S&DS 365 / 665

Intermediate Machine Learning

Smoothing and Mercer Kernels

Monday, September 12

Yale

Please note

- Materials posted to `http://interml.ydata123.org`
- Readings from “Probabilistic Machine Learning: An Introduction”
- `https://probml.github.io/pml-book/book1.html`

Some reminders

- Quiz 1: Great job!
- Assn 1 posted on Wednesday
- Topics: Lasso, smoothing, Mercer kernels, some neural nets
- Questions?

Topics for today

- Recap: Smoothing methods
- Demo of smoothing with various kernels
- Mercer kernels

Nonparametric Regression

Given $(X_1, Y_1), \ldots, (X_n, Y_n)$ predict Y from X.

Assume only that $Y_i = m(X_i) + \epsilon_i$ where where $m(x)$ is a smooth function of x.

The most popular methods are *kernel methods*. However, there are two types of kernels:

1. Smoothing kernels
2. Mercer kernels

Smoothing kernels involve local averaging.
Mercer kernels involve regularization.

Smoothing Kernels

- Smoothing kernel estimator:

$$\widehat{m}_h(x) = \frac{\sum_{i=1}^n Y_i \, K_h(X_i, x)}{\sum_{i=1}^n K_h(X_i, x)}$$

where $K_h(x, z)$ is a *kernel* such as

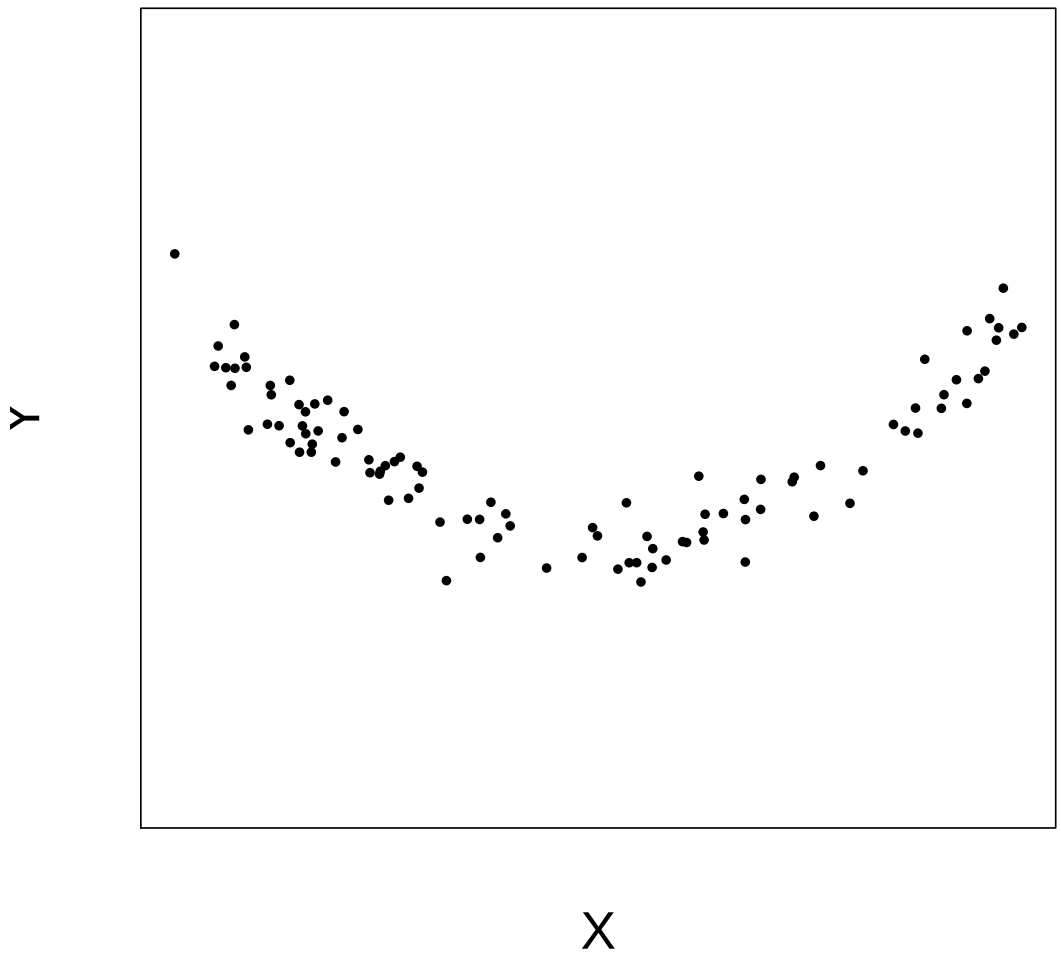
$$K_h(x, z) = \exp\left(-\frac{\|x - z\|^2}{2h^2}\right)$$

and $h > 0$ is called the *bandwidth*.

