Can we kernelize regression?

In order to “kernelize” an algorithm, the general approach consists of three main steps:

1. Show that the training process only involves the training data via inner products (i.e., $\mathbf{x}_i^T \mathbf{x}_j$)

2. Show that applying the decision rule to a new $\mathbf{x}$ only involves computing inner products (i.e., $\mathbf{w}^T \mathbf{x}$)

3. Replace all inner products with evaluations of the kernel function $k(\cdot, \cdot)$

This approach extends well beyond SVMs

Ridge regression revisited

Ridge regression: Given $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$

$$(\widehat{\boldsymbol{\beta}}, \widehat{\beta}_0) = \arg\min_{\boldsymbol{\beta}, \beta_0} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0)^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

Solution: $\dfrac{\partial}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^T \mathbf{x}_i - \beta_0) = 0$

$$\widehat{\beta}_0 = \frac{1}{n} \sum_{i=1}^{n} y_i - \widehat{\boldsymbol{\beta}}^T \mathbf{x}_i \qquad \bar{y} = \frac{1}{n} \sum_i y_i$$

$$= \bar{y} - \widehat{\boldsymbol{\beta}}^T \bar{\mathbf{x}} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_i \mathbf{x}_i$$

Ridge regression revisited

Plugging this back in we are left to minimize

$$\sum_{i=1}^{n}(y_i - \bar{y} - \boldsymbol{\beta}^T(\mathbf{x}_i - \bar{\mathbf{x}}))^2 + \lambda\|\boldsymbol{\beta}\|_2^2$$

with respect to $\boldsymbol{\beta}$

$$\widehat{\boldsymbol{\beta}} = \left(\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{X}}^T\widetilde{\mathbf{y}}$$

$$\widetilde{\mathbf{X}} = \begin{bmatrix}(\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T\end{bmatrix} \qquad \widetilde{\mathbf{y}} = \begin{bmatrix}y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y}\end{bmatrix}$$

Kernel ridge regression?

$$\widehat{\boldsymbol{\beta}} = \left(\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{X}}^T\widetilde{\mathbf{y}}$$

$$\widetilde{\mathbf{X}} = \begin{bmatrix}(\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T\end{bmatrix} \qquad \widetilde{\mathbf{y}} = \begin{bmatrix}y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y}\end{bmatrix}$$

$$\widehat{f}(\mathbf{x}) = \bar{y} + \widehat{\boldsymbol{\beta}}^T(\mathbf{x} - \bar{\mathbf{x}})$$

Can we express ridge regression in terms of inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ and $\langle \mathbf{x}_i, \mathbf{x} \rangle$?

Not immediately. $\left[\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}}\right](i,j) \neq \widetilde{\mathbf{x}}_i^T\widetilde{\mathbf{x}}_j$

Useful fact

One way to express the solution to the regularized least squares problem is via

$$\widehat{\boldsymbol{\beta}} = \left(\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{X}}^T\widetilde{\mathbf{y}}$$

It is not hard to show (you may do it on a future homework!) that, in fact

$$\left(\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{X}}^T\widetilde{\mathbf{y}} = \widetilde{\mathbf{X}}^T\left(\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{y}}$$

We can just as well take $\widehat{\boldsymbol{\beta}} = \widetilde{\mathbf{X}}^T\left(\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T + \lambda\mathbf{I}\right)^{-1}\widetilde{\mathbf{y}}$

Kernel ridge regression? (v2)

$$\widehat{\boldsymbol{\beta}} = \widetilde{\mathbf{X}}^T \left(\widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^T + \lambda \mathbf{I} \right)^{-1} \widetilde{\mathbf{y}}$$

$$\widetilde{\mathbf{X}} = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix} \qquad \widetilde{\mathbf{y}} = \begin{bmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{bmatrix}$$

$$\widehat{f}(\mathbf{x}) = \bar{y} + \widehat{\boldsymbol{\beta}}^T (\mathbf{x} - \bar{\mathbf{x}})$$

If we set $K(i,j) = (\mathbf{x}_i - \bar{\mathbf{x}})^T (\mathbf{x}_j - \bar{\mathbf{x}})$, then $\mathbf{K} = \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^T$

$$\widehat{\boldsymbol{\beta}} = \widetilde{\mathbf{X}}^T \left(\mathbf{K} + \lambda \mathbf{I} \right)^{-1} \widetilde{\mathbf{y}}$$

Kernel ridge regression? (v3)

