriminative classifier modelling probabilities

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

We might use Bayes' theorem and rewrite it to

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

The so-called **Naive Bayes** classifier assumes all x_i are independent given C_k , so we can write

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

as

$$p(C_k|\mathbf{x}) \propto p(C_k) \prod_i p(x_i|C_k).$$

There are several used naive Bayes classifiers, depending on the distribution $p(x_i | C_k)$.

Gaussian NB

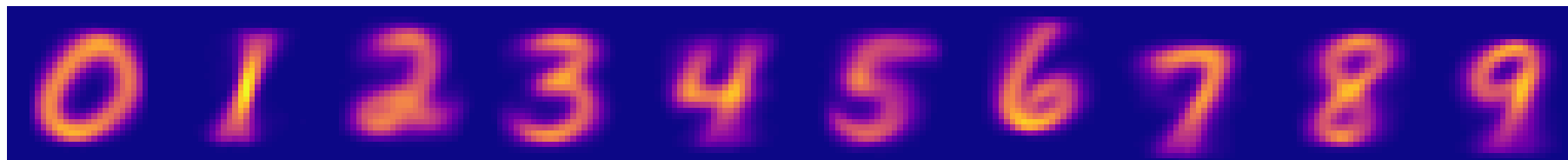
The probability $p(x_i | C_k)$ is modeled as a normal distribution $\mathcal{N}(\mu_{i,k}, \sigma_{i,k}^2)$.

The parameters $\mu_{i,k}$ and $\sigma_{i,k}^2$ are estimated directly from the data. However, the variances are usually smoothed (increased) by a given constant α to avoid too sharp distributions.

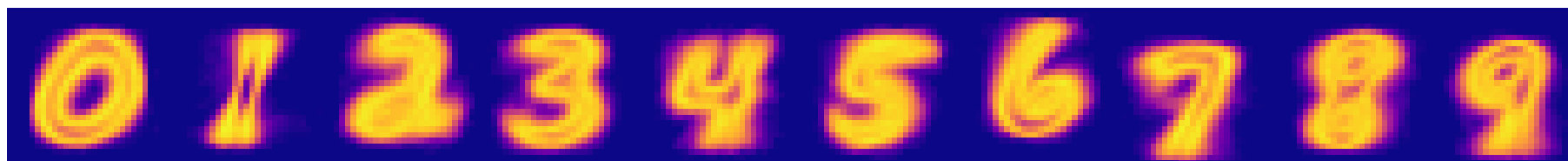
- The default value of α in Scikit-learn is 10^{-9} times the largest variance of all features.

Gaussian NB is useful if we expect a continuous feature has normal distribution for a given C_k .

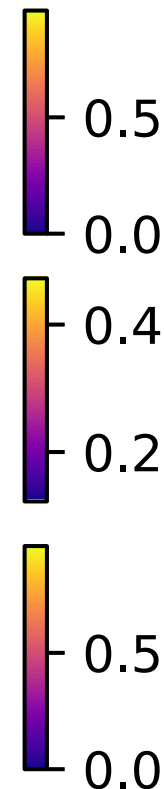
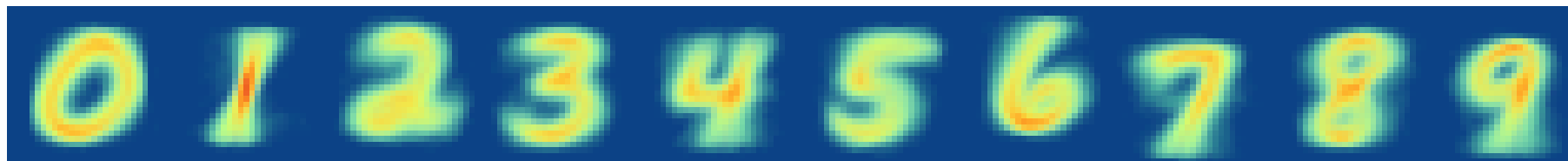
Estimated means



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.

