CS 412 Introduction to Machine Learning

Regression

Instructor: Wei Tang

Department of Computer Science
University of Illinois at Chicago
Chicago IL 60607

https://tangw.people.uic.edu
tangw@uic.edu

Slides credit: Sargur N. Srihari

Simple Regression Problem

- Begin discussion on ML by introducing a simple regression problem
  - It motivates a no. of key concepts
- Problem:
  - Observe input variable x
  - Use x to predict real-valued target variable t
- We consider an artificial example using synthetically generated data
  - Because we know the process that generated the data, it can be used for comparison against a learned model

Correct a typo

Eigendecomposition

$$\begin{cases} A v^{(1)} = \lambda_1 v^{(1)} \\ \vdots \\ A v^{(n)} = \lambda_n v^{(n)} \end{cases}$$

$A \in \mathbb{R}^{n \times n}$ $\quad v^{(i)} \in \mathbb{R}^n$ $\quad \lambda_i \in \mathbb{R}$

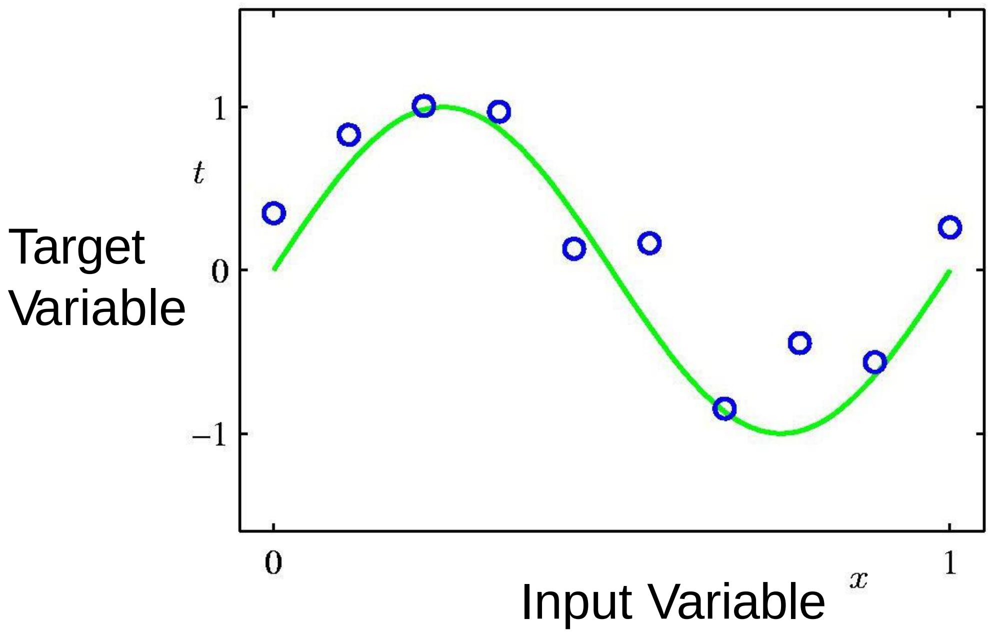
$$\begin{aligned} AV &= A[v^{(1)}, \dots, v^{(n)}] \\ &= [Av^{(1)}, \dots, Av^{(n)}] \\ &= [\lambda_1 v^{(1)}, \dots, \lambda_n v^{(n)}] \\ &= [v^{(1)}, \dots, v^{(n)}] \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} \\ &= V \operatorname{diag}(\lambda) \end{aligned}$$

$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

$$\Rightarrow A = V \operatorname{diag}(\lambda) V^{-1}$$

Synthetic Data for Regression

- Data generated from the function $\sin(2\pi\ x)$
  - Where x is the input
- Random noise in target values



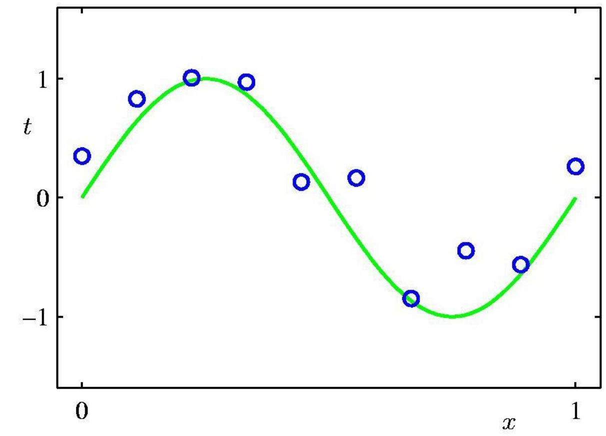
Input values $\{x_n\}$ generated uniformly in range $(0,1)$. Corresponding target values $\{t_n\}$ Obtained by first computing corresponding values of $\sin\{2\pi x\}$ then adding random noise with a Gaussian distribution with std dev 0.3