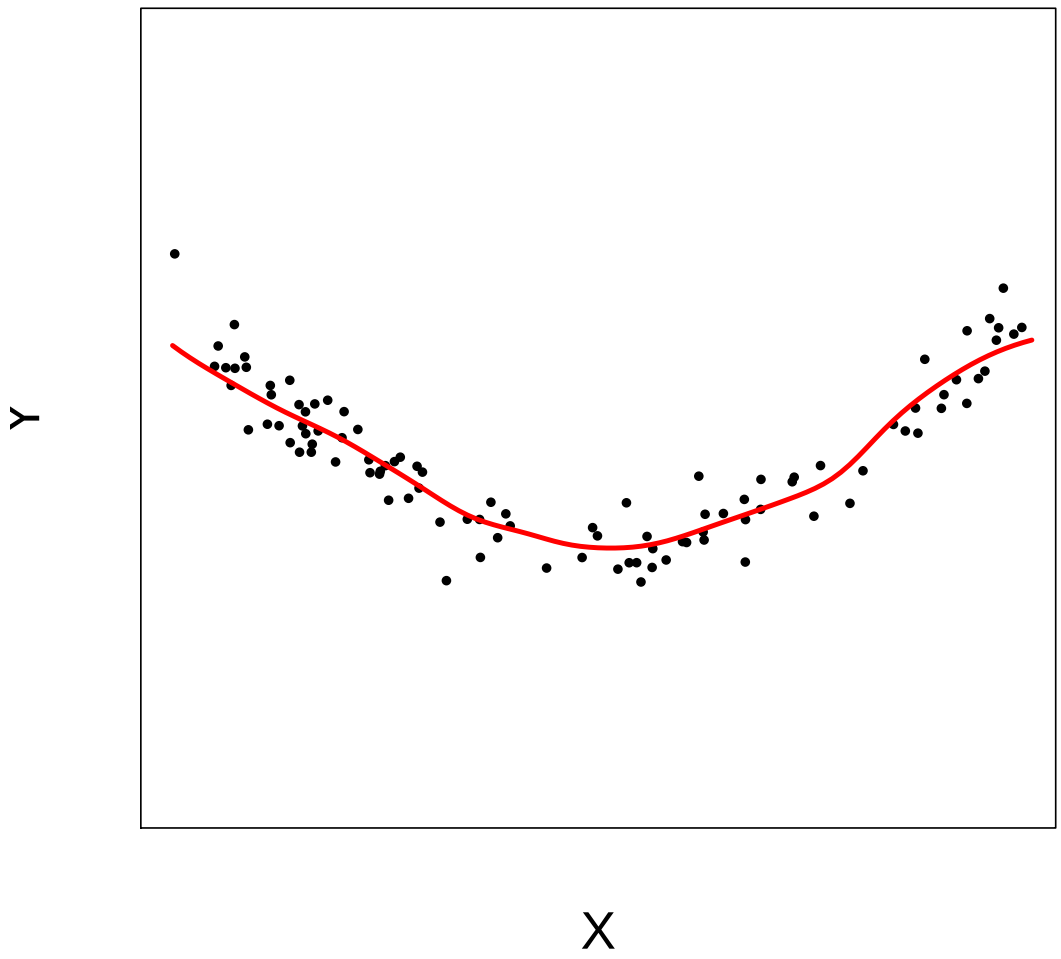
- $\widehat{m}_h(x)$ is just a local average of the Y_i's near x.

- The bandwidth h controls the bias-variance tradeoff:
Small h = large variance while *large h = large bias*.