A naïve implementation: compute $\widehat{\boldsymbol{\beta}} = \widetilde{\mathbf{X}}^T (\mathbf{K} + \lambda \mathbf{I})^{-1} \widetilde{\mathbf{y}}$, then apply

$$\widehat{f}(\mathbf{x}) = \bar{y} + \widehat{\boldsymbol{\beta}}^T (\mathbf{x} - \bar{\mathbf{x}})$$

Instead, note that

$$\begin{aligned}
\widehat{f}(\mathbf{x}) &= \bar{y} + \widehat{\boldsymbol{\beta}}^T (\mathbf{x} - \bar{\mathbf{x}}) \\
&= \bar{y} + \left(\widetilde{\mathbf{X}}^T (\mathbf{K} + \lambda \mathbf{I})^{-1} \widetilde{\mathbf{y}} \right)^T (\mathbf{x} - \bar{\mathbf{x}}) \\
&= \bar{y} + \widetilde{\mathbf{y}}^T (\mathbf{K} + \lambda \mathbf{I})^{-1} \widetilde{\mathbf{X}}^T (\mathbf{x} - \bar{\mathbf{x}})
\end{aligned}$$

Kernel ridge regression!

We now have a fully “kernelizable” algorithm

$$\widehat{f}(\mathbf{x}) = \bar{y} + \widetilde{\mathbf{y}}^T \left(\mathbf{K} + \lambda \mathbf{I}\right)^{-1} \mathbf{k}(\mathbf{x})$$

$$\mathbf{k}(\mathbf{x}) = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) \end{bmatrix}$$

Given any $\mathbf{x}$, we can compute $\widehat{f}(\mathbf{x})$ using only inner products!

Note: $\mathbf{K}$ and $\mathbf{k}(\mathbf{x})$ can be computed without explicitly computing $\bar{\mathbf{x}}$ by simply writing $\bar{\mathbf{x}}$ as a sum and using linearity

Homogenous kernel ridge regression

For many kernels, $\Phi(\mathbf{x})$ already contains a constant component, in which case we often omit β_0

- inhomogenous polynomial kernel
- Gaussian kernel does not seem to require a constant

In this case, the kernel ridge regression solution becomes

$$\widehat{f}(\mathbf{x}) = \mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$$

where $\mathbf{y} = \left[y_1, \cdots, y_n\right]^T$

$$\mathbf{k}(\mathbf{x}) = \left[k(\mathbf{x}_1, \mathbf{x}), \cdots, k(\mathbf{x}_n, \mathbf{x})\right]^T$$

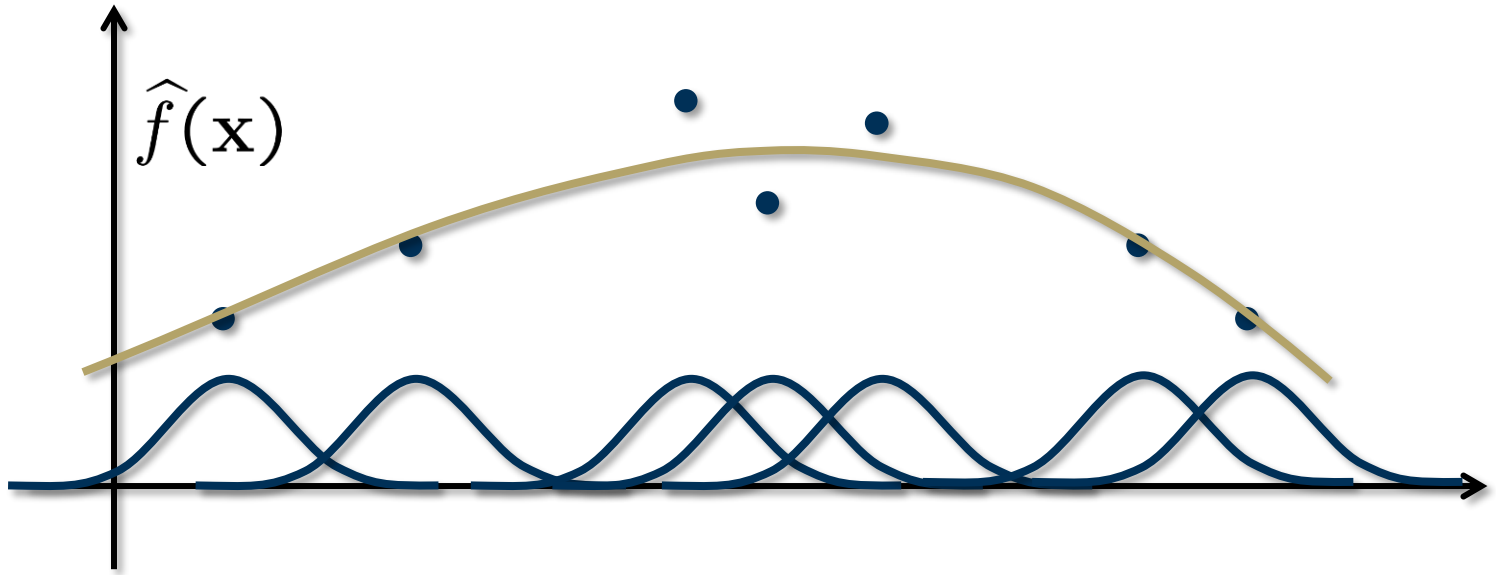
$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Example: Gaussian kernel

