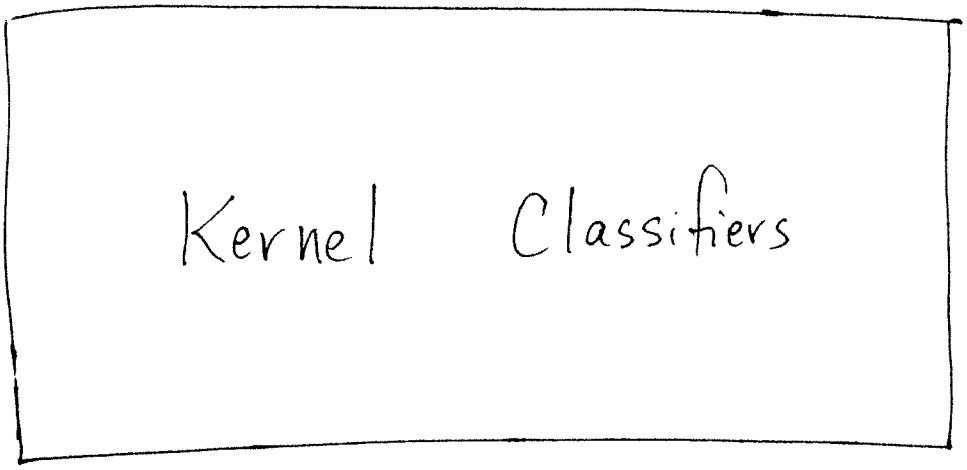

Kernel Classifiers

Kernel Classifiers

A kernel classifier is a plug-in classifier obtained using a kernel density estimate for each class-conditional density.

Kernel Density Estimation

Let $x_1, \ldots, x_n$ be a random sample from a distribution with density $g(x)$. A kernel density estimate of g is a nonparametric estimate given by

$$\hat{g}(x) = \frac{1}{n} \sum_{i=1}^{n} k(x - x_i)$$

where $k(x)$ is a kernel function.

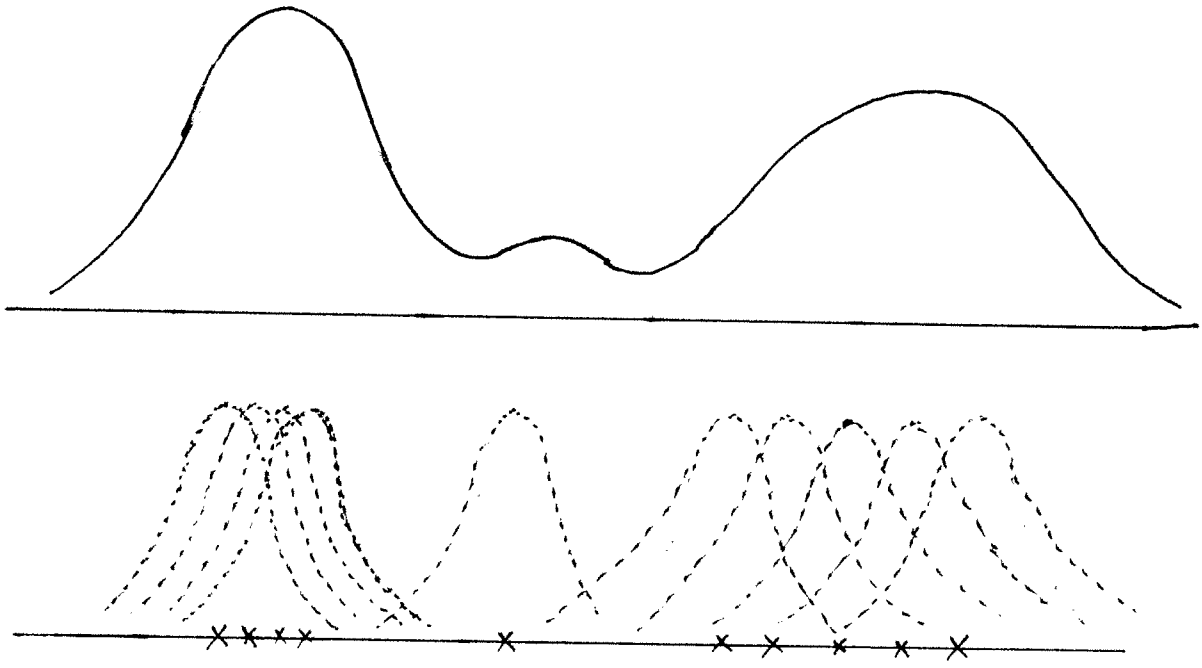
This estimate is also called the Parzen window estimate.