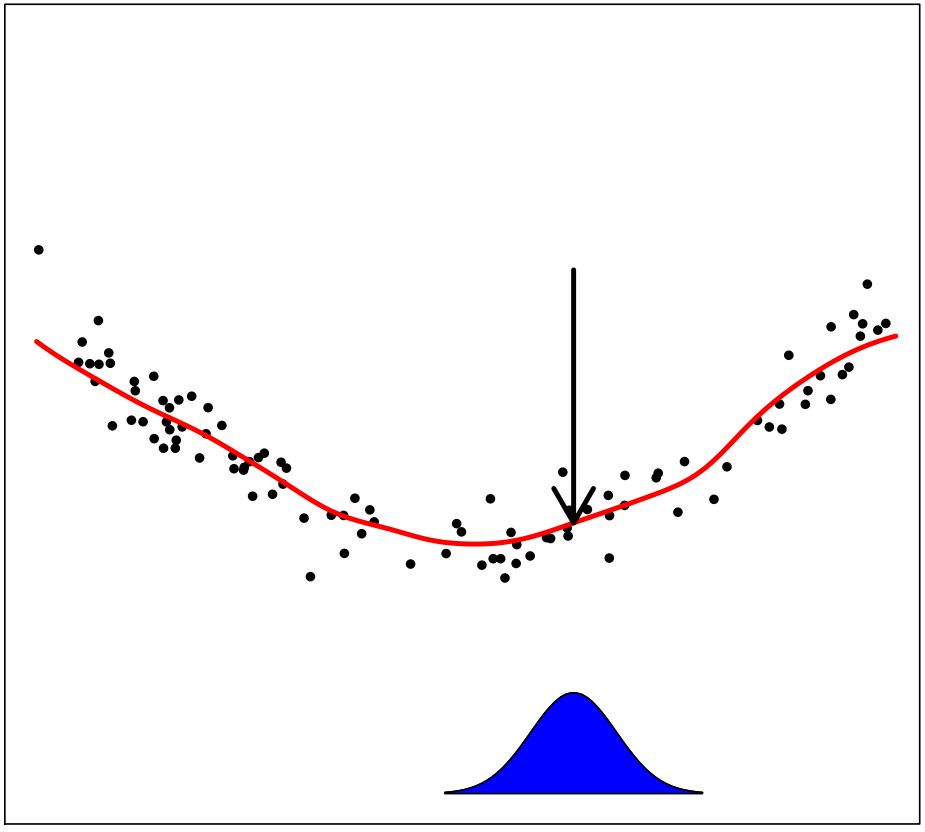
Example: Some Data – Plot of Y_i versus X_i