$$k(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u} - \mathbf{v}\|_2^2}{2\sigma^2}\right)$$

If we omit β_0, then $\widehat{f}(\mathbf{x}) = \mathbf{y}^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$

$$= \boldsymbol{\alpha}^T\mathbf{k}(\mathbf{x}) = \sum_{i=1}^{n} \alpha(i)k(\mathbf{x}, \mathbf{x}_i)$$



Beyond ridge regression

Recall that ridge regression can be viewed as a particular instance of the following general approach to regression

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \; L(\boldsymbol{\theta}) + \lambda r(\boldsymbol{\theta})$$

- $L(\boldsymbol{\theta})$ is a ***loss function***, enforces data fidelity

$$f_{\boldsymbol{\theta}}(\mathbf{x}_i) \approx y_i$$

- $r(\boldsymbol{\theta})$ is a ***regularizer*** which serves to quantify the "complexity" of $\boldsymbol{\theta}$

Can we repeat the same kernelization approach with other regularizers or loss functions?

Kernelized LASSO?

Can we kernelize this?

$$\widehat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1$$

Not exactly…

Nevertheless, we can impose (with no justification) that

$$\widehat{\boldsymbol{\theta}} = \sum_i \alpha_i \mathbf{x}_i$$

and then replace $\|\boldsymbol{\theta}\|_1$ with $\|\boldsymbol{\alpha}\|_1$

This yields an algorithm that can be easily kernelized, although it is really something different than the LASSO

- promotes sparsity in $\boldsymbol{\alpha}$, not $\boldsymbol{\theta}$

Robust regression

What about alternative loss functions?

Mean absolute error

$$L_{AE}(r) = |r|$$

Huber loss

$$L_H(r) = \begin{cases} \frac{1}{2}r^2 & \text{if } |r| \leq c \\ c|r| - \frac{c^2}{2} & \text{if } |r| > c \end{cases}$$

ϵ-insensitive loss

$$L_\epsilon(r) = \begin{cases} 0 & \text{if } |r| \leq \epsilon \\ |r| - \epsilon & \text{if } |r| > \epsilon \end{cases}$$

Regularized robust regression

Suppose we combine this loss with an ℓ_2 regularizer

$$\widehat{\boldsymbol{\beta}}, \beta_0 = \underset{(\boldsymbol{\beta}, \beta_0)}{\arg\min} \sum_{i=1}^{n} L_\epsilon(y_i - (\boldsymbol{\beta}^T \mathbf{x}_i + \beta_0)) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2$$

Note that the ϵ -insensitive loss has no penalty as long as your prediction is within a "margin" of ϵ

This looks like an SVM…

Support vector regression

The previous problem can also be cast in the following dual form

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \sum_i ((\epsilon - y_i)\alpha_i^* + (\epsilon + y_i)\alpha_i) + \frac{1}{2}\sum_{i,j}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathbf{x}_j^T\mathbf{x}_i$$

$$\text{subject to} \quad 0 \le \alpha_i^*, \alpha_i \le \tfrac{1}{\lambda}$$

$$\sum_i (\alpha_i^* - \alpha_i) = 0$$

$$\alpha_i^*\alpha_i = 0$$

The solution has the form $\widehat{f}(\mathbf{x}) = \sum_i (\widehat{\alpha}_i^* - \widehat{\alpha}_i)\mathbf{x}_i^T\mathbf{x} + \beta_0$