Training Set

- N observations of x

  $\mathrm{x} = (x_1,..,x_N)^T$

  $\mathrm{t} = (t_1,..,t_N)^T$



Data Generation:
N = 10
Spaced uniformly in range [0,1]
Generated from $\sin(2\pi x)$ by adding small Gaussian noise
Noise typical due to unobserved variables

- Goal is to exploit training set to predict an output value for some new input value

A Simple Approach to Curve Fitting

- Fit the data using a *polynomial function*

$$y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + .. + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

  – where M is the order of the polynomial
- Is higher value of M better? We ll see shortly!
- Coefficients $w_0, \ldots w_M$ are collectively denoted by vector $\mathbf{w}$
- It is a nonlinear function of x, but a linear function of the unknown parameters $\mathbf{w}$
- Have important properties and are called Linear Models

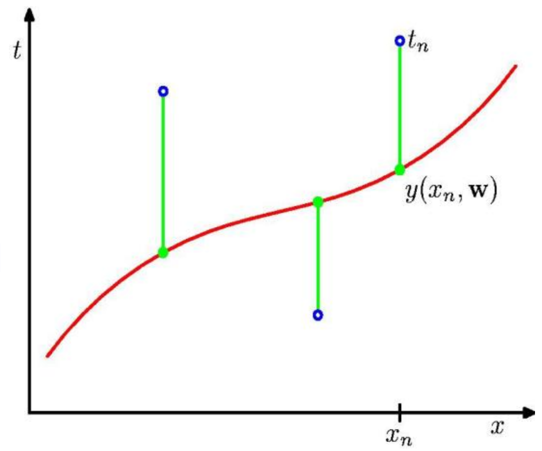
Error Function

- We can obtain a fit by minimizing an error function
  – Sum of squares of the errors between the predictions $y(x_n,\mathbf{w})$ for each data point x_n and target value t_n

$$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathrm{w}) - t_n\}^2$$

  – Factor ½ included for later convenience

Red line is best polynomial fit



- Solve by choosing value of w for which small as possible

Minimization of Error Function

- Error function is a quadratic in coefficients w
- Thus derivative with respect to coefficients will be linear in elements of w
- Thus error function has a unique solution which can be found in closed form
  - Unique minimum denoted w*
- Resulting polynomial is $y(x, \mathrm{w}^*)$

$$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathrm{w}) - t_n\}^2$$

Since $y(x, \mathrm{w}) = \sum_{j=0}^{M} w_j x^j$

$$\frac{\partial E(\mathrm{w})}{\partial w_i} = \sum_{n=1}^{N}\{y(x_n, \mathrm{w}) - t_n\}x_n^i$$

$$= \sum_{n=1}^{N}\{\sum_{j=0}^{M} w_j x_n^j - t_n\}x_n^i$$

Setting equal to zero

$$\sum_{n=1}^{N}\sum_{j=0}^{M} w_j x_n^{i+j} = \sum_{n=1}^{N} t_n x_n^i$$

Set of M+1 equations (i=0,..,M) over M+1 variables are solved to get elements of w*

Solving Simultaneous equations

- $Aw = b$

A is $(M+1) \times (M+1)$
w is $(M+1) \times 1$
b is $(M+1) \times 1$

Can be solved via matrix inversion
or Gaussian elimination

Solving Linear Equations

$$\begin{matrix} a_{11}x_1 + & a_{12}x_2 + \cdots + & a_{1n}x_n = b_1 \\ a_{21}x_1 + & a_{22}x_2 + \cdots + & a_{2n}x_n = b_2 \\ \vdots & \vdots & \vdots \quad \vdots \\ a_{m1}x_1 + & a_{m2}x_2 + \cdots + & a_{mn}x_n = b_m \end{matrix}$$

$$x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

1. Matrix Formulation: $A\mathbf{x}=\mathbf{b}$
Solution: $\mathbf{x}=A^{-1}\mathbf{b}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Here $m=n=M+1$