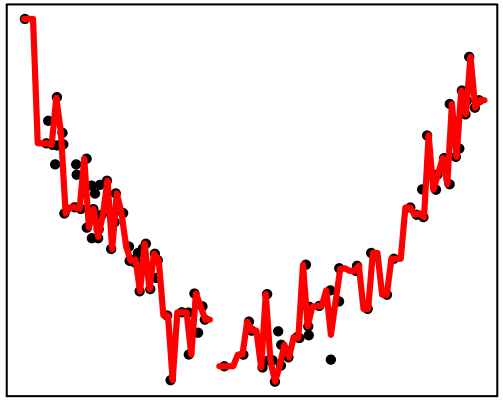
Example: $\widehat{m}(x)$



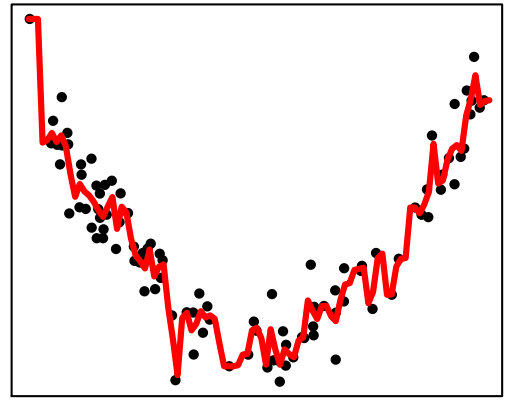
$\widehat{m}(x)$ is a local average



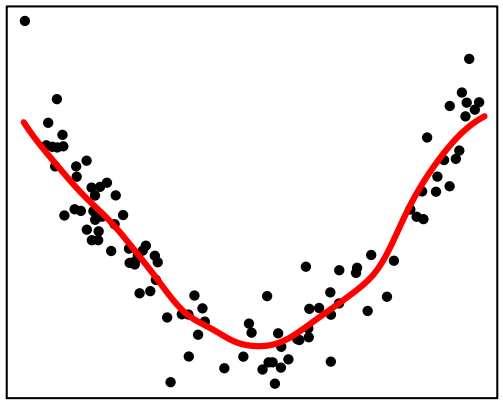
Effect of the bandwidth *h*



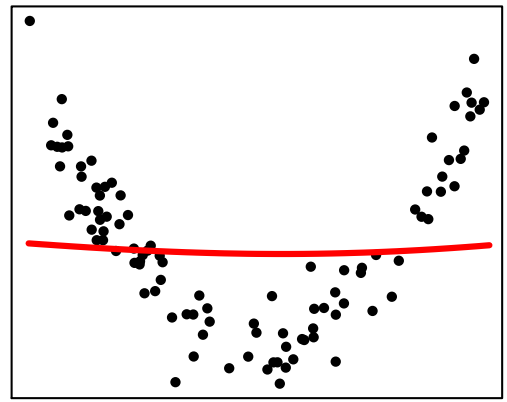
very small bandwidth



small bandwidth



medium bandwidth



large bandwidth

Smoothing Kernels

$$\text{Risk} = \mathbb{E}(Y - \widehat{m}_h(X))^2 = \text{bias}^2 + \text{variance} + \sigma^2.$$

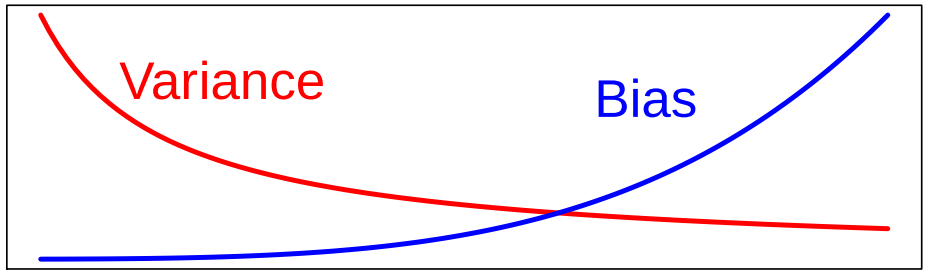
$$\text{bias}^2 \approx h^4,$$

$$\text{variance} \approx \frac{1}{nh^p} \quad \text{where } p = \text{dimension of } X.$$

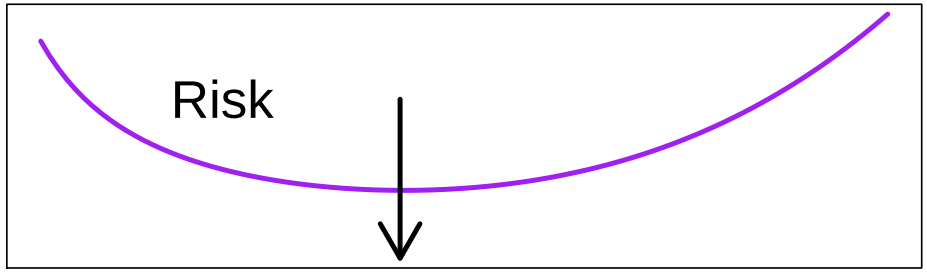
$\sigma^2 = \mathbb{E}(Y - m(X))^2$ is the unavoidable prediction error.

small h: low bias, high variance (undersmoothing)
large h: high bias, low variance (oversmoothing)

Risk Versus Bandwidth



h



optimal h

Estimating the Risk: Cross-Validation

To choose h we need to estimate the risk $R(h)$. We can estimate the risk by using *cross-validation*.

1. Omit (X_i, Y_i) to get $\widehat{m}_{h,(i)}$, then predict: $\widehat{Y}_{(i)} = \widehat{m}_{h,(i)}(X_i)$.
2. Repeat this for all observations.
3. The cross-validation estimate of risk is:

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_{(i)})^2.$$

Shortcut formula: Whenever $\widehat{Y} = LY$ we can use the shortcut

$$\widehat{R}(h) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2.$$

In this case $L_{ii} = K_h(X_i, X_i)/\sum_t K_h(X_i, X_t)$.

