Basis Expansion

Big Data Lectures – Chapter 5

Lexin Li

Division of Biostatistics
University of California, Berkeley

Outline

- list of topics:
  - basis expansion: basics
  - splines
    - linear and cubic splines
    - polynomial regression splines
    - natural cubic splines
    - smoothing splines
    - multidimensional splines
  - generalized additive model
  - wavelet
  - kernel methods

Basis Expansion: Basics

Introduction

- why linear models?
  - convenient and easy to fit
  - easy to interpret
  - the first-order Taylor approximation to $f(\boldsymbol{X}) = E(Y|\boldsymbol{X})$
  - when n is small and/or p is large, linear models do not overfit

Introduction

- why linear models?
  - convenient and easy to fit
  - easy to interpret
  - the first-order Taylor approximation to $f(\boldsymbol{X}) = E(Y|\boldsymbol{X})$
  - when n is small and/or p is large, linear models do not overfit

- **basis expansion:**
  - key idea: augment or replace the original input features with their **transformations**, then fit a linear model in the new space of derived input features
  - a more **flexible** representation:

$$f(\boldsymbol{X}) = \sum_{m=1}^{M} \beta_m h_m(\boldsymbol{X}) = \beta_1 h_1(\boldsymbol{X}) + \beta_2 h_2(\boldsymbol{X}) + \ldots + \beta_M h_M(\boldsymbol{X})$$

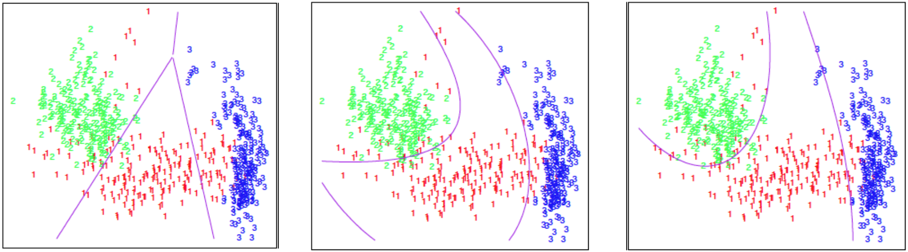
  where $h_m(\cdot)$ are **basis functions**

UNIVERSITY OF CALIFORNIA BERKELEY

Introduction

- example:

LDA on x_1, x_2 | LDA on x_1, x_2 $x_1 x_2, x_1^2, x_2^2$ | QDA on x_1, x_2



Introduction

still fit a linear regression model but after fitting a function on the predictors

- more examples:
  - generalized additive model:

$$f(\boldsymbol{X}) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

  - projection pursuit model:

$$f(\boldsymbol{X}) = \beta_0 + \sum_{m=1}^{M} \beta_m \, \sigma(\alpha_{m0} + \boldsymbol{\alpha}_m^{\mathsf{T}} \boldsymbol{X})$$

restrictive, only considers a linear combination of the predictors

Introduction

- more examples:
  - generalized additive model:

$$f(\boldsymbol{X}) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

  - projection pursuit model:

$$f(\boldsymbol{X}) = \beta_0 + \sum_{m=1}^{M} \beta_m \, \sigma(\alpha_{m0} + \boldsymbol{\alpha}_m^{\mathsf{T}} \boldsymbol{X})$$

- key components:
  - **dictionary**: a collection of very large number of basis functions
  - **complexity control**:
    - restriction method: use a (small) number of pre-specified transformation functions; e.g., splines
    - regularized selection: use the entire dictionary but restrict the coefficients through regularization; e.g., wavelet
    - implicit basis transformation: kernel methods

