Lecture 12: Non-linear Machine Learning

Statistical Learning and Data Mining

Xingye Qiao

Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

Read: About kernel - ELSII Chs. 5.8, 6, 14.5.4
About smoothing (nonlinear) methods: ISLR Ch. 7, and ELSII Chs. 5, 6, and 9.1.

Outline

The next section would be . . . . . .

Disclaimer

- The kernel trick that I showed you applies much more broadly than SVM, but we'll use it for SVM's for illustration.
- Warning: This lecture is technical. Try to get the basic idea even if you don't catch all the details.

What we know so far about support vector machines:

- A generalization of the max margin classifier that allows the margin to be violated by an amount governed by a slack variable ξ_i.
- Kernel SVM is linear SVM applied to augmented (transformed) data $(\phi(x_i), y_i)_{i=1,\ldots,n}$.
- However, we only need to replaced $\phi(x_i)^T \phi(x_j)$ by $k(x_i, x_j)$.
- Sometimes, we have explicit form for ϕ corresponding to a kernel function k. Sometimes we don't.

Road Map

- definitions (inner product, Hilbert space, kernel)
- some intuitions
- a general Hilbert space whose inner product is the kernel
- reproducing property
- create a totally different representation of the space, which is a more intuitive way to express the kernel
- representer theorem
- nice properties of kernels, and how you might construct them

Inner Product

An **inner product** takes two elements of a vector space $\mathcal{X}$ and outputs a number. An inner product could be a usual dot product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^T \boldsymbol{v}$ or something fancier. An inner product $\langle \cdot, \cdot \rangle$ must satisfy the following conditions:

- Symmetry: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \langle \boldsymbol{v}, \boldsymbol{u} \rangle$
- Bilinearity: $\langle a\boldsymbol{u} + b\boldsymbol{w}, \boldsymbol{v} \rangle = a\langle \boldsymbol{u}, \boldsymbol{v} \rangle + b\langle \boldsymbol{w}, \boldsymbol{v} \rangle$
- Strict Positive Definiteness:

$$\langle \boldsymbol{u}, \boldsymbol{u} \rangle \geq 0$$

where equality hold only if $\boldsymbol{u} = \boldsymbol{0}$

Hilbert Space

A **Hilbert space** is a complete inner product space. ('Complete' means sequences converge to elements of the space - there aren't any "holes" in the space.)
Examples of Hilbert spaces

- $\mathbb{R}^p$
- The space of square summable sequences:
  $\{(u_1, u_2, \dots) : \sum_{i=1}^{\infty} u_i^2 < \infty\}$ with inner product
  $\langle u, v \rangle = \sum_{i=1}^{\infty} u_i v_i$
- The space $L_2(\mathcal{X}, \mu)$ of square integrable functions:
  $\{f : \int f(x)^2 d\mu(x)\}$ with inner product
  $\langle f, g \rangle_{L_2(\mathcal{X},\mu)} = \int f(x)g(x)d\mu(x)$

Kernel function (Finite States)

Suppose the sample space has at most n different inputs.

- Goal: use a function $k(\cdot, \cdot)$ to define inner product.
- Symmetric: $k(x_i, x_j) = k(x_j, x_i)$
- Let matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ be $\mathbf{K}_{ij} = k(x_i, x_j)$. Then $\mathbf{K}$ is a symmetric matrix.
- Eigen-decomposition: $\mathbf{K} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$.
- Let $\phi(x_i) = (\boldsymbol{v}^{(i)}\mathbf{\Lambda}^{1/2})^T$ where $\boldsymbol{v}^{(i)}$ is the ith row of $\mathbf{V}$. Easy to see that

$$\mathbf{K}_{ij} = (\boldsymbol{v}^{(i)}\mathbf{\Lambda}^{1/2})(\mathbf{\Lambda}^{1/2}{\boldsymbol{v}^{(j)}}^T) = \phi(x_i)^T\phi(x_j)$$

  So $k(x_i, x_j)$ is indeed the dot product of some features.
- All λ_t's should be nonnegative. (WHY?)

Conclusion: **$\mathbf{K}$ should be positive semi-definite.**

Kernel function

- For infinite case, pick m inputs. We require $\mathbf{K}$ to be positive semi-definite no matter what examples we get.
- Officially, a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if
  1. k is symmetric.
  2. For any $x_1, \ldots, x_n$ chosen from $\mathcal{X}$, the Gram matrix defined by $\mathbf{K}_{ij} = k(x_i, x_j)$ is PSD (positive semi-definite).

A Special Hilbert Space

Define $R^{\mathcal{X}} := \{f : \mathcal{X} \mapsto \mathbb{R}\}$, the space of all functions that map from $\mathcal{X}$ to $\mathbb{R}$.

We define the feature map $\phi : \mathcal{X} \mapsto R^{\mathcal{X}}$ so that it maps x to $k(\cdot, x)$:

$$\phi(x) : x \mapsto k(\cdot, x)$$

Note that when the second argument is fixed as x, $k(\cdot, x)$ only has first argument left and is a member of $R^{\mathcal{X}}$.

Bear with me for this temporary confusion. Is $\phi(x)$ now a function or a mapped feature (like $(x_1^2, x_2^2, \sqrt{2}x_1x_2)'$ discussed before)? It will come clearer very soon.

$\phi(x)$ is a function

So we turned each x into a function on the domain $\mathcal{X}$. Each function could be thought of as an infinite dimensional vector. These functions will be elements of our Hilbert space. (But what is the inner product in this Hilbert space??)