Shortcut formula

Let's prove the shortcut formula. Let $K_{ij} = K_h(X_i, X_j)$. We have

$$\begin{aligned}
\widehat{Y}_{(i)} &= \frac{\sum_{j \neq i} K_{ij} Y_j}{\sum_{\jmath \neq i} K_{ij}} \\
&= \frac{\sum_j K_{ij} Y_j - K_{ii} Y_i}{\sum_j K_{ij} - K_{ii}} \\
&= \frac{\sum_j L_{ij} Y_j - L_{ii} Y_i}{1 - L_{ii}} \\
&= \frac{\widehat{Y}_i - L_{ii} Y_i}{1 - L_{ii}}
\end{aligned}$$

To show this for OLS regression we can use the formula for the inverse of a matrix plus a rank-1 matrix.

Shortcut formula

It follows that

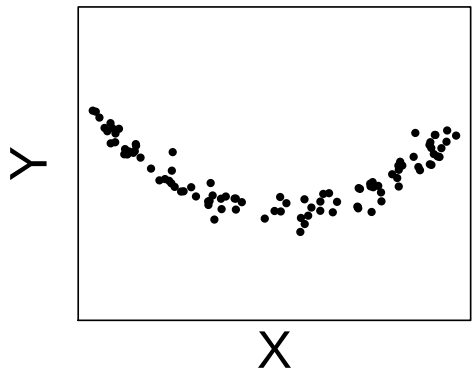
$$\left(Y_i - \widehat{Y}_{(i)}\right)^2 = \left(Y_i - \frac{\widehat{Y}_i - L_{ii}Y_i}{1 - L_{ii}}\right)^2$$
$$= \left(\frac{Y_i - \widehat{Y}_i}{1 - L_{ii}}\right)^2$$

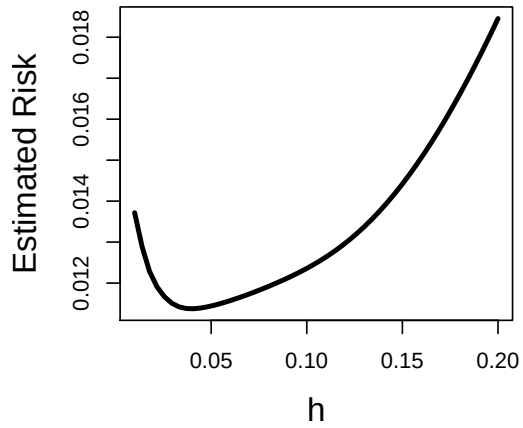
To show this for OLS regression we can use the formula for the inverse of a matrix plus a rank-1 matrix.

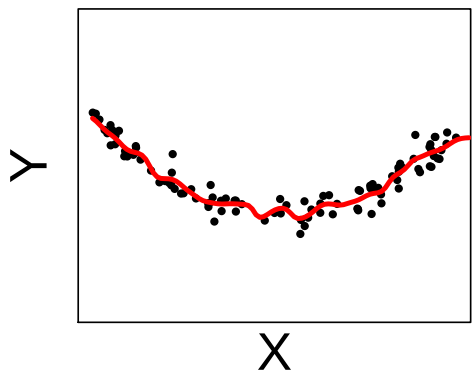
Summary so far

1. Compute $\widehat{m}_h$ for each h.
2. Estimate the risk $\widehat{R}(h)$.
3. Choose bandwidth $\widehat{h}$ to minimize $\widehat{R}(h)$.
4. Let $\widehat{m}(x) = \widehat{m}_{\widehat{h}}(x)$.

Example







Let’s revisit the notebook

Another Approach: Mercer Kernels

Instead of using local smoothing, we can optimize the fit to the data subject to regularization (penalization). Choose $\widehat{m}$ to minimize

$$\sum_i (Y_i - \widehat{m}(X_i))^2 + \lambda \text{ penalty}(\widehat{m})$$

where penalty($\widehat{m}$) is a *roughness penalty*.

λ is a parameter that controls the amount of smoothing.