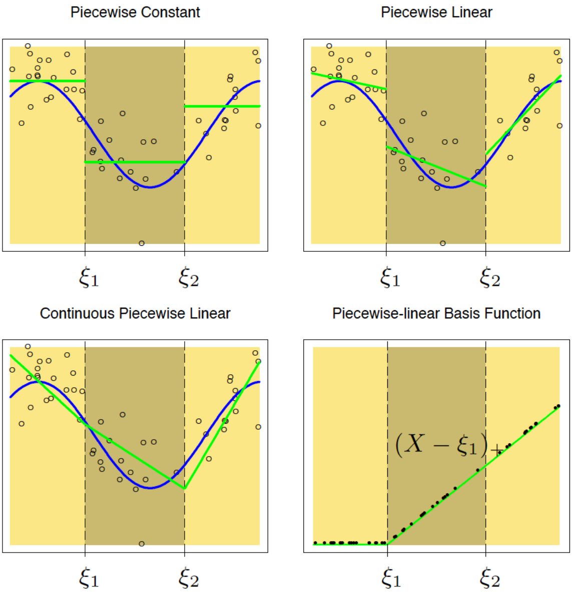
Splines

Splines

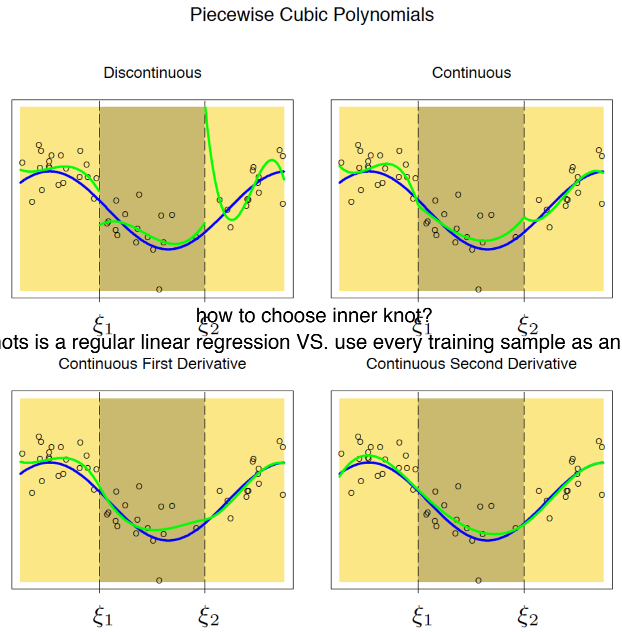
- what are splines:
  - **piecewise polynomial functions**
  - divide the domain of $\boldsymbol{X}$ into continuous intervals and fit separate polynomials in each interval
- examples:
  - linear splines, cubic splines, B-splines, natural cubic splines, smoothing splines, ...
- knots: assume the range of x is $[a, b]$. Let the points

$$a < \xi_1 < \xi_2 < \cdots < \xi_K < b$$

  be a partition of the interval $[a, b]$
  - call $\{\xi_1, ..., \xi_K\}$ the interior knots
  - call $\{\xi_0 = a, \xi_{K+1} = b\}$ the boundary knots. It is possible $a = -\infty$ and $b = \infty$
  - fixed knots: often use quantiles of x


Piecewise Constant
ξ_1
ξ_2
Piecewise Linear
ξ_1
ξ_2
Continuous Piecewise Linear
ξ_1
ξ_2
Piecewise-linear Basis Function
$(X-\xi_1)_+$
ξ_1
ξ_2

how many predictors are we looking at? Just one predictor (and one response variable) this is too simple to be useful



how to choose inner knot?

0 inner knots is a regular linear regression VS. use every training sample as an inner knot

Cubic splines

- piecewise constant, piecewise linear function
- piecewise cubic function: for each interval $[\xi_j, \xi_{j+1}), j = 0, ..., K$,

$$f(x) = \beta_{0j} + \beta_{1j}x + \beta_{2j}x^2 + \beta_{3j}x^3$$

- having continuous first and second derivatives at the interior knots,

$$\begin{aligned} f(\xi_j^-) &= f(\xi_j^+) \\ f'(\xi_j^-) &= f'(\xi_j^+) \\ f''(\xi_j^-) &= f''(\xi_j^+), \quad j = 1, ..., K \end{aligned}$$

- what is the total degrees of freedom? $K + 4$

$$\begin{aligned} (K+1) \text{ regions} &\times (4 \text{ parameters per region}) \\ -K \text{ knots} &\times (3 \text{ constraints per knots}) \end{aligned}$$

Cubic splines

- a more direct way to take care of continuity constraints: use a basis that is piecewise polynomial and continuous at knots
- truncated power series basis:
  - piecewise linear with continuity at knots

$$h_1(x) = 1, h_2(x) = x, h_3(x) = (x - \xi_1)_+, \cdots, h_{K+2}(x) = (x - \xi_K)_+.$$

  - piecewise cubic with continuous first and second derivatives at knots

$$1, x, h_3(x) = x^2, h_4(x) = x^3, h_5(x) = (x - \xi_1)_+^3, \cdots, h_{K+4} = (x - \xi_K)_+^3.$$

for cubic splines,

$$\text{Model space} = \text{span}\{h_1(x), ..., h_{K+4}(x)\}$$

Cubic splines

- polynomial regression splines with order M
  - K fixed (interior) knots: $\xi_1, \cdots, \xi_K$
  - piecewise polynomial of order $M-1$; continuous derivatives up to order $M-2$
- the truncated power series basis functions are