Will show: k is an inner product of this Hilbert space. To do this:

1. Define feature map $\phi : \mathcal{X} \mapsto R^{\mathcal{X}}$ (already did), then form a vector space.
2. Define an inner product $\langle \phi(x), \phi(z) \rangle_{H_k}$
3. Show that

$$k(x, z) = \langle \phi(x), \phi(z) \rangle_{H_k}$$

Step 1: build the vector space

A typical element in the vector space looks like:

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$$

Imagine that $f(\cdot)$ is a "vector" and so is $k(\cdot, x_i)$.
The vector space is

$$\left\{ f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) : m \in \mathbf{N}, x_i \in \mathcal{X}, \alpha_i \in \mathbb{R} \right\}$$

Step 2: define the inner product

For two functions in this space $f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$ and $g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\cdot, x'_j)$, we **define** the inner product as

$$\langle f, g \rangle_{H_k} = \sum_i \sum_j \alpha_i \beta_j k(x_i, x'_j)$$

Is this really a valid inner product??? Easy to check it is

1. Symmetric
2. Bilinear
3. Positive semi-definite $\langle f, f \rangle_{H_k} \geq 0$

and $\langle f, f \rangle_{H_k} = 0$ only when $f = 0$ (will come back for this)

Step 3: magic happens

The inner product of $k(\cdot, x)$ and f is just the evaluation of f at x

$$\langle k(\cdot, x), f \rangle_{H_k} = f(x)$$

(WHY??) Then, as a special case:

$$\langle k(\cdot, x), k(\cdot, x') \rangle_{H_k} = k(x, x')$$

Or state it differently

$$\langle \phi(x), \phi(x') \rangle_{H_k} = k(x, x')$$

This is why k is called a **reproducing kernel**, and the Hilbert space is a **reproducing kernel Hilbert space (RKHS)** (almost).

Come back to the missing piece

Must show that $\langle f, f \rangle_{H_k} = 0$ only when $f = 0$.

For any x,

$$|f(x)|^2 = \underbrace{|\langle k(\cdot, x), f \rangle_{H_k}|^2 \leq \langle k(\cdot, x), k(\cdot, x) \rangle_{H_k} \langle f, f \rangle_{H_k}}_{\text{CauchySchwarz inequality}} = k(x, x) \langle f, f \rangle_{H_k}$$

Therefore, if $\langle f, f \rangle_{H_k} = 0$, then $f = 0$ for all x.

The completeness

So this space is a vector space with a valid inner product. But we also need completeness to make it a Hilbert space.

This is not a big deal since we can fill in the "holes" by adding the limit points of sequences that converge in the norm

$$\|f\|_{H_k} := \langle f, f \rangle_{H_k}.$$

Then finally, we have a **reproducing kernel Hilbert space (RKHS)**

The role of RKHS in SVM

For linear SVM, the linear kernel $k(x, z) = x'z$ is used and $f(x) = x^T\omega = \sum_{i=1}^n \alpha_i x^T x_i = \sum_{i=1}^n \alpha_i k(x, x_i)$. In this case $\|f\|_{H_k} = \langle f, f\rangle_{H_k} = \sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j k(x_i, x_j) = \sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j x_i^T x_i = (\sum_{i=1}^n \alpha_i x_i)^T(\sum_{j=1}^n \alpha_j x_j) = \omega^T\omega = \|\omega\|^2$

Generally the SVM optimization problem (assuming no intercept) can be expressed as

$$f^* = \underset{f\in H_k}{\operatorname{argmin}} \sum_{i=1}^n H(y_i f(x_i)) + \lambda\|f\|_{H_k}$$

Or consider a generic loss function

$$f^* = \underset{f\in H_k}{\operatorname{argmin}} \sum_{i=1}^n \ell(y_i f(x_i)) + \lambda\|f\|_{H_k}$$

Representer Theorem

$$f^* = \operatorname*{argmin}_{f \in H_k} \sum_{i=1}^{n} \ell(y_i f(x_i)) + \lambda \|f\|_{H_k}$$

Theorem (Representer Theorem)

The solution to the problem above can be written as

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$$

This shows that to solve the SVM optimization problem, we only need to solve for the α_i's. This agrees with the solution from the Lagrangian formulation for SVM.

Proof: Project f onto subspace $S := \{k(x_i, \cdot) : 1 \leq i \leq n\}$, obtaining f_s and $f_\perp$ with $f = f_s + f_\perp$

1. $\|f\|_{H_k} = \|f_s\|_{H_k} + \|f_\perp\|_{H_k} \geq \|f_s\|_{H_k}$. So $\|f\|_{H_k}$ is minimized if $f_\perp = 0$
2. Using the reproducing property: $f(x_i) = \langle f, k(x_i, \cdot)\rangle_{H_k} = \langle f_s, k(x_i, \cdot)\rangle_{H_k} + \langle f_\perp, k(x_i, \cdot)\rangle_{H_k} = \langle f_s, k(x_i, \cdot)\rangle_{H_k}$. So the loss function depends on only the component of f in the subspace S.

To minimize: we just let $f_\perp = 0$ so that we can express

$$f^*(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot)$$

Examples of popular kernels

- The linear kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}'\boldsymbol{y}.$$

  This leads to the original, linear SVM.

- The polynomial kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = (c + \boldsymbol{x}'\boldsymbol{y})^d.$$

  We can write down the expansion explicitly, so the mapping ϕ can be explicitly expressed.

- The Gaussian (radial basis function) kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{1}{\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2\right).$$

  The feature space where $\phi(\boldsymbol{x})$ lies is infinite dimensional. (The mapping ϕ is only implicitly defined)

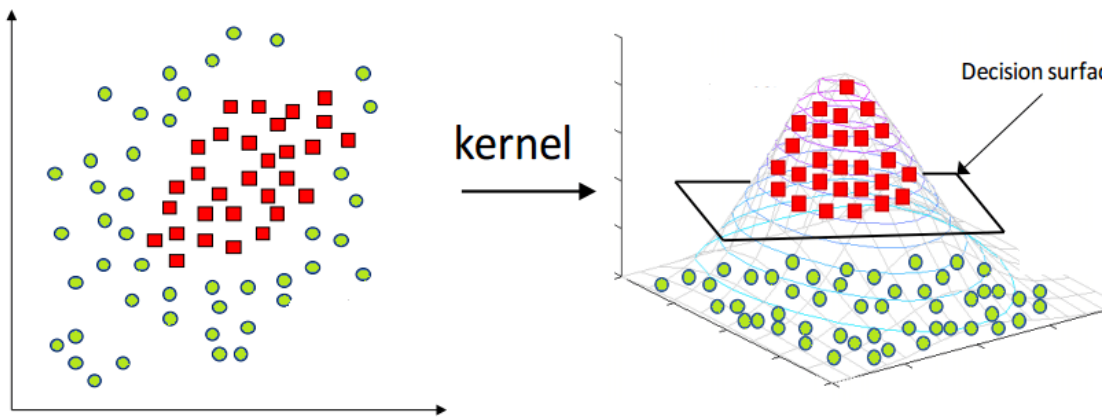
Constructing Kernels

Construct new kernels from previously defined kernels. Suppose k_1 and k_2 are two valid kernels. The followings are also valid

- $k(x, z) = \alpha_1 k_1(x, z) + \alpha_2 k_2(x, z)$ for $\alpha_1, \alpha_2 > 0$
- $k(x, z) = k_1(x, z) k_2(x, z)$
- $k(x, z) = k(f(x), f(z))$ for $f : \mathcal{X} \mapsto \mathcal{X}$.
- $k(x, z) = g(x) g(z)$ for $g : \mathcal{X} \mapsto \mathbb{R}$.
- $k(x, z) = f(k_1(x, z))$ for f a polynomial with positive coefficient.
- $k(x, z) = \exp(k_1(x, z))$ - WHY?
- $k(x, z) = \exp(-\|x - z\|^2 / \sigma^2)$ - WHY?