How do we construct a penalty that measures roughness? One approach is: *Mercer Kernels* and *RKHS = Reproducing Kernel Hilbert Spaces.*

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of "similarity" between points x and x'.

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of "similarity" between points x and x'.

A Mercer kernel has a special property: For any set of points $x_1, \ldots, x_n$ the $n \times n$ matrix

$$\mathbb{K} = \left[K(x_i, x_j)\right]$$

is positive semidefinite (no negative eigenvalues)

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of "similarity" between points x and x'.

A Mercer kernel has a special property: For any set of points $x_1, \ldots, x_n$ the $n \times n$ matrix

$$\mathbb{K} = \left[K(x_i, x_j)\right]$$

is positive semidefinite (no negative eigenvalues)

This property has many important (and beautiful!) mathematical consequences.

Mercer Kernels: Key example

A Gaussian gives us a Mercer kernel:

$$K(x, x') = e^{-\frac{\|x-x'\|^2}{2h^2}}$$

Note: Here we fix the bandwidth h.

What is a Mercer Kernel?

A *Mercer kernel* $K(x, x')$ is symmetric and positive semidefinite bivariate function:

$$\int \int f(x) f(x') K(x, x') \, dx \, dx' \geq 0$$

for all (univariate) functions f.

Basis functions

We can create a set of *basis functions* based on K.

Fix z and think of $K(z, x)$ as a function of x. That is,

$$K(z, x) = K_z(x)$$

is a function of the second argument, with the first argument fixed.

Defining a norm from the kernel

Because of the positive semidefinite property, we can create an *inner product* and *norm* over the span of these functions

If $f(x) = \sum_r \alpha_r K_{z_r}(x)$, $g(x) = \sum_s \beta_s K_{y_s}(x)$, the inner product is

$$\begin{aligned}\langle f, g \rangle_K &= \sum_r \sum_s \alpha_r \beta_s K(z_r, y_s) \\ &= \alpha^T \mathbb{K} \beta\end{aligned}$$

where $\mathbb{K} = [K(z_r, y_s)]$

Defining a norm from the kernel

Because of the positive semidefinite property, we can create an *inner product* and *norm* over the span of these functions

The norm is

$$\begin{aligned} \|f\|_K^2 = \langle f, f \rangle_K &= \sum_r \sum_s \alpha_r \alpha_s K(z_r, z_s) \\ &= \alpha^T \mathbb{K} \alpha \geq 0 \end{aligned}$$

Defining a Hilbert space from the kernel

This gives us an infinite dimensional space of functions with a geometry — a notion of angle from the inner product $\langle \cdot, \cdot \rangle_K$

Technically speaking, we define the Hilbert space by "completing" the functions to include the limits of all Cauchy sequences with respect to the norm.

Defining a Hilbert space from the kernel

This gives us an infinite dimensional space of functions with a geometry — a notion of angle from the inner product $\langle \cdot, \cdot \rangle_K$

It is called a *Reproducing Kernel Hilbert Space* (RKHS) because

$$\langle f, K_x(\cdot) \rangle_K = f(x)$$

That is, the kernel "reproduces" the values of the functions through the inner products

---

Technically speaking, we define the Hilbert space by "completing" the functions to include the limits of all Cauchy sequences with respect to the norm.

Defining a Hilbert space from the kernel

This gives us an infinite dimensional space of functions with a geometry — a notion of angle from the inner product $\langle \cdot, \cdot \rangle_K$

It is called a *Reproducing Kernel Hilbert Space* (RKHS) because

$$\langle f, K_x(\cdot) \rangle_K = f(x)$$

That is, the kernel "reproduces" the values of the functions through the inner products

Exercise: Verify this identity!

Technically speaking, we define the Hilbert space by "completing" the functions to include the limits of all Cauchy sequences with respect to the norm.

Nonparametric regression using Mercer kernels

The norm gives us a way to penalize functions for being too complex.

We carry out least squares regression subject to this penalty:

Minimize

$$\sum_{i=1}^{n}(Y_i - m(X_i))^2 + \lambda \|m\|_K^2.$$

over the RKHS of functions

Dilemma?

How do we carry out this penalized regression? It looks complicated!

Or maybe intractable...

Linear algebra to the rescue!