A kernel function should satisfy

1. $\int k(x)\,dx = 1$
2. $k(x) \geq 0$
3. $k(-x) = k(x)$

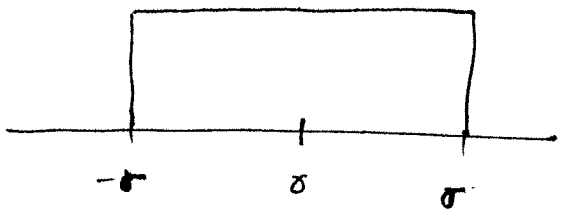
Primary example | Gaussian kernel with spherical cov.

$$k(x) = k_\sigma(x) = (2\pi\sigma^2)^{-\frac{d}{2}} \exp\left\{-\frac{\|x\|^2}{2\sigma^2}\right\}$$

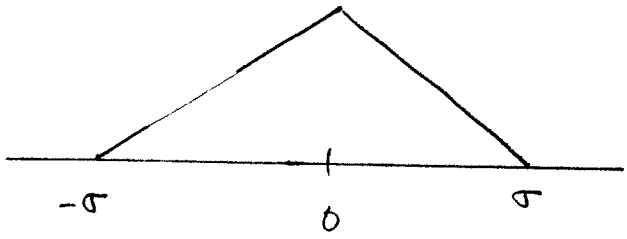


Other examples

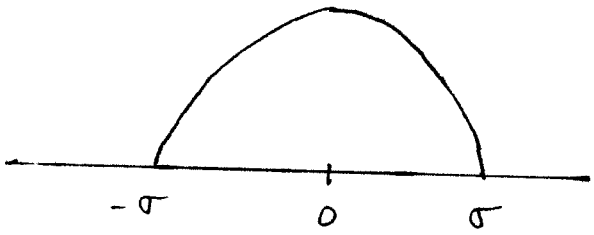
Uniform kernel: $k_\sigma(x) = c_\sigma I_{\{\|x\| \le \sigma\}}$



Triangular kernel:



Epanechnikov kernel (parabolic)



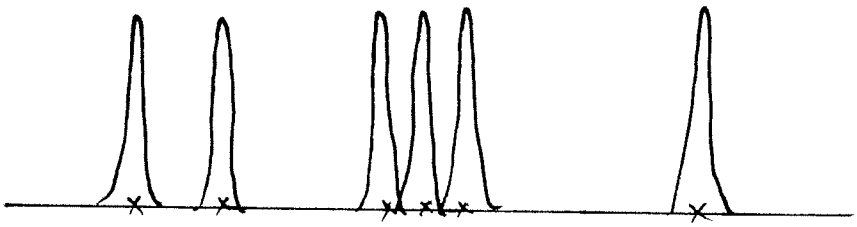
Thus, a KDE requires that we specify

1. a kernel
2. the <u>width</u> σ of the kernel.

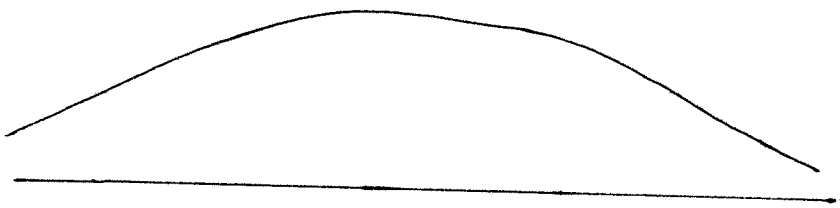
A great deal of research has been conducted on how to answer these questions. The consensus is that

- the kernel is not that critical
- the width σ is critical.

<u>σ too small</u>



<u>σ too large</u>



Bandwidth Selection: Two Approaches

1. Estimate the probability of error of the kernel classifier as a function of σ, and choose the σ that minimizes this estimate.

2. Estimate the quality of $\hat{g} = \hat{g}_\sigma$ as a density estimate and choose σ to optimize the quality.

Bandwidth Selection by Error Estimation

Recall: The probability of error of a classifier f is

$$R(f) = P(f(X) \neq Y).$$

Let $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training dataset.

Let $\mathcal{A}_n^\sigma : D_n \longmapsto \hat{f}_n^\sigma$ be the kernel discrimination rule / classifier based on a (fixed) kernel having bandwidth σ.

How should we estimate $R(\hat{f}_n^\sigma)$?