Which kernel to use?

- No one is uniformly better than the others
- My personal favorite (other than linear kernel) is Gaussian RBF kernel



Which kernel to use?

- Hyper-parameter tuning is also important: use the overall scale as a benchmark. Try the 25%, 50% and 75% quantiles of pairwise distances.
- In low-dimensional data, if the separating boundary seems to be nonlinear, then kernel SVM usually works better
- For high-dimensional data ($d \gg n$), linear SVM is often sufficient and gives better interpretations. **Simpler solution is a better solution.**

The next section would be . . . . . .

1. Kernels & Reproducing Kernel Hilbert Space (RKHS)
2. Kernel Extensions of Other Methods
3. Overview of Non-linear Regression & Regularization

Kernel Ridge Regression

Ridge Regression: $f(x) = \omega^T x$ where ω minimizes

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - \omega^T x_i)^2 + \frac{1}{2}\lambda\|\omega\|^2$$

- What if $f(x)$ is not a linear function of x but is from a RKHS H_k which looks like $\{f(\cdot) = \sum_{j=1}^{\infty} a_j k(\cdot, x_j) : x_j \in \mathcal{X}\}$?

$$\min_{f \in H_k} \frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \frac{1}{2}\lambda\|f\|_{H_k}$$

$$\min_{f \in H_k} \frac{1}{2} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \frac{1}{2} \lambda \| f \|_{H_k}$$

The representer's theorem tells us that f can be expressed as

$$f(x) = \sum_{j=1}^{n} \alpha_j k(x_j, x) \text{ where } x_j \in \text{training data}$$

so that the problem becomes

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \| \boldsymbol{Y} - \mathbf{K}\boldsymbol{\alpha} \|_2^2 + \frac{1}{2} \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}$$

which has solution

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbb{I})^{-1} \boldsymbol{Y} \text{ and prediction}$$

$$\hat{f}(x) = \sum_{j=1}^{n} \hat{\alpha}_j k(x_j, x)$$

Ridge Regression and KRR

Recall that the solution to the ridge regression is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbb{I}_p)^{-1}\mathbf{X}^T\boldsymbol{y} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbb{I}_n)^{-1}\boldsymbol{y}$$

and the prediction for $\mathbf{X}^*$ is

$$\mathbf{X}^*\hat{\boldsymbol{\beta}} = \mathbf{X}^*\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbb{I}_n)^{-1}\boldsymbol{y}$$

Recall the **kernel trick**:
replace $\mathbf{X}\mathbf{X}^T$ by the matrix $\mathbf{K}$ with $\mathbf{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$; and replace $\mathbf{X}^*\mathbf{X}^T$ in similar manner.

Gaussian Process Regression

KRR has Bayesian interpretation via Gaussian Process Regression:

- $\boldsymbol{y} = (y_1, \ldots, y_n)'$ and $\boldsymbol{f} = (f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n))'$. Have likelihood

$$p(\boldsymbol{y}|f) = N(\boldsymbol{y}|\boldsymbol{f}, \lambda\mathbb{I})$$

- Impose a Gaussian process prior over functions f

$$p(\boldsymbol{f}) = N(\boldsymbol{f}|0, \mathbf{K}) \text{ with covariance } \mathbf{K} \in \mathbb{R}^{n\times n}$$

- Then $\boldsymbol{y}$'s marginal distribution $p(\boldsymbol{y}) = N(\boldsymbol{y}|0, \mathbf{K} + \lambda\mathbb{I}_n)$
- Consider a new y^* corresponding to x^*. We have

$$p(\boldsymbol{y}, y^*) = N(\begin{bmatrix}\boldsymbol{y}\\ y^*\end{bmatrix} | \begin{bmatrix}\mathbf{0}\\ 0\end{bmatrix}, \begin{bmatrix}\mathbf{K}+\lambda\mathbb{I} & \boldsymbol{k}\\ \boldsymbol{k}' & c\end{bmatrix})$$

Gaussian process regression: easy to show that

$$\mathrm{E}[y^*|\boldsymbol{y}] = \boldsymbol{k}^T(\mathbf{K}+\lambda\mathbb{I})^{-1}\boldsymbol{y},\ \mathrm{Var}(y^*|\boldsymbol{y}) = c - \boldsymbol{k}^T(\mathbf{K}+\lambda\mathbb{I})^{-1}\boldsymbol{k}.$$

PCA and Kernel PCA

Principal Component Analysis can also be kernelized. This will be discussed when we learn PCA in details.

Central idea: replace $\mathbf{X}\mathbf{X}^T$ in the procedure by $\mathbf{K}$

Kernels - Take Home Messages

- Construct proper kernels and don't worry about the math.
- Replace inner products with kernels in ANY statistical learning problem.

The next section would be . . . . . .

1. Kernels & Reproducing Kernel Hilbert Space (RKHS)
2. Kernel Extensions of Other Methods
3. Overview of Non-linear Regression & Regularization

Non-linear Regression & Regularization

We now give a survey of other non-linear methods.