Representer Theorem

Let $\widehat{m}$ minimize

$$J(m) = \sum_{i=1}^{n} (Y_i - m(X_i))^2 + \lambda \|m\|_K^2.$$

Then

$$\widehat{m}(x) = \sum_{i=1}^{n} \alpha_i K(X_i, x)$$

for some $\alpha_1, \ldots, \alpha_n$.

So, we only need to find the coefficients

$$\alpha = (\alpha_1, \ldots, \alpha_n).$$

Mercer kernel regression

Plug $\widehat{m}(x) = \sum_{i=1}^{n} \alpha_i K(X_i, x)$ into J:

$$J(\alpha) = \|Y - \mathbb{K}\alpha\|^2 + \lambda \alpha^T \mathbb{K} \alpha$$

where $\mathbb{K}_{jk} = K(X_j, X_k)$

Now we find α to minimize J. We get (Assn 1):

$$\widehat{\alpha} = (\mathbb{K} + \lambda I)^{-1} Y$$

$$\widehat{m}(x) = \sum_i \widehat{\alpha}_i K(X_i, x)$$

Mercer kernel regression

The estimator depends on the amount of regularization λ.

Again, there is a bias-variance tradeoff.

We choose λ by cross-validation. This is like the bandwidth in smoothing kernel regression.

Takeaways

- Mercer kernels have a special property: When restricted to a finite sample they give positive semidefinite matrices
- This allows us to define an inner product and a norm
- We use the norm to do *penalization* of the functions

The underlying math is rich—see the notes if you want to learn more!

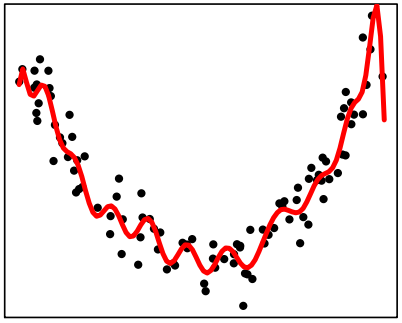
Smoothing Kernels *Versus* Mercer Kernels

Smoothing kernels: bandwidth h controls the amount of smoothing.

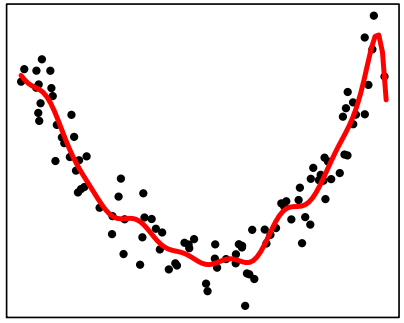
Mercer kernels: norm $\|f\|_K$ controls the amount of smoothing.

In practice these two methods give answers that are very similar.

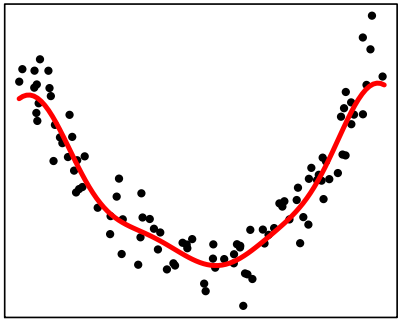
Mercer Kernels: Examples



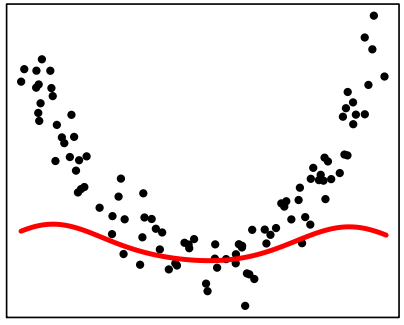
very small lambda



small lambda



medium lambda



large lambda

The importance of being Kernel-est

- Mercer kernels play a central role in machine learning
- Why? We can define similarity functions that are kernels for all kinds of data — graphs, molecules, text documents
- Mercer kernels are also important for modern understanding of deep neural networks

Summary for today

- Smoothing methods compute local averages, weighting points by a kernel. The details of the kernel don't matter much
- Mercer kernels using penalization rather than smoothing
- Defining property: Matrix $\mathbb{K}$ is always positive semidefinite
- Equivalent to a type of ridge regression in function space
- The curse of dimensionality limits use of both approaches