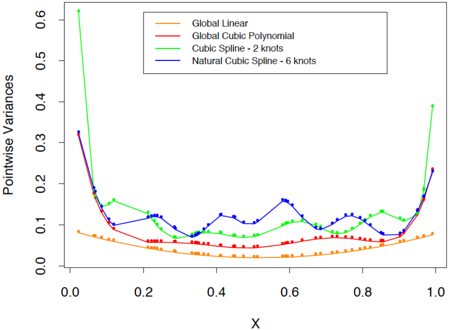
$$h_j(x) = x^{j-1}, \quad j = 1, \cdots, M; \quad h_{M+l}(x) = (x - \xi_l)_+^{M-1}, \quad l = 1, \cdots, K.$$

- total degrees of freedom is $df = K + M$
- popular choices of $M = 1, 2, 4$:
  - piecewise constant is order-1 spline
  - piecewise linear with continuity is order-2 spline
  - cubic spline is order-4 spline

cubic splines are the lowest-order spline for which knot-discontinuity is not visible to human eyes.

Natural cubic splines

- boundary effects:
  - the behavior of polynomials fit tends to be erratic near the boundaries
  - the polynomials fit beyond the boundary knots behave more wildly than the corresponding global polynomials in that region
- check the point-wise variance of spline function fits by least squares
  - In general, the variance near the boundary is large for all spline fits
  - cubic spline has the worst (largest) point-wise variance near the boundary



Natural cubic splines

- natural cubic splines: add additional constraint such that the function is linear beyond the two boundary knots
- remarks:
  - frees up four degrees of freedom, which can be spent more profitably by putting more knots in the interior region
  - the price is the bias near two boundaries
  - since there is less information, assuming the function being linear near the boundaries is reasonable
- degrees of freedom: $K + 4 - 4 = K$ (same as the number of knots)
- basis functions:

$$N_1(x) = 1, N_2(x) = x, N_{k+2}(x) = d_k(x) - d_{k-1}(x),$$

where

$$d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_K)_+^3}{\xi_K - \xi_k}, \quad k = 1, \cdots, K - 2,$$

each of these basis functions have zero second and third derivatives for $x > \xi_K$.

Example – South African heart disease

- South African heart disease – coronary risk-factor study (CORIS) baseline survey
  - rural areas in Western Cape, South Africa (high-incidence region)
  - to establish the intensity of ischemic heart disease risk factors

- data set
  - white males between age 15-64
  - 160 cases and 302 controls
  - Y = the presence of absence of myocardial infraction (MI) at the time of the survey (the overall prevalence of MI was 5.1% in that region)

- natural cubic spline fitting model:

$$\log[\mathrm{Pr}(chd)] = \theta_0 + h_1(X_1)^\intercal\theta_1 + \cdots + h_p(X_p)^\intercal\theta_p,$$

θ_j is the vector of coefficients of natural spline basis functions h_j.

Example – South African heart disease

- derived input features:
  - X_1=systolic blood pressure
    $h_1(X_1)$ = a basis consisting of four basis functions
  - similarly for variables: tobacco, ldl, obesity, age
  - X_4=family history (two-level factor): dummy; single coefficient

- spline model fit: for each variable j,

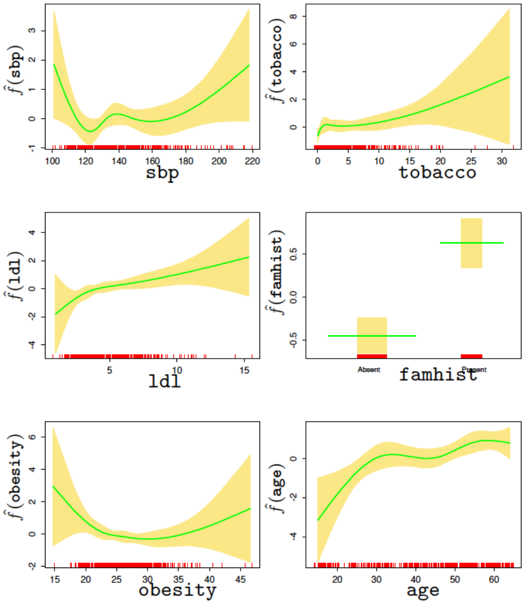
$$\hat{f}_j(X_j) = h_j(X_j)^\mathsf{T}\hat{\theta}_j$$