1. Piecewise Polynomials
2. Step functions
3. Regression splines
4. Smoothing splines
5. Local regression
6. Generalized additive models
7. Adaptive trend filtering

Polynomial Regression

Standard linear model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

is replaced by

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i$$

which is solved by least square. Note three problems:

1. Multicollinearity
2. Variance of the fit is large at the boundary (so-called boundary effect.)
3. Overfit.

Step functions

Define (choose) cut points (or knots) $c_1, c_2, \ldots, c_k$ and step (indicator) functions

$$\begin{aligned} C_0(x) &= \mathbb{1}\{x < c_1\} \\ C_1(x) &= \mathbb{1}\{c_1 \leq x < c_2\} \\ &\vdots \\ C_{k-1}(x) &= \mathbb{1}\{c_{k-1} \leq x < c_k\} \\ C_k(x) &= \mathbb{1}\{x \geq c_k\} \end{aligned}$$

Then use model

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_k C_k(x_i) + \epsilon_i$$

No multicollinearity. Actually, the fitted value is the mean of the observations in each interval.

Basis Functions

Polynomial and piecewise-constant regression models are in fact special cases of a basis function approach. The idea is to have at hand a family of basis function or transformations that can be applied to a variable X, $b_1(x)$, $b_2(x)$, $\ldots$, $b_k(x)$, then fit

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_k b_k(x_i) + \epsilon_i$$

Next we introduce **Regression splines**

Cubic splines

- Define a set of knots $c_1 < c_2 < \cdots < c_k$
- We want the function $Y = f(X)$ to
  1. Be a **cubic polynomial** between every pair of knots
  2. Be **continuous** at each knot, and
  3. Have **continuous first and second derivatives** at each knot.

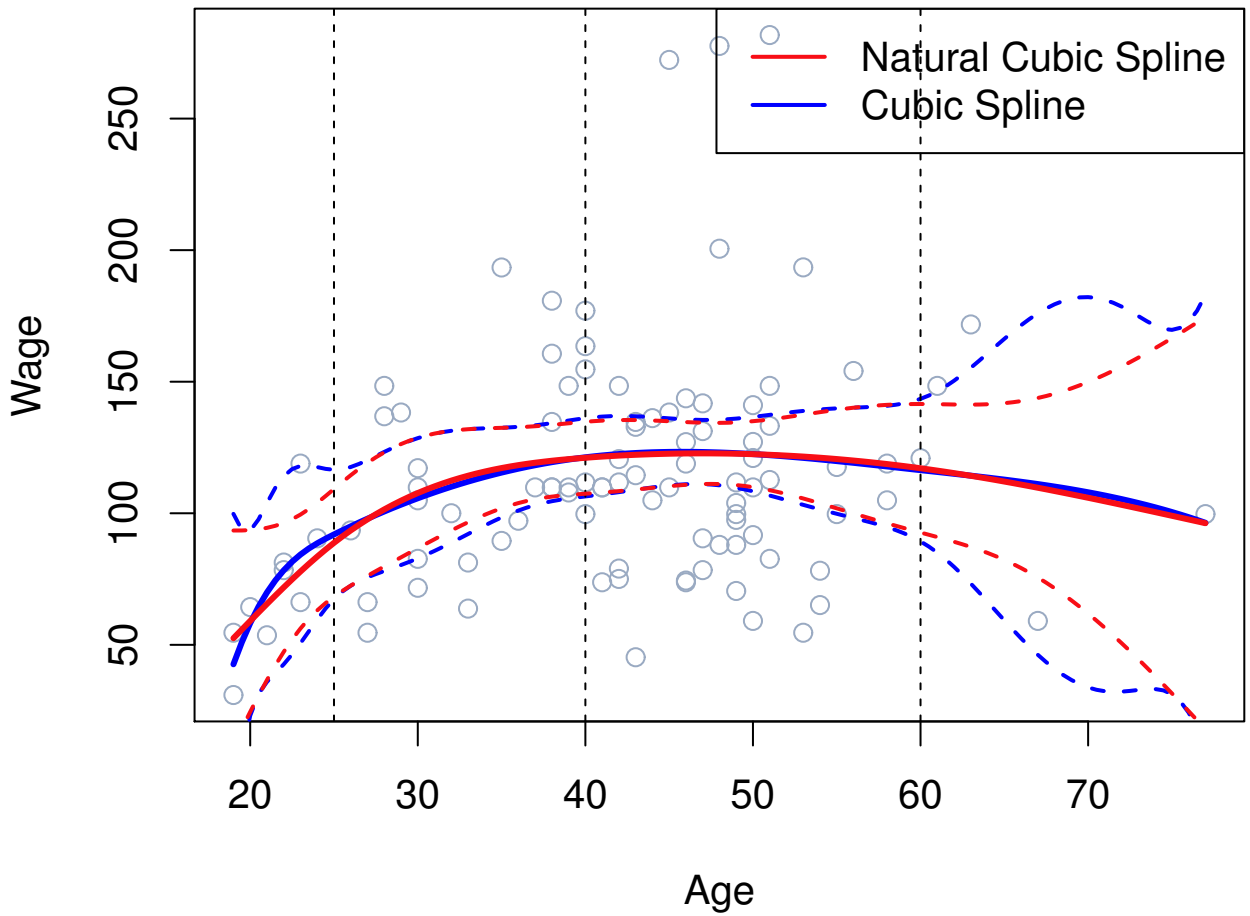
$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 h(X, c_1) + \cdots + \beta_{k+3} h(X, c_k)$$

$$h(X, c) = \begin{cases} (x - c)^3 & \text{if } x > c \\ 0 & \text{otherwise} \end{cases}$$

- Popular because (allegedly) most human eyes cannot detect discontinuities of third derivate at knots.