Straightforward to kernelize…

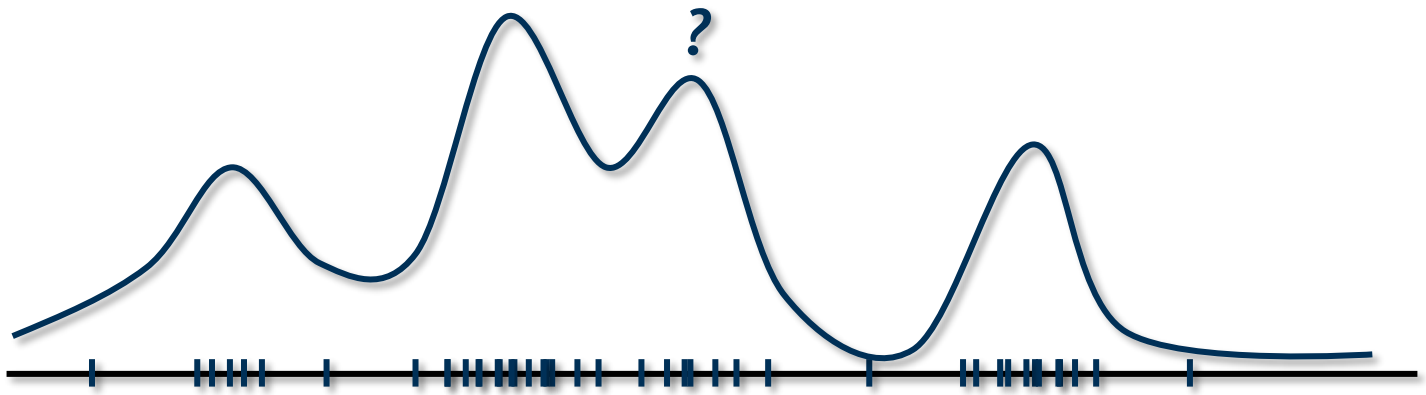
Where else can we use kernels?

- Classification
- Regression
- Density estimation
- PCA/Dimensionality reduction

Density estimation

In density estimation problems, we are given a random sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ from an unknown density $f(\mathbf{x})$

Our objective is to estimate $f(\mathbf{x})$



Applications

Classification

- If we estimate the density for each class, we can simply plug this into the formula for the Bayes' classifier
- Density estimation (for each feature) is the key component in Naïve Bayes

Clustering

- Clusters can be defined by the density: given a point $\mathbf{x}$, climb the density until you reach a local maximum

Anomaly detection

- Given a density estimate $\widehat{f}(\mathbf{x})$, we can use the test

$$\widehat{f}(\mathbf{x}) \leqslant \gamma$$

  to detect anomalies in future observations

Kernel density estimation

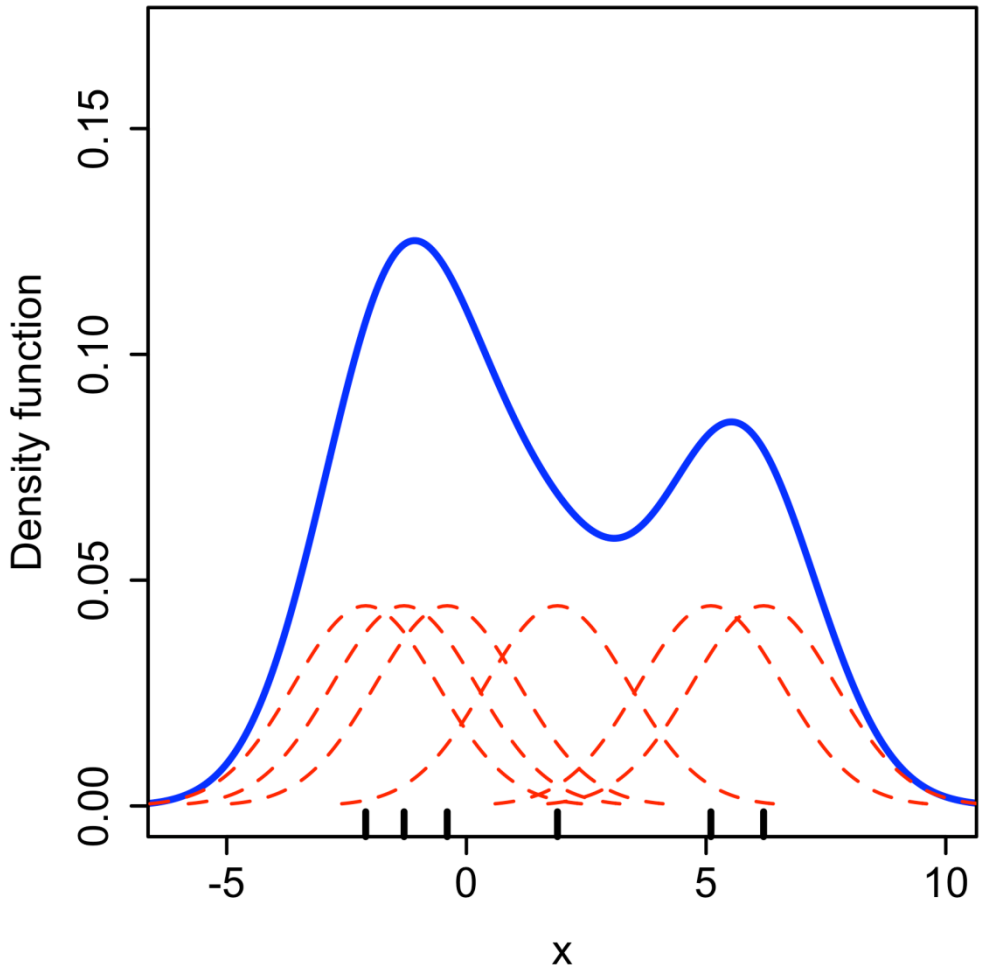
A ***kernel density estimate*** has the form