  - the variance of parameter estimator

$$\widehat{\mathrm{cov}}(\hat{\theta}) = \hat{\Sigma} = (H^T WH)^{-1}$$

    W the diagonal weight matrix from logistic regression, $w_j = p_j(1 - p_j)$
  - the pointwise variance function of $\hat{f}_j$

$$\mathrm{var}[\hat{f}(X_j)] = h_j(X_j)^\mathsf{T}\hat{\Sigma}_{jj}h_j(X_j)$$

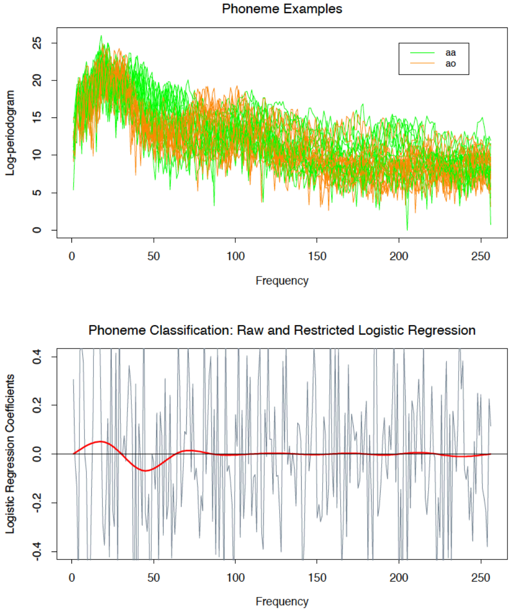

f̂(sbp)
sbp
100 120 140 160 180 200 220
f̂(tobacco)
tobacco
0 5 10 15 20 25 30
f̂(ldl)
ldl
5 10 15
f̂(famhist)
famhist
Absent
Present
f̂(obesity)
obesity
15 20 25 30 35 40 45
f̂(age)
age
20 30 40 50 60

Example – phoneme recognition

- phoneme recognition
  - two phonemes "aa" and "ao" measured at 256 frequencies
  - to classify a spoken phoneme
- data set
  - 695 "aa" and 1022 "ao"
  - functional modeling

$$\log \frac{\Pr(aa|\boldsymbol{X})}{\Pr(ao|\boldsymbol{X})} = \int X(t)\beta(t)dt$$

- approximate by an unrestricted logistic regression $\sum_{j=1}^{256} X_j\beta_j$
- smooth regularization via natural cubic splines
  - represent $\beta(t)$ as an expansion of splines $\beta(t) = \sum_{m=1}^{M} h_m(t)\theta_m$
  - $\boldsymbol{\beta} = \boldsymbol{H}\boldsymbol{\theta}$, where $\boldsymbol{H}$ is a $p \times M$ natural cubic splines basis matrix, with knots uniformly placed on $1, 2, \ldots, 256$
  - replace the input features $\boldsymbol{X}$ with its filtered version $\boldsymbol{X}^* = \boldsymbol{H}^\mathsf{T}\boldsymbol{X}$
    $\hat{\beta}(t) = h(t)^\mathsf{T}\hat{\theta}$


Phoneme Examples
aa
ao
Log-periodogram
0
5
10
15
20
25
0
50
100
150
200
250
Frequency
Phoneme Classification: Raw and Restricted Logistic Regression
Logistic Regression Coefficients
-0.4
-0.2
0.0
0.2
0.4
0
50
100
150
200
250
Frequency

Smoothing splines

- motivation: use all observations as knots, so avoid knot selection
- solve:

$$\min_{f \in W_2[a,b]} \frac{1}{n} \sum_{i=1}^{n} [y_i - f(x_i)]^2 + \lambda \int_a^b [f''(x)]^2 dx$$

  - first term measures the closeness/loyalty of the model to the data; related to the bias
  - second term penalizes the roughness/curvature of the function; related to the variance of the estimate; the **roughness penalty** (regularization)
  - $\lambda = 0$: point-wise interpolation; $\lambda = \infty$: least squares line ($f''(x) = 0$)
- solution:
  - natural spline basis: $f(x) = \sum_{j=1}^{n} N_j(x)\theta_j$
  - fitted values:

$$\hat{\boldsymbol{f}} = \boldsymbol{N}(\boldsymbol{N}^\top \boldsymbol{N} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{N}^\top \boldsymbol{y} = \boldsymbol{S}_\lambda \boldsymbol{y}$$

where $\boldsymbol{N}_{ij} = N_j(\boldsymbol{x}_i), \boldsymbol{\Omega}_{jk} = \int N_j^{''}(t) N_k^{''}(t) dt$, and $\boldsymbol{N}, \boldsymbol{\Omega} \in \mathbb{R}^{n \times n}$