Natural cubic spline

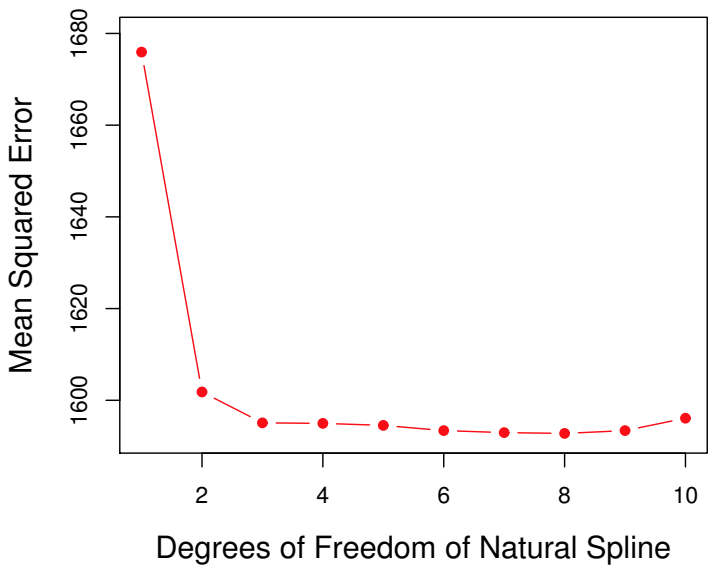
Cubic spline with additional boundary constraints: **be linear at the boundary.**

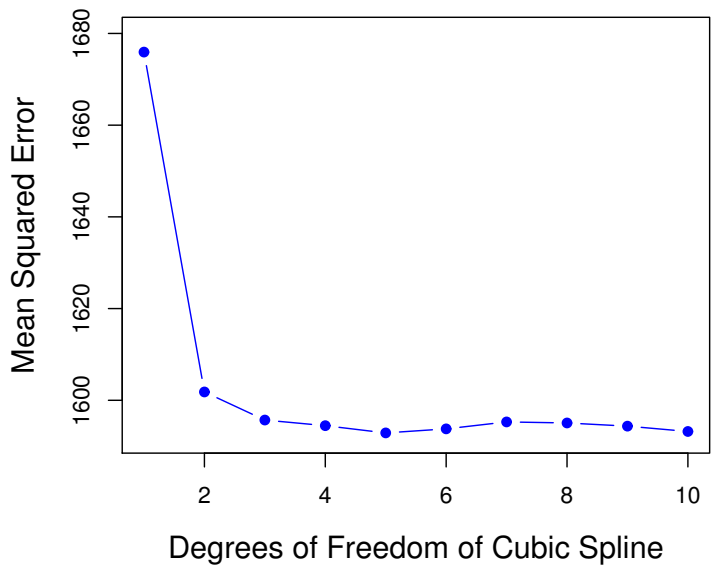


The predictions are more stable for extreme values of X.

Choosing the number and locations of knots

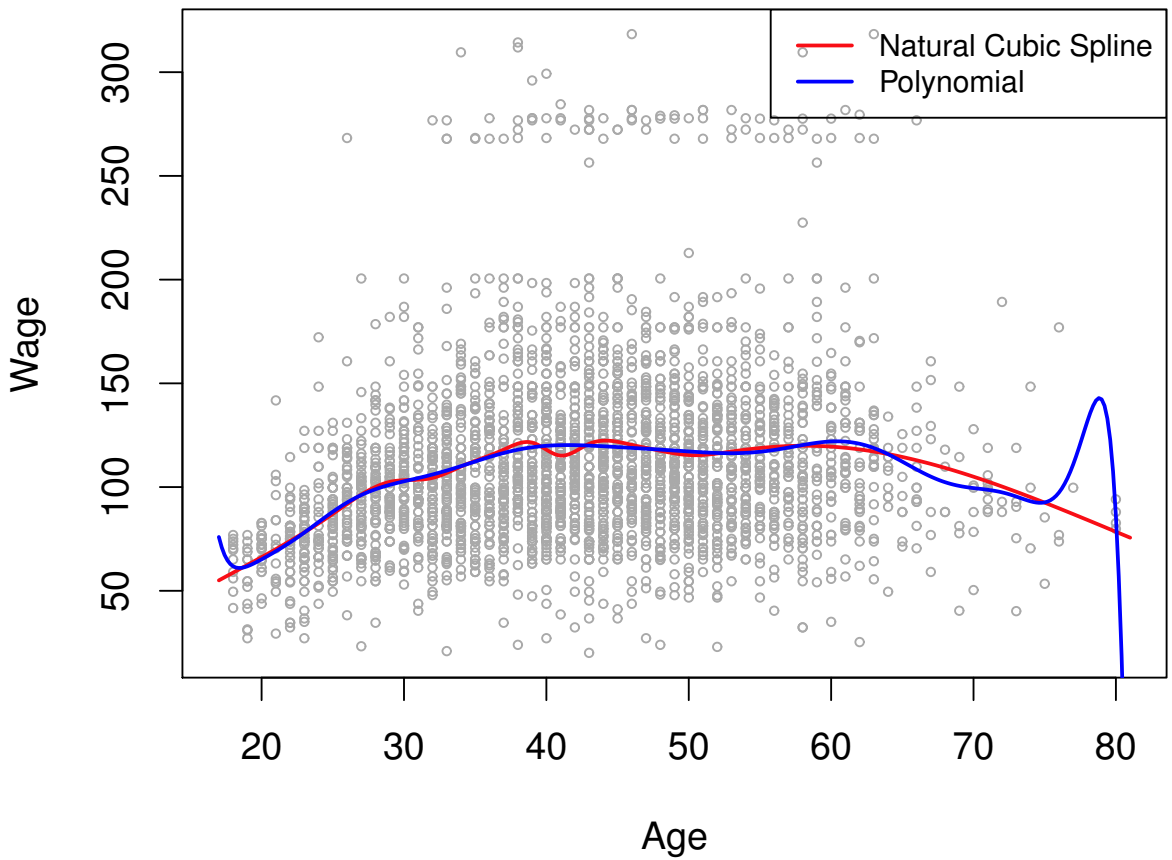
- The locations of the knots are typically quantiles of X.
- The number of knots is chosen by cross validation:





Natural cubic splines vs. poly. regression

- Splines can fit complex functions with few parameters: they are low degree polynomials
- Polynomials require high degree terms to be flexible, which can be unstable at the edges of the dataset.
- Degree 15 polynomial vs. spline with 15 degrees of freedom:



Smoothing spline

- **Regression splines**: choose knots, choose basis function, then least square. No explicit penalty on roughness. Smoothness/roughness is reflected by the basis chosen.
- **Smoothing spline**: a function that makes RSS small, but that is also smooth.

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int f''(x)^2 d(x)$$

- The RSS of the model.
- A penalty for the roughness of the function.

- **Magical fact**: The minimizer $\hat{f}$ for smoothing spline is a natural cubic spline, with knots at each sample point $x_1, \ldots, x_n$
- However, it is NOT the natural cubic spline obtained by "choose knots at the sample points, choose basis function, then least square."
- Instead, obtaining $\hat{f}$ similar to a ridge regression.