Multinomial Distribution

We have already discussed Bernoulli distribution and categorical distribution.

The **binomial distribution** is a generalization of Bernoulli distribution, where we perform n independent binary trials, each with fixed probability of success. The binomial distribution gives the probability of a given number of successes.

It is parametrized with a success probability $p \in [0, 1]$ and a number of trials $n \in \{1, 2, \dots\}$, it is denoted as $B(n, p)$ and the probability of k successes is $\binom{n}{k} p^k (1 - p)^{n-k}$.

The **multinomial distribution** can be seen as a generalization of both the binomial and categorical distributions. Assuming we are performing n independent trials, each with k outcomes, where the outcomes have fixed probability, the multinomial distribution gives the probability of every possible combination of successes of every category.

It is parametrized with a probability distribution \mathbf{p} and a number of trials $n \in \{1, 2, \dots\}$, and the probability of x_k outcomes of category k is $\binom{n}{x_1 x_2 \dots x_k} p_1^{x_1} p_2^{x_2} \dots p_k^{x_k}$.

Both these distributions can be extended to a continuous numbers of trials and successes using the *gamma function* Γ .

Multinomial NB

When the distribution $p(\mathbf{x}|C_k)$ is multinomial, $p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{i,k}^{x_i}$, so the

$$\log p(C_k|\mathbf{x}) + c = \log p(C_k) + \sum_i \log p_{i,k}^{x_i} = \log p(C_k) + \sum_i x_i \log p_{i,k} = b + \mathbf{x}^T \mathbf{w}$$

is a linear model in the log space with $b = \log p(C_k)$ and $\mathbf{w}_i = \log p_{i,k}$. The constant c depends on \mathbf{x} and its value is not needed neither for estimation nor prediction.

Denoting $n_{i,k}$ as the sum of features x_i for a class C_k , the probabilities $p_{i,k}$ are usually estimated as

$$p_{i,k} = \frac{n_{i,k} + \alpha}{\sum_j n_{j,k} + \alpha D}$$

where α is a *smoothing* parameter accounting for terms not appearing in any document of class C_k (we can view it as a *pseudocount* given to every term in every document).

Bernoulli NB

When the input features are binary, the $p(x_i|C_k)$ might also be a Bernoulli distribution

$$p(x_i|C_k) = p_{i,k}^{x_i} \cdot (1 - p_{i,k})^{(1-x_i)},$$

and as in the Multinomial NB case, we can write

$$\log p(C_k|\mathbf{x}) + c = \log p(C_k) + \sum_i (x_i \log \frac{p_{i,k}}{1-p_{i,k}} + \log(1 - p_{i,k})) = b + \mathbf{x}^T \mathbf{w}.$$

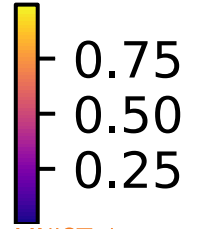
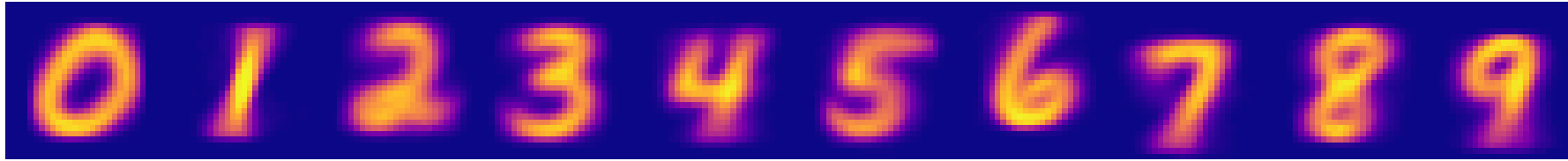
Similarly to the Multinomial NB, the probabilities are usually estimated as

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

The difference with respect to Multinomial NB is that Bernoulli NB explicitly models also the *absence of terms* by $(1 - p_{i,k})$, while $p_{i,k}^0 = 1$ is used in Multinomial NB. However, the cost is that the input features must be binary (so for example TF-IDF cannot be used).