Multidimensional splines

- tensor product basis
  - so far, we have focused on $X \in \mathbb{R}^1$
  - suppose $X \in \mathbb{R}^2$:
    - a set of basis $h_{1k}(X_1), k = 1, .., M_1$, for functions of coordinate X_1
    - a set of basis $h_{2k}(X_2), k = 1, .., M_2$, for functions of coordinate X_2

  the $M_1 \times M_2$ dimensional **tensor product basis** is

$$g_{jk}(X_1, X_2) = h_{1j}(X_1)h_{2k}(X_2), \quad j = 1, ..., M_1, \quad k = 1, ..., M_2$$

  - then represent a two-dimensional function as

$$g(X_1, X_2) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X_1, X_2)$$

- thin-plate splines via regularization

Generalized Additive Model

Generalized additive model

- generalized additive models (GAM):

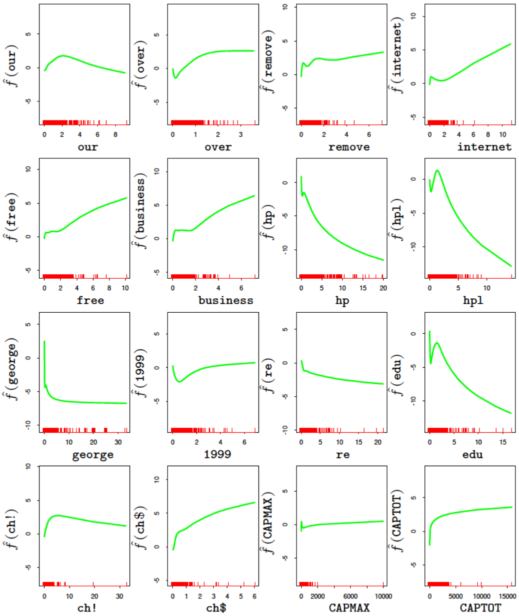
$$g\left(\mu(\boldsymbol{X})\right) = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

  - $g(\mu) = \mu$, the identity link for Gaussian response
  - $g(\mu) = \log\{\mu/(1-\mu)\}$, the logit link for binary response
  - $g(\mu) = \Phi^{-1}(\mu)$, the probit link for binary response, where Φ is the Gaussian cumulative distribution function
  - $g(\mu) = \log(\mu)$, the log-linear link for Poisson count response
- remarks:
  - the key idea is to replace each predictor (also, a linear identity function of the predictor) with a flexible function of the predictor that may identify and characterize nonlinear regression effects
  - provide a useful extension of linear models, making them more flexible, while still retaining much of their interpretability
  - fit each f_j using a scatterplot smoother: cubic smoothing spline or kernel smoother (this is a 1-dimensional regression)
  - can have limitations when p is very large

Example – email spams prediction

- spam email data:
  - screen email for "spam" (junk email): $Y = 1/0$ if a spam/not spam
  - totally 4601 email messages, randomly divided to a training data of 3605 and a testing data of 1536
  - $p = 57$ predictors: 54 quantitative percentage of words/characters in the email matching a given word (e.g., "free", "business", "george") or character (e.g., “$"), the average/max/sum of length of uninterrupted sequences of capital letters
- a GAM fit:
  - most predictors have a very long tails; log-transform each variable
  - use a cubic smoothing spline with $df = 4$ for each predictor
- classification result:

	Prediction	
True	email (0)	spam (1)
email (0)	58.3%	2.5%
spam (1)	3.0%	36.3%


$\hat{f}$(our)
our
$\hat{f}$(over)
over
$\hat{f}$(remove)
remove
$\hat{f}$(internet)
internet
$\hat{f}$(free)
free
$\hat{f}$(business)
business
$\hat{f}$(hp)
hp
$\hat{f}$(hpl)
hpl
$\hat{f}$(george)
george
$\hat{f}$(1999)
1999
$\hat{f}$(re)
re
$\hat{f}$(edu)
edu
$\hat{f}$(ch!)
ch!
$\hat{f}$(ch$)
ch$
$\hat{f}$(CAPMAX)
CAPMAX
$\hat{f}$(CAPTOT)
CAPTOT

Generalized additive model

- model fitting: **backfitting**
  - the key idea is to fit one $f_j(X_j)$ at a time
  - the building block is the scatterplot smoother for fitting nonlinear effects in a flexible way
- additive regression model:

$$\sum_{i=1}^{n}\left\{y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij})\right\}^2 + \sum_{j=1}^{p}\lambda_j \int f_j''(t_j)^2 dt_j$$