- Solution to the smoothing spline problem is a natural cubic spline, which can be written in terms of its basis functions.

$$f(x) = \beta_0 + \beta_1 f_1(x) + \beta_2 f_2(x) + \dots \beta_{n+3} f_{n+3}(x)$$

- Let $\mathbf{N}$ be a matrix with $\mathbf{N}_{ij} = f_j(x_i)$, the objective function of smoothing spline can be written as

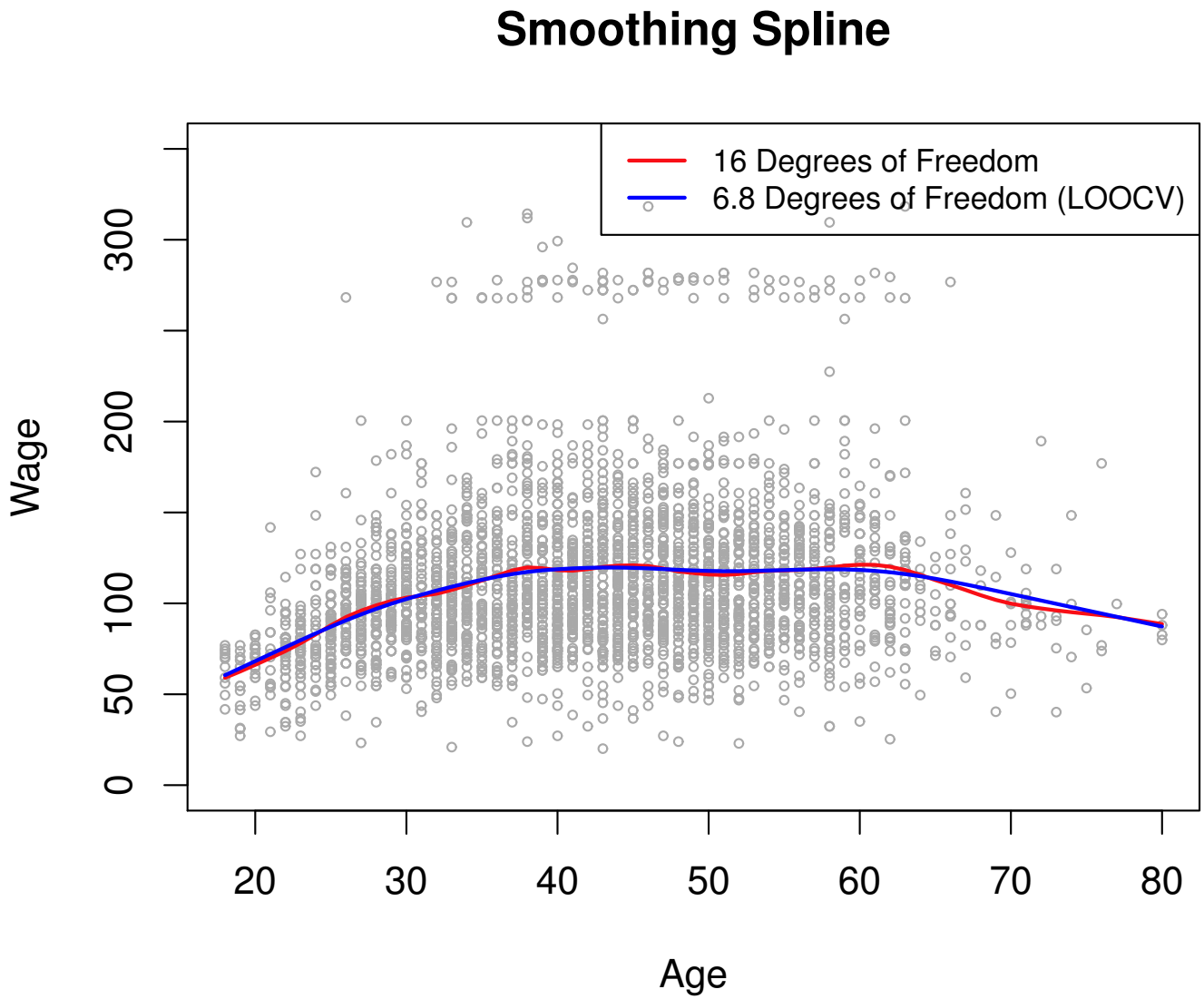
$$(\boldsymbol{y} - \mathbf{N}\boldsymbol{\beta})^T(\boldsymbol{y} - \mathbf{N}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \Omega_{\mathbf{N}} \boldsymbol{\beta}$$

  where $\Omega_{\mathbf{N}}(i,j) = \int f_i''(t) f_j''(t) dt$.
- The solution to the above is

$$\hat{\boldsymbol{\beta}} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega_{\mathbf{N}})^{-1}\mathbf{N}^T\boldsymbol{Y}$$

  (recall ridge regression)
- In-sample predictions are

$$\hat{\boldsymbol{y}} = \underbrace{\mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega_{\mathbf{N}})^{-1}\mathbf{N}^T}_{\mathbf{S}_\lambda}\boldsymbol{Y}$$



λ is chosen by CV or LOOCV (which has a short-cut formula)