Training error estimate

$$\hat{R}\left(\hat{f}_n^{\sigma}\right) = \frac{1}{n}\sum_{i=1}^{n} I_{\left\{\hat{f}_n^{\sigma}(x_i) \neq y_i\right\}}$$

where

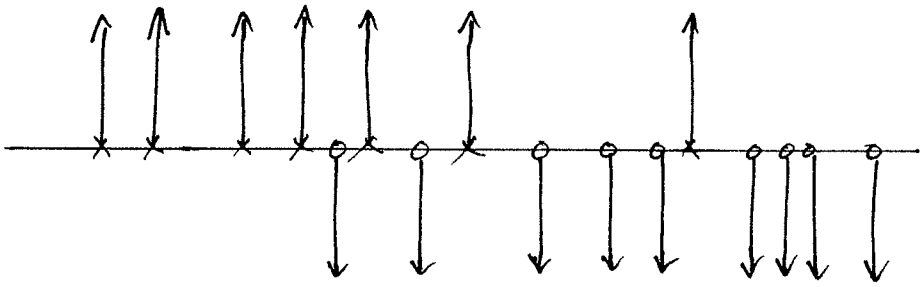
$$I_{\{p\}} = \begin{cases} 1 & \text{if } p \text{ is true} \\ 0 & \text{if } p \text{ is false} \end{cases}$$

How would we choose σ to minimize this error estimate?

Write the plug-in classifier as

$$\hat{\pi}_1 \hat{g}_1^{\sigma}(x) - \hat{\pi}_0 \hat{g}_0^{\sigma}(x) \underset{0}{\overset{1}{\gtrless}} 0$$

Letting $\sigma \to 0$ we get this picture:



That is, $\hat{R}(\hat{f}_n^{\sigma}) = 0$ for σ sufficiently small

$\Rightarrow$ <u>overfitting</u>

The training error estimate is also called the <u>apparent</u> error rate or <u>resubstitution</u> error estimate.

Holdout Error Estimate

Partition the training data into two batches

$$D_n \begin{cases} \nearrow D_m = \{(x_1, y_1), \ldots, (x_m, y_m)\} \\ \searrow D_n \setminus D_m = \{(x_{m+1}, y_{m+1}), \ldots, (x_n, y_n)\} \end{cases}$$

Basic Idea: Run f_m^σ on D_m and evaluate $\hat{f}_m^\sigma$ on $D_n \setminus D_m$.

Formally, the holdout error estimate is

$$\hat{R}_{HO}(\hat{f}_m^\sigma) = \frac{1}{n-m} \sum_{i=m+1}^{n} I_{\{\hat{f}_m^\sigma(x_i) \neq y_i\}}$$

What is the expected value of $\hat{R}_{HO}(\hat{f}_m^\sigma)$?

$$E\left\{\hat{R}_{HO}\left(\hat{f}_m^{\sigma}\right)\right\} = \frac{1}{n-m}\sum_{i=m+1}^{n} E\left\{I_{\left\{\hat{f}_m^{\sigma}(X_i)\neq Y_i\right\}}\right\}$$

since D_m, $D_n \setminus D_m$ are independent →

$$= \frac{1}{n-m}\sum_{i=m+1}^{n} P\left(\hat{f}_m^{\sigma}(X_i)\neq Y_i\right)$$

$$= \frac{1}{n-m}\sum_{i=m+1}^{n} R\left(\hat{f}_m^{\sigma}\right)$$

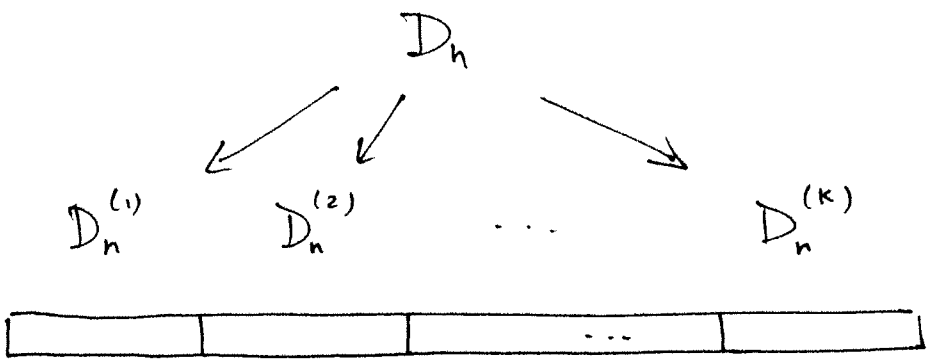
$$= R\left(\hat{f}_m^{\sigma}\right)$$

$\Rightarrow$ $\hat{R}_{HO}$ is an unbiased error estimate.