$$\widehat{f}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} k(\mathbf{x}, \mathbf{x}_i)$$

where k is a kernel

- Another name for this is the ***Parzen window method***
- Looks just like kernel ridge regression, but with equal weights
- The kernel will need to satisfy certain assumptions in order to produce a valid density, but does not necessarily need to be an inner product kernel

Example



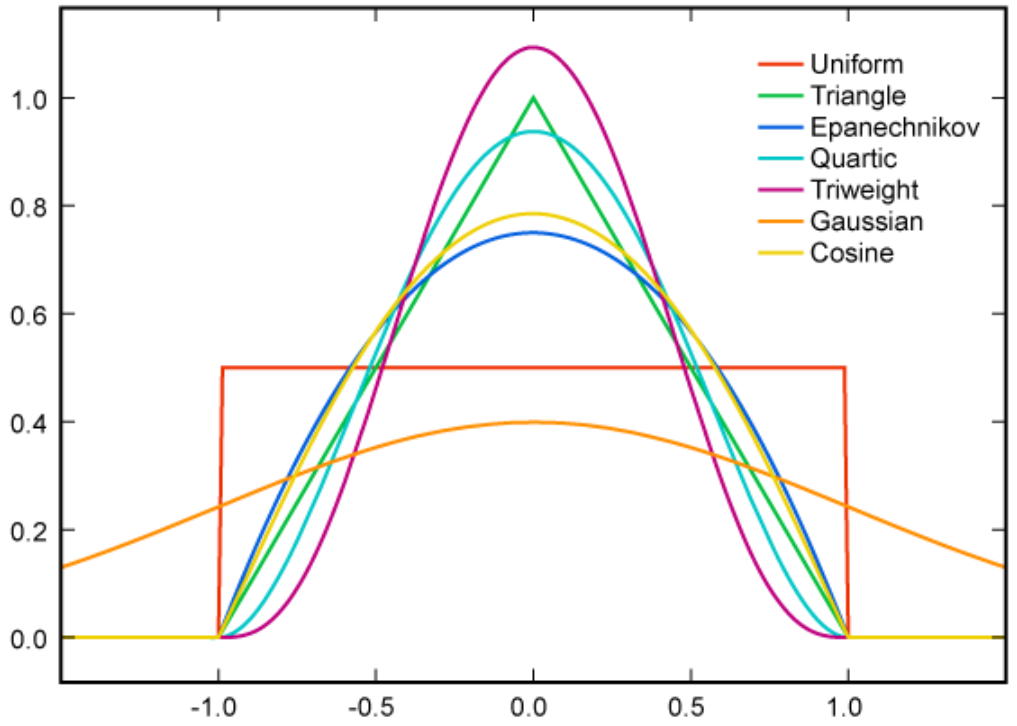
Kernels

In the context of density estimation, a kernel should satisfy

1. $\int k(x,y)dy = 1$
2. $k(x,y) \geq 0$
3. $k(x,y) = D_\sigma(\|y-x\|)$ for some function D_σ