Local linear regression

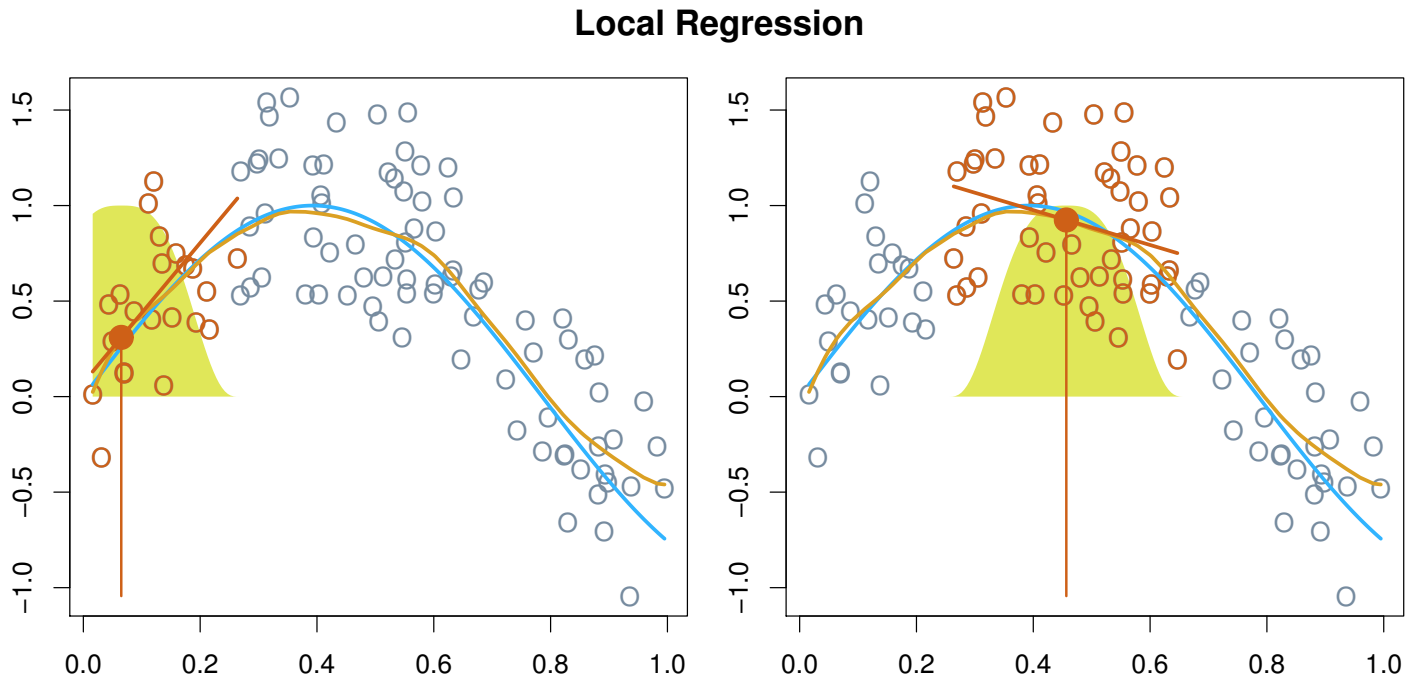


Figure: blue=truth; red=fitted curve

- **KNN regression**: At each point, use constant function fit only to nearest neighbors of that point.
- Local linear regression generalizes KNN regression by assigning weights to training point for each x to be predicted.

To predict the regression function f at an input x:

- Assign a weight K_i to the training point x_i, such that K_i decreases when the distance $d(x, x_i)$ increases.
  - Example: Gaussian kernel

$$K_i = k(x, x_i) = \exp(-(x - x_i)^2/\nu)$$

- Perform a weighted least squares regression:

$$\underset{\beta_0,\beta_1}{\operatorname{argmin}} \sum_{i=1}^{n} K_i(y_i - \beta_0 - \beta_1 x_i)^2$$

- Predict $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$.

Generalized Additive Models (GAMs)

Extension of all non-linear models above to multiple predictors: Turn

$$\texttt{wage} = \beta_0 + \beta_1 \times \texttt{year} + \beta_2 \times \texttt{age} + \beta_3 \times \texttt{education} + \epsilon$$

to

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon$$

If the functions f_k have a basis representation, we can simply use least squares:

- Natural cubic splines
- Polynomials
- Step functions

Backfitting

Otherwise, for smoothing splines and local regression (which have no basis representation), we can use **backfitting**:

- Keep $f_2, \ldots, f_p$ fixed and fit f_1 using the partial residual

$$y_i - \beta_0 - f_2(x_{i2}) - \cdots - f_p(x_{ip})$$

  as the response.

- Keep $f_1, f_3, \ldots f_p$ fixed and fit f_2 using the partial residual

$$y_i - \beta_0 - f_1(x_{i1}) - f_3(x_{i3}) - \cdots - f_p(x_{ip})$$

  as the response.

- ...
- Iterate

Remarks on GAM

- Allows nonlinearity
- Potentially make more accurate predictions
- Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed.
- The smoothness can be summarized via degrees of freedom.
- (Drawback) Additivity assumption may be restrictive. This can be partially addressed by adding key interaction variables $X_i \times X_j$ or $f_{ij}(X_i, X_j)$

Adaptive Trend Filtering

$$y_i = f(x_i) + \epsilon_i$$

Assume $x_i = i/n$.

- The discrete difference operator $D^{(k+1)}$ is defined recursively as:

$$D^{(k+1)} = D^{(1)} \cdot D^{(k)} \in \mathbb{R}^{(n-k)\times n}$$

where $D^{(1)}$ is defined as

$$D^{(1)} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix}$$

straightforward to check that

$$D^{(2)} = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ 0 & 0 & 1 & -2 & \dots & 0 \\ \vdots & & & & & \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} -1 & 3 & -3 & 1 & \dots & 0 \\ 0 & -1 & 3 & -3 & \dots & 0 \\ 0 & 0 & -1 & 3 & \dots & 0 \\ \vdots & & & & & \end{bmatrix}$$

Trend filtering estimate

The kth order trend filtering estimate is defined by

$$\underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|y-\theta\|_2^2 + \frac{n^k}{k!}\lambda \|D^{(k+1)}\theta\|_1$$

- Matrix $D^{(k+1)}$ can be thought of as the discrete analogy to the $(k+1)$st order derivative operator
- Penalty term in penalizes the discrete $(k+1)$st derivative of the vector θ
- When $k = 0$, recall fused lasso.