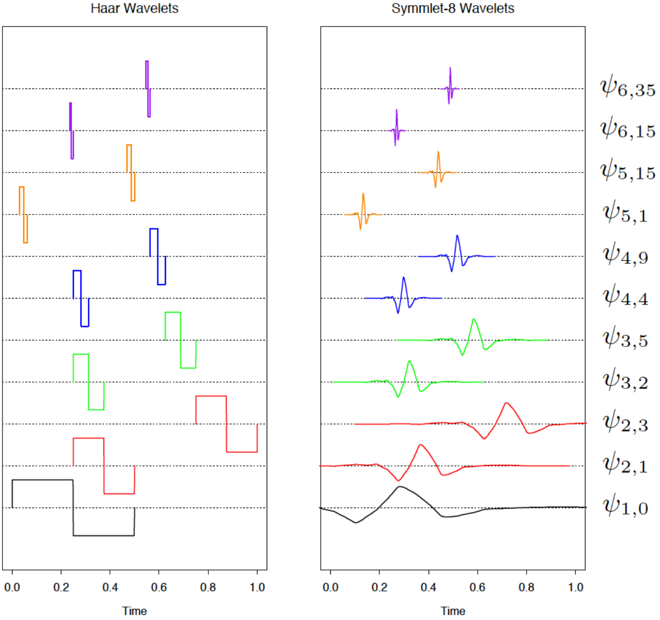
the backfitting algorithm:

initialize: $\hat{\alpha} = \bar{y}, \hat{f}_j = 0$
repeat
  for $j = 1, \ldots, p$ **do**
    $\hat{f}_j = S_j\left[\left\{y_i - \hat{\alpha} - \sum_{k\neq j}\hat{f}_k(x_{ik})\right\}_1^n\right]$
  end for
until $\hat{f}_j$ changes less than a threshold

Wavelet

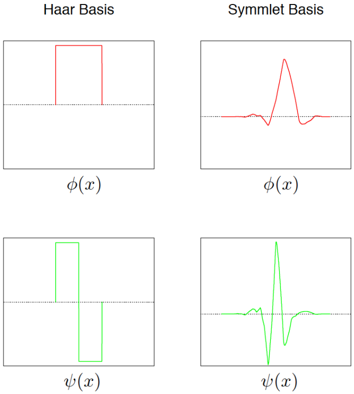
Wavelet

- wavelet basis:
  - use a set of **complete orthonormal** basis functions
  - shrink and select the coefficients towards a **sparse** representation
- applications:
  - very popular in signal processing and compression
  - capable of representing both smooth and locally bumpy functions in an efficient way
- some popular wavelet basis:
  - Harr wavelets (simpler)
  - Daubechies symmlet-8 wavelets (smoother)
- good statistical properties:
  - adapt for spatially inhomogeneous curves
  - nearly minimax (rate) for a large class of functions with unknown degrees of smoothness
- a very brief glimpse of this vast and growing field


Haar Wavelets
Symmlet-8 Wavelets
$\psi_{6,35}$
$\psi_{6,15}$
$\psi_{5,15}$
$\psi_{5,1}$
$\psi_{4,9}$
$\psi_{4,4}$
$\psi_{3,5}$
$\psi_{3,2}$
$\psi_{2,3}$
$\psi_{2,1}$
$\psi_{1,0}$
0.0 0.2 0.4 0.6 0.8 1.0
Time
0.0 0.2 0.4 0.6 0.8 1.0
Time

Wavelet

- Haar wavelet construction:
  - father wavelet: $\phi(x) = I(x \in [0, 1])$, $\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)$ ("grand mean", "rough")
  - mother wavelet: $\psi(x) = \phi(2x) - \phi(2x - 1)$, $\psi(x) = 2^{j/2}\psi(2^j x - k)$ ("contrast", "detail")



Wavelet

- adaptive wavelet filtering:
  - wavelets are particularly useful when the data are measured on a uniform lattice, such as a discretized signal, image, or time series
  - Stein Unbiased Risk Estimation (SURE) Donoho and Johnstone (1994)

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{\theta}\|_2^2 + 2\lambda\|\boldsymbol{\theta}\|_1$$

    where $\boldsymbol{y}$ is the response vector, and $\boldsymbol{W}$ is the $n \times n$ orthonormal wavelet basis matrix evaluated at the n uniformly spaced observations
  - the least squares coefficients are translated toward zero, and truncated at zero

$$\hat{\theta}_j = \mathrm{sign}(y_j^*)(|y_j^*| - \lambda)_+$$

    where $\boldsymbol{y}^* = \boldsymbol{W}^\top \boldsymbol{y}$ is the wavelet transform of $\boldsymbol{y}$
- the basis are hierarchically structured from coarse to detailed
  the L_1 penalty does both shrinkage and selection