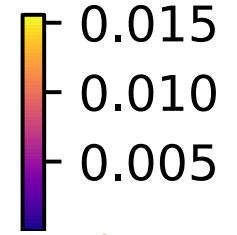
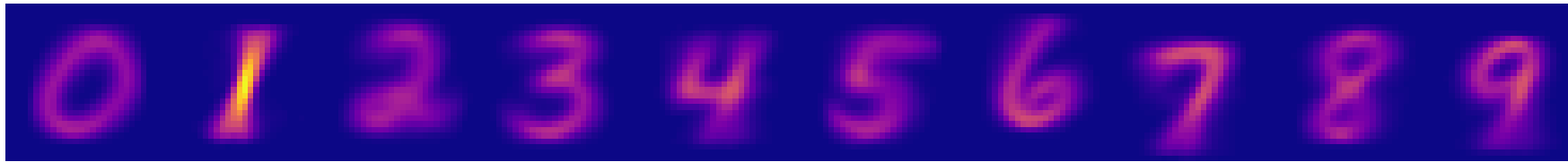
Naive Bayes Example

Estimated probabilities



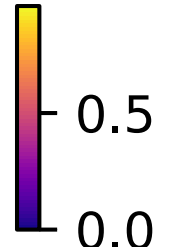
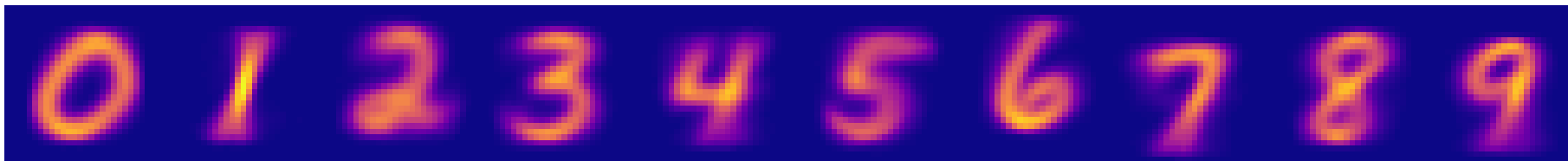
Probabilities estimated by Bernoulli NB on a subset of the MNIST dataset.

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.

Naive Bayes Classifier as a Generative Model

Given that a Multinomial/Bernoulli NB fits $\log p(C_k, \mathbf{x})$ as a linear model and a logistic regression also fits $\log p(C_k | \mathbf{x})$ as a linear model, naive Bayes and logistic regression form a so-called **generative-discriminative** pair, where the naive Bayes is a **generative** model, while logistic regression is a **discriminative** model.

Generative and Discriminative Models

So far, most of our models have been **discriminative**, modeling a *conditional distribution* $p(\mathbf{t}|\mathbf{x})$ (predicting some output distribution). Empirically, such models usually perform better in classification tasks, but because they do not estimate the probability of \mathbf{x} , it might be difficult for them to recognize outliers (out-of-distribution data).

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

The term generative comes from a (theoretical) possibility of “generating” random instances (either of (\mathbf{x}, \mathbf{t}) or \mathbf{x} given \mathbf{t}). However, just being able to evaluate $p(\mathbf{x}|\mathbf{t})$ does not necessarily mean there must be an efficient procedure of actually sampling (generating) \mathbf{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 $p(\mathbf{x})$.

Maximum A Posteriori Estimation

We already discussed maximum likelihood estimation

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

Instead, we may want to maximize *maximum a posteriori (MAP)* point estimate:

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

Using Bayes' theorem

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

we get

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

L2 Regularization as MAP

Another way to arrive at L2 regularization is to employ the MAP estimation and assume that the prior probabilities $p(\mathbf{w})$ of the parameter values (our *preference* among the models) is $\mathcal{N}(\mathbf{w}; 0, \sigma^2)$.

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

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