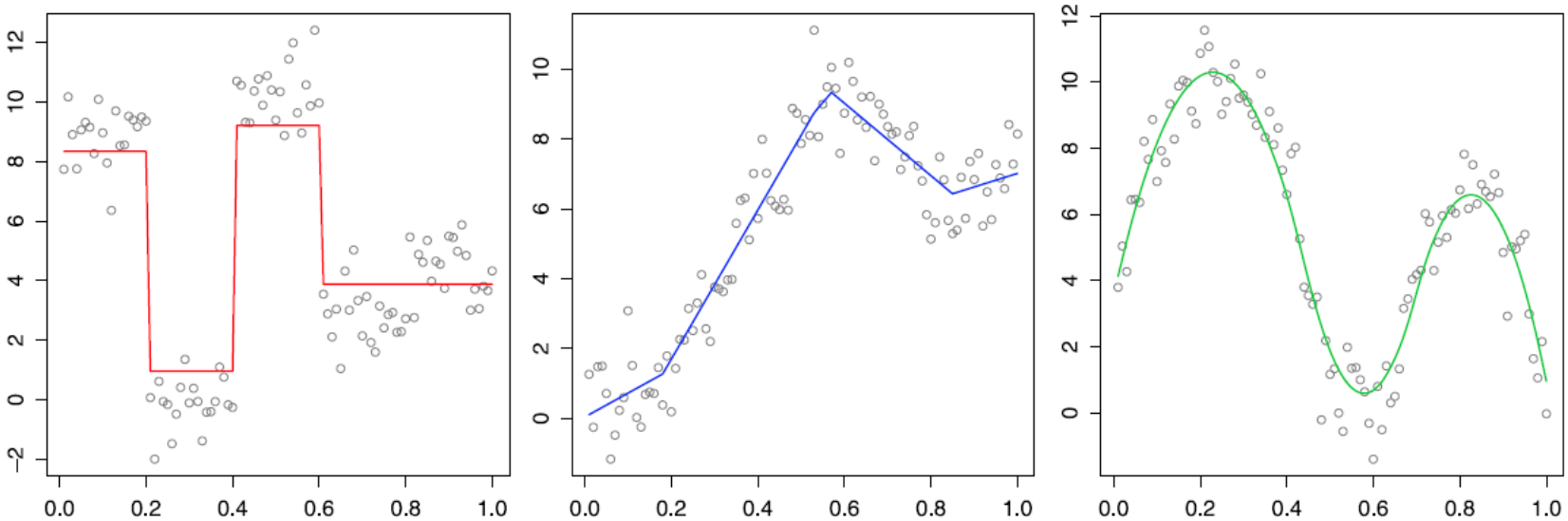
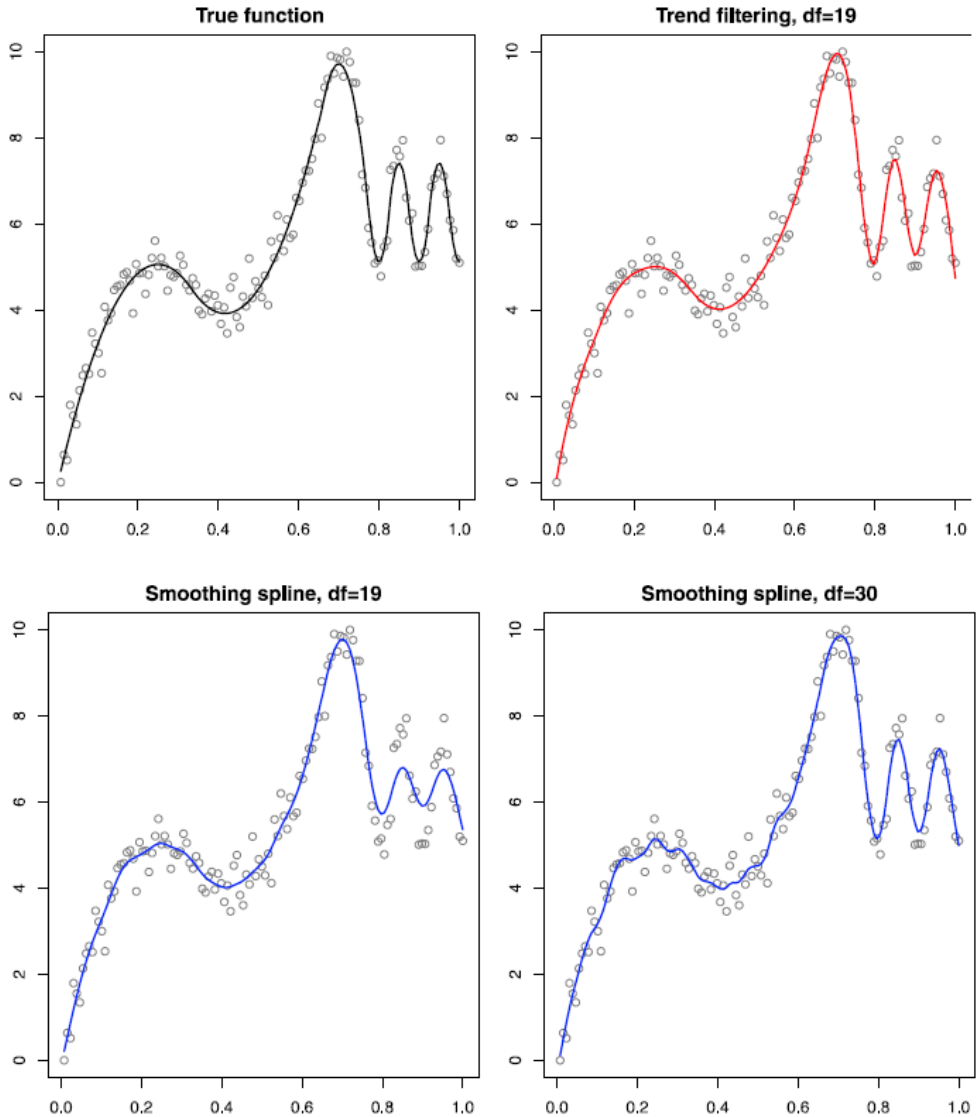


FIG. 1. *Simple examples of trend filtering for constant, linear, and quadratic orders* ($k = 0, 1, 2$, *resp.*), *shown from left to right. Although the trend filtering estimates are only defined at the discrete inputs* $x_i = i/n$, $i = 1, \ldots, n$, *we use linear interpolation to extend the estimates over* $[0, 1]$ *for visualization purposes* (*this is the default for all figures in this paper*).

Comparison to smoothing spline

- **Smoothing Splines**: Not locally adaptive.
- **Locally Adaptive Regression Spline**: Computational Expensive ($O(n^3)$)
- **Adaptive trend filtering**: locally adaptive and computationally cheap
- Smoothing spline utilizes ℓ_2 penalty while trend filtering uses ℓ_1 penalty. Thus later one shrinks some components of $D\theta$ to zero which therefore exhibits a finer degree of local adaptivity.


True function
Trend filtering, df=19
Smoothing spline, df=19
Smoothing spline, df=30
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0