2. Gaussian Elimination followed by back-substitution

$$\begin{aligned} x + 3y - 2z &= 5 \\ 3x + 5y + 6z &= 7 \\ 2x + 4y + 3z &= 8 \end{aligned}$$

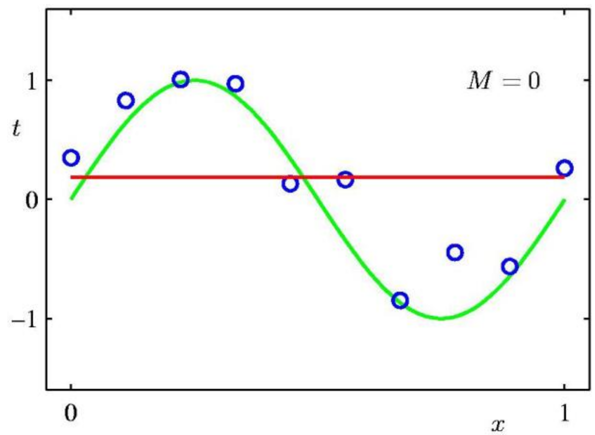
$L_2\text{-}3L_1 \rightarrow L_2$ $\qquad$ $L_3\text{-}2L_1 \rightarrow L_3$ $\qquad$ $\text{-}L_2/4 \rightarrow L_2$

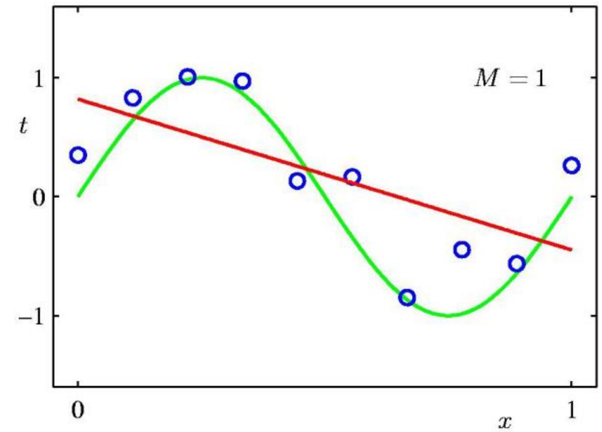
$$\left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 3 & 5 & 6 & 7 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 2 & 4 & 3 & 8 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & -4 & 12 & -8 \\ 0 & -2 & 7 & -2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & -2 & 7 & -2 \end{array}\right]$$

$$\sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & -2 & 5 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 3 & 0 & 9 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & -15 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

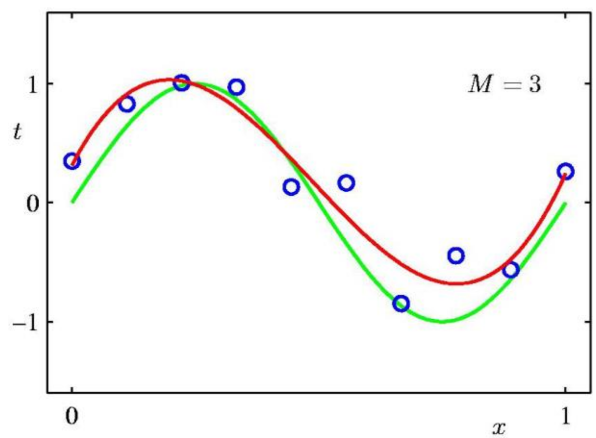
Choosing the order of M

- Model Comparison or Model Selection
- Red lines are best fits with
  - M = 0,1,3,9 and N=10

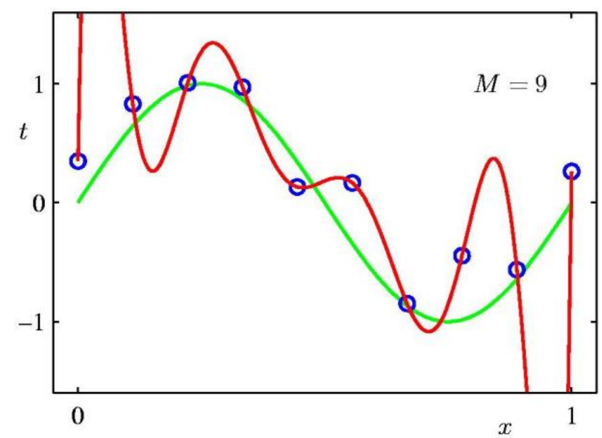




← Poor representations of sin(2πx)