Unfortunately, by holding out some of our data, our classifier is based on a fraction of the available data, which is undesirable, especially if n is small.

Cross-Validation

Partion D_n into K equally sized groups



K-fold Cross-Validation

1. For each $k = 1, \ldots, K$, hold out $D_n^{(k)}$, run $\mathcal{A}^\sigma$ on $D_n \setminus D_n^{(k)}$ to get $\hat{f}_{(k)}^\sigma$.

2. Estimate the error probability of $\hat{f}_{(k)}^\sigma$:

$$\hat{R}^{(k)}\left(\hat{f}_{(k)}^\sigma\right) = \frac{1}{|D_n^{(k)}|} \sum_{(x_i, y_i) \in D_n^{(k)}} I\left\{\hat{f}_{(k)}^\sigma(x_i) \neq y_i\right\}$$

3. Average:

$$\hat{R}_{cv}\left(\hat{f}_n^\sigma\right) = \frac{1}{K} \sum_{k=1}^{K} \hat{R}^{(k)}\left(\hat{f}_{(k)}^\sigma\right)$$

The most popular choice is $K = n$, which is called leave-one-out cross validation (LOOCV).

How to choose K?

- Large K $\Rightarrow$ low bias, high variance, high computational cost
- Small K $\Rightarrow$ higher bias, lower variance

$K = 5, 10, n$ are the most common choices.

For small datasets (small n), $K = 5$ can lead to unacceptably small training sets $D_n \setminus D_n^{(k)}$.

Ten Dollar Question

Why is the bias not 0?

Bandwidth Selection by Least-Squares Cross-Validation

Let us measure the performance of the KDE $\hat{g}_\sigma$ using the integrated squared error:

$$ISE(\sigma) = \int \left(\hat{g}_\sigma(x) - g(x) \right)^2 dx$$

Idea: Choose σ to minimize this expression

But it depends on g, which is unknown.

What can we do?

Observe:

$$ISE(\sigma) = \int \hat{g}_\sigma(x)^2 dx - 2 \int g(x)\, \hat{g}_\sigma(x)\, dx + \int g(x)^2 dx$$

$\int g(x)\, \hat{g}_\sigma(x)\, dx$: expectation w.r.t. $g(x)$

$\int g(x)^2 dx$: independent of σ

We can approximate

$$\int g(x)\, \hat{g}_\sigma(x)\, dx = E_{g(x)}\left\{ \hat{g}_\sigma(X) \right\}$$

$$\approx \frac{1}{n} \sum_{i=1}^{n} \hat{g}_\sigma(x_i)$$

true mean

↑ LLN

sample mean

This suggests that to select σ, we should minimize

$$\int \hat{g}_\sigma(x)^2 dx - \frac{2}{n} \sum_{i=1}^{n} \hat{g}_\sigma(x_i)$$

However, by using the sample mean we are prone to <u>overfitting</u>. Instead, we can employ a cross-validation estimate of $g(x_i)$. Define

$$\hat{g}_\sigma^{-i}(x) = \frac{1}{n-1} \sum_{\substack{j=1 \\ j \neq i}}^{n} K_\sigma(x - x_j)$$

Then $\hat{g}_\sigma^{-i}(x_i)$ is a LOOCV estimate of $g(x_i)$

Exercise | It can be shown that

$$\int K_\sigma(x - x_i) \cdot K_\sigma(x - x_j)\, dx = K_{\sqrt{2}\sigma}(x_i - x_j).$$

Use this fact to derive a computable expression for

$$LSCV(\sigma) := \int \hat{g}_\sigma(x)^2\, dx - \frac{2}{n} \sum_{i=1}^{n} \hat{g}_\sigma^{-i}(x_i)$$

least-squares cross-validation

Solution

$$LSCV(\sigma) = \int \left(\frac{1}{n}\sum_{i=1}^{n} K_\sigma(x-x_i)\right)\left(\frac{1}{n}\sum_{j=1}^{n} K_\sigma(x-x_j)\right) dx$$

$$- \frac{2}{n}\sum_{i=1}^{n} \frac{1}{n-1}\sum_{j\neq i} K_\sigma(x_i-x_j)$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \int K_\sigma(x-x_i)\,K_\sigma(x-x_j)\,dx$$

$$- \frac{2}{n(n-1)} \sum_{\substack{i\neq j \\ 1\le i,j\le n}} K_\sigma(x_i-x_j)$$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} K_{\sqrt{2}\sigma}(x_i-x_j)$$