Kernel Methods

Kernel methods

- kernel methods
  - extremely popular in machine learning literature
  - powerful, flexible, $n < p$, ...
  - we do **not** mean **kernel smoothing** here
- reproducing kernel Hilbert space (RKHS)
- some representative kernel methods:
  - **kernel (nonlinear) support vector machine**
  - kernel least squares and kernel logistic regression
  - kernel principal component analysis
  - can **"kernelize"** many learning methods ...
- some challenges:
  - develop appropriate kernel functions
  - variable selection for the kernel methods

Reproducing kernel Hilbert space

- space:
  - a **Hilbert space** is an infinite dimensional Euclidean space; it is a vector space (i.e., is closed under addition and scalar multiplication, obeys the distributive and associative laws, etc.); it is also endowed with an **inner product** $\langle \cdot, \cdot \rangle$
  - a **reproducing kernel Hilbert space** is, conceptually, a "smaller" Hilbert space that contains restricted, smooth functions

- kernel:
  - **kernel function**: $k(\boldsymbol{x}, \boldsymbol{x}')$
  - **Gram matrix** $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$: $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$
  - k is a positive definite kernel if its Gram matrix is positive definite
  - **reproducing kernel map**: $\Phi : \boldsymbol{x} \to k(\cdot, \boldsymbol{x})$

- commonly used kernels:
  - dth degree polynomial: $k(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^\mathsf{T} \boldsymbol{x}')^d$
  - Gaussian radial basis: $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$
  - neural network: $k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\kappa_1 \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \kappa_2)$

Reproducing kernel Hilbert space

- construction:
  - consider the space of functions generated by all linear combinations of the functions $k(\cdot, \boldsymbol{x})$:
  $$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, \boldsymbol{x}_i)$$
  - define an inner product: let $g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\cdot, \boldsymbol{x}_j')$, and
  $$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j k(\boldsymbol{x}_i, \boldsymbol{x}_j')$$
- properties:
  - the representer of evaluation:
  $$\langle k(\cdot, \boldsymbol{x}), f \rangle = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x})$$
  - the reproducing property:
  $$\langle k(\cdot, \boldsymbol{x}), k(\cdot, \boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}')$$

Kernel methods

- a general optimization problem with regularization:

$$\min_{f \in \mathcal{H}} \left[\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) + \lambda J(f) \right]$$

  - $L(y, f(\boldsymbol{x}))$ is a point-wise loss function, $J(f)$ is a penalty functional, and $\mathcal{H}$ is a space of candidate functions

- an important subclass of problems:

$$\min_{f \in \mathcal{H}_\mathcal{K}} \left[\sum_{i=1}^{n} L(y_i, f(\boldsymbol{x}_i)) + \lambda \|f\|^2_{\mathcal{H}_\mathcal{K}} \right]$$

  - narrow the search to a subclass of functions in a RKHS

Kernel methods

- **the representer theorem**: the minimizer has the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

  - basis expansion with basis function: $h_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$
  - solution hinges on estimating $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^{\mathsf{T}} \in \mathbb{R}^n$

- the equivalent optimization problem:

$$\min_{\boldsymbol{\alpha}} \left[L(\boldsymbol{y}, \boldsymbol{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{K} \boldsymbol{\alpha} \right]$$

  - thanks to the reproducing property:

$$J(f) = \sum_{i=1}^{n} \sum_{j=1}^{n} k(\boldsymbol{x}_i, \boldsymbol{x}_j) \alpha_i \alpha_j$$

  - optimization is now over a **finite dimensional** $\boldsymbol{\alpha} \in \mathbb{R}^n$

Kernel methods

- **transformed predictor view:**
  - $k(\cdot, \boldsymbol{x}_i)$ acts as basis function
  - recall the reproducing kernel map $\Phi : \boldsymbol{x} \rightarrow k(\cdot, \boldsymbol{x})$, $\Phi(\boldsymbol{x})$ can be viewed as a transformation of the original feature vector $\boldsymbol{x}$
- **kernel trick**

$$\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle = \langle k(\cdot, \boldsymbol{x}), k(\cdot, \boldsymbol{x}') \rangle = k(\boldsymbol{x}, \boldsymbol{x}')$$