← Best Fit to sin(2πx)



Over Fit
Poor representation of sin(2πx)

Generalization Performance

- Consider separate *test* set of 100 points
- For each value of M evaluate

$$E(\mathrm{w}^*) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathrm{w}^*) - t_n\}^2$$

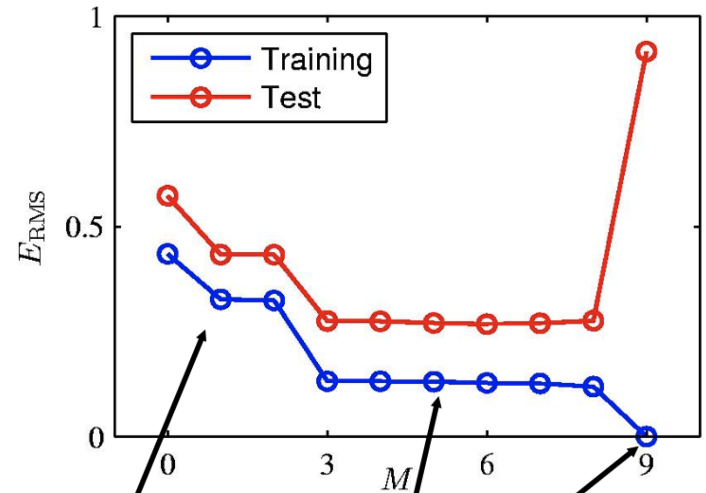
$$\boxed{y(x,\mathrm{w}^*) = \sum_{j=0}^{M} w_j^* x^j}$$

for training data and test data

- Use RMS error

$$E_{RMS} = \sqrt{2E(\mathrm{w}^*)\,/\,N}$$

  – Division by N allows different sizes of N to be compared on equal footing
  – Square root ensures E_{RMS} is measured in same units as t



Poor due to Inflexible polynomials

Small Error

M=9 means ten degrees of freedom. Tuned exactly to 10 training points (wild oscillations in polynomial)

Values of Coefficients *w** for different polynomials of order *M*

	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

As *M* increases magnitude of coefficients increases
At $M=9$ finely tuned to random noise in target values

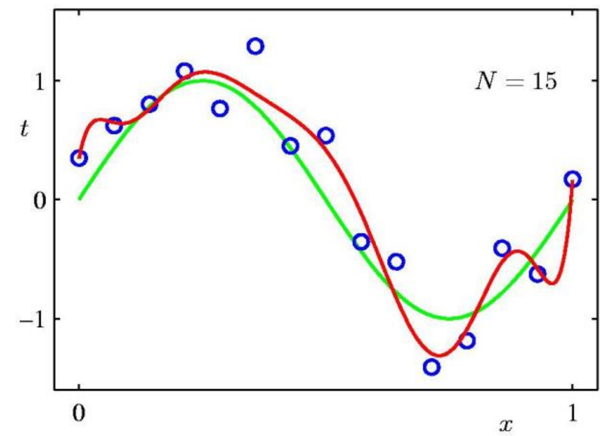
Increasing Size of Data Set

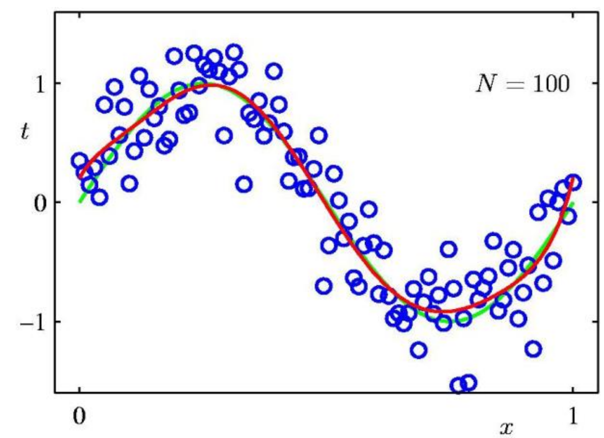
N=15, 100

For a given model complexity overfitting problem is less severe as size of data set increases