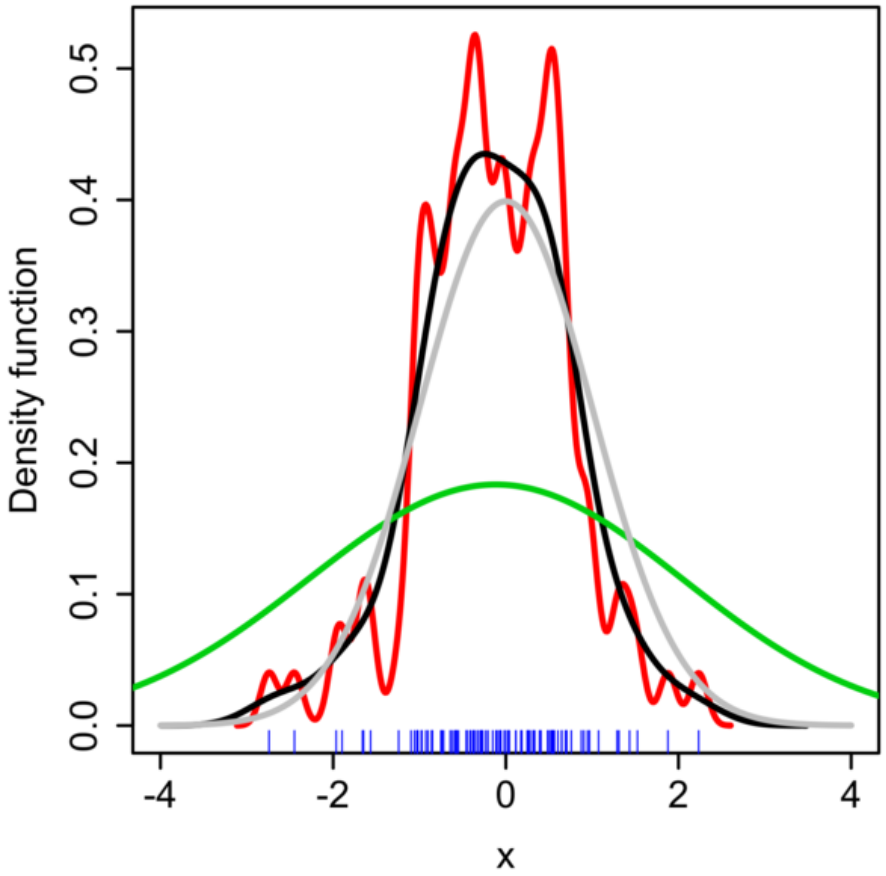
Examples (in $\mathbb{R}$)

- Uniform kernel
- Triangular kernel
- Epanichnikov kernel
- Gaussian
- …



Kernel bandwidth

The accuracy of a kernel density estimate depends critically on the "bandwidth" σ of the kernel



Setting the bandwidth - Theory

Theorem

Let $\widehat{f}_\sigma(\mathbf{x})$ be a kernel density estimate based on the kernel k_σ

Suppose $\sigma = \sigma_n$ is such that

- $\sigma_n \to 0$ as $n \to \infty$
- $n\sigma_n^d \to \infty$ as $n \to \infty$

Then

$$\mathbb{E}\left[\int |\widehat{f}_\sigma(\mathbf{x}) - f(\mathbf{x})| d\mathbf{x}\right] \to 0$$

as $n \to \infty$, regardless of the true density $f(\mathbf{x})$

Proof: See Devroye and Lugosi, *Combinatorial Methods in Density Estimation* (1987)

Setting the bandwidth - Practice

Silverman's rule of thumb

If using the Gaussian kernel, a good choice for σ is

$$\sigma \approx 1.06 \widehat{\sigma} n^{-1/5}$$

where $\widehat{\sigma}$ is the standard deviation of the samples

Alternatively, we can use "model selection" techniques (to be discussed more next time), e.g.:

- Randomly split the data into two sets
- Obtain a kernel density estimate for the first
- Measure how well the second set fits this estimate
  - e.g., compute another kernel density estimate on the second set and calculate the KL divergence between the two

Density estimation is hard…

Kernel density estimation works fairly well if you have lots of data in extremely low-dimensional data

- e.g., 1 or 2 dimensions

Fortunately, it is not strictly necessary in many applications…

Midterm Exam Details

- Midterm exam will be March 7, **in class**
  - Distance students will take exam asynchronously
    - 1 week (self-scheduled) for online students
- You may bring a single (standard sized) sheet of handwritten notes (you can write on both sides)
  - Otherwise, the exam is closed notes
- No calculators
- I will post a past sample test in Canvas
- Exam will cover everything up to maximum margin hyperplanes (not today)
  - Will consist mostly of short answer/multiple choice questions