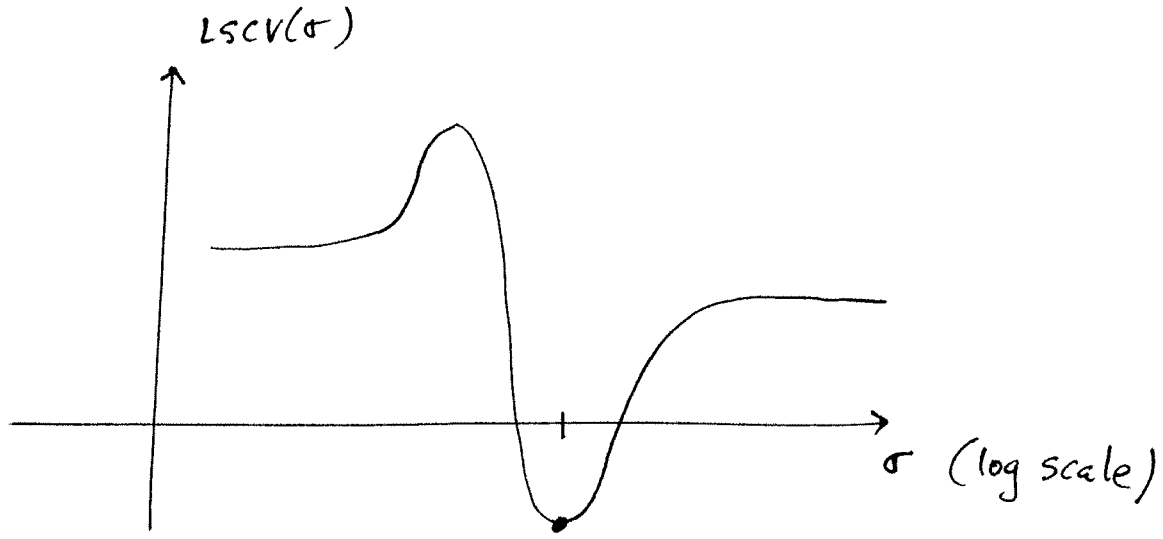
$$- \frac{2}{n(n-1)} \sum_{\substack{1\le i,j\le n \\ i\neq j}} K_\sigma(x_i-x_j)$$

$$= \frac{1}{n^2}\cdot\sum_{i=1}^{n} K_{\sqrt{2}\sigma}(0)$$

$$+ \frac{1}{n}\sum_{i\neq j}\left(\frac{1}{n}K_{\sqrt{2}\sigma}(x_i-x_j) - \frac{2}{n-1}K_\sigma(x_i-x_j)\right)$$

$$= \boxed{\frac{(4\pi\sigma^2)^{-\frac{d}{2}}}{n} + \frac{1}{n}\sum_{i\neq j}\left(\frac{1}{n}K_{\sqrt{2}\sigma}(x_i-x_j) - \frac{2}{n-1}K_\sigma(x_i-x_j)\right)}$$

In practice, one evaluates $LSCV(\sigma)$ on a predetermined grid of points and selects the minimizer



The graph is usually smooth with a clearly visible global minimizer. Local minima may also be present, however.

Naive Bayes Classifier

The naive Bayes classifier makes the following unrealistic assumption:

The density $g(x)$ $(= g_0(x)$ or $g_1(x))$ for either class factors as

$$g(x) = \prod_{i=1}^{d} g_i(x^{(i)})$$

Where $x^{(i)}$ is the i^{th} coordinate of x.
In other words, the coordinates of $X \mid Y=y$ are independent.

Under this assumption, g can be estimated by estimating each coordinate's density separately:

For each $i = 1, \ldots, d$, use $x_1^{(i)}, \ldots, x_n^{(i)}$ to estimate $g_i(x)$ using a kernel density estimator.

The final plug-in classifier is

$$\frac{\hat{g}_1(x)}{g_0(x)} = \frac{\prod_{i=1}^{d} \hat{g}_{1,i}(x^{(i)})}{\prod_{i=1}^{d} \hat{g}_{0,i}(x^{(i)})} \underset{0}{\overset{1}{\gtrless}} \frac{\hat{\pi}_0}{\hat{\pi}_1}.$$

The obvious advantage of the Naïve Bayes classifier is its simplicity: It is much easier to estimate a univariate density than a multivariate density.

Does this tradeoff outweigh the clearly incorrect modeling assumption? In many cases, yes. Keep in mind that in classification, the ultimate goal is not to learn the densities g_0, g_1, but to learn the set $\{x : f^*(x) = 1\}$. In many cases the deficiencies of $\hat{g}_0, \hat{g}_1$ occur away from the decision boundary and therefore do not influence the classifier.