Larger the data set, the more complex we can afford to fit the data

Data should be no less than 5 to 10 times adaptive parameters in model





Regularization of Least Squares

- Using relatively complex models with data sets of limited size
- Add a penalty term to error function to discourage coefficients from reaching large values

$$\tilde{E}(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathrm{w}) - t_n\}^2 + \frac{\lambda}{2} || \mathrm{w} ||^2$$

where

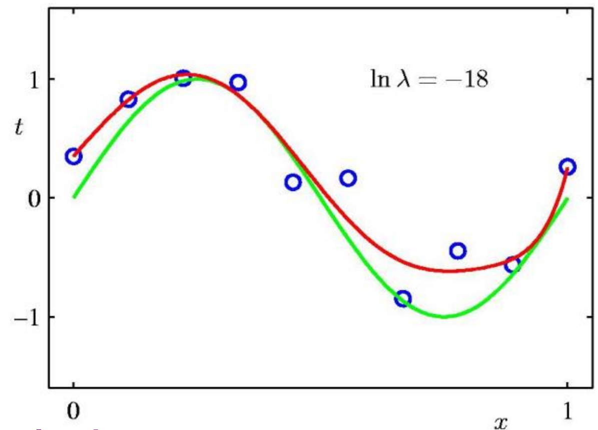
$$|| \mathrm{w} ||^2 \equiv w^T w = w_0^2 + w_1^2 + .. + w_M^2$$

- λ determines relative importance of regularization term to error term
- Can be minimized exactly in closed form
- Known as *ridge regression*
  Weight decay in neural networks

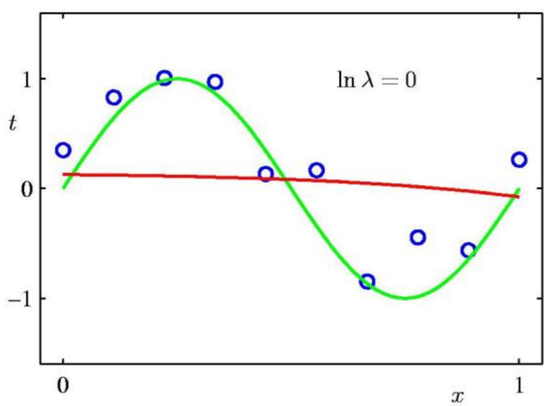
Effect of Regularizer

M=9 polynomials using regularized error function

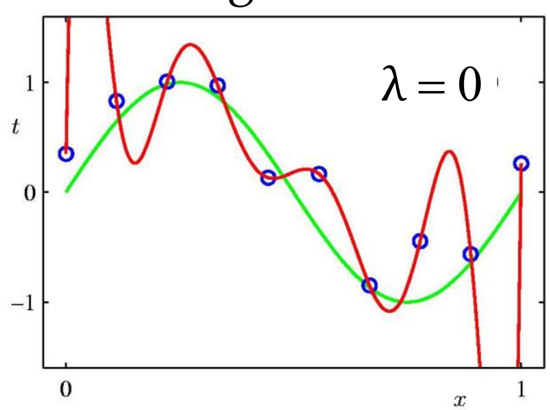
Optimal



Large Regularizer



No Regularizer



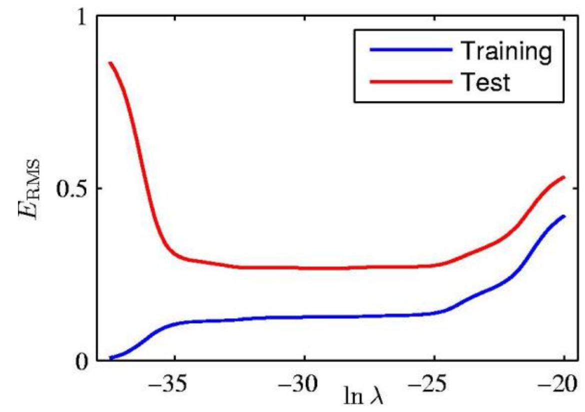
No Regularizer $\lambda = 0$

Large Regularizer $\lambda = 1$

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Impact of Regularization on Error

- λ controls the complexity of the model and hence degree of over-fitting
  - Analogous to choice