- an example:
  - a degree-2 (quadratic) polynomial kernel with $p = 2$:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}') &= (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^2 \\ &= (1 + x_1 x_1' + x_2' x_2')^2 \\ &= 1 + 2x_1 x_1' + 2x_2' x_2' + (x_1 x_1')^2 + (x_2' x_2')^2 + 2x_1 x_1' x_2' x_2' \end{aligned}$$

  - a set of 6 basis functions:

$$\begin{array}{lll} \phi_1(\boldsymbol{x}) = 1 & \phi_2(\boldsymbol{x}) = \sqrt{2}x_1 & \phi_3(\boldsymbol{x}) = \sqrt{2}x_2 \\ \phi_4(\boldsymbol{x}) = x_1^2 & \phi_5(\boldsymbol{x}) = x_2^2 & \phi_6(\boldsymbol{x}) = \sqrt{2}x_1 x_2 \end{array}$$

  - $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$

Kernel support vector machine

- linear support vector machine:

$$\min_{\beta_0, \boldsymbol{\beta}} \left[\frac{1}{n} \sum_{i=1}^{n} \{1 - y_i(\beta_0 + \boldsymbol{\beta}^\mathsf{T} \boldsymbol{x}_i)\}_+ + \frac{\lambda}{2} \|\boldsymbol{\beta}\|^2 \right]$$

  - $\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$
  - $f(\boldsymbol{x}) = \boldsymbol{x}^\mathsf{T} \boldsymbol{\beta} + \beta_0$

- kernel support vector machine:

$$\min_{f \in \mathcal{H}_\mathcal{K}} \left[\frac{1}{n} \sum_{i=1}^{n} \{1 - y_i f(\boldsymbol{x}_i)\}_+ + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_\mathcal{K}} \right]$$

or "equivalently", transformed predictor / basis expansion $\Phi(\boldsymbol{x})$

$$\min_{\beta_{0\phi}, \boldsymbol{\beta}_\phi} \left[\frac{1}{n} \sum_{i=1}^{n} \{1 - y_i(\beta_{0\phi} + \boldsymbol{\beta}_\phi^\mathsf{T} \Phi(\boldsymbol{x}_i))\}_+ + \frac{\lambda}{2} \|\boldsymbol{\beta}_\phi\|^2 \right]$$

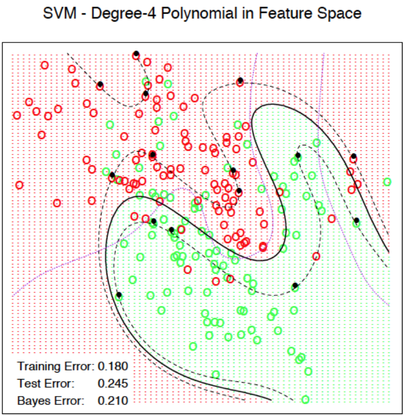
Kernel support vector machine

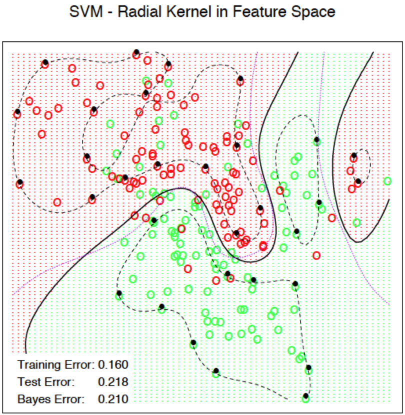
- kernel support vector machine: (cont'd)
  - $\boldsymbol{\beta}_\phi = \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{x}_i)$
  - $f(\boldsymbol{x}) = \Phi(\boldsymbol{x})^\mathsf{T} \boldsymbol{\beta}_\phi + \beta_{0\phi}$, then

$$\begin{aligned} f(\boldsymbol{x}) &= \Phi(\boldsymbol{x})^\mathsf{T} \sum_{i=1}^n \alpha_i y_i \Phi(\boldsymbol{x}_i) + \beta_{0\phi} \\ &= \sum_{i=1}^n \alpha_i y_i \left\langle \Phi(\boldsymbol{x}),\ \Phi(\boldsymbol{x}_i) \right\rangle + \beta_{0\phi} \end{aligned}$$

  - all we need to know is: $\langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_i) \rangle = k(\boldsymbol{x}, \boldsymbol{x}_i), i = 1, \ldots, n$
    do not need to know anything about the actual $\Phi(\cdot)$
  - can deal with $p >> n$

Kernel support vector machine





Additional readings

- Hastie, H., Tibshirani, R., and Friedman, J. (2001). *Elements of Statistical Learning*. Springer. Chapters 5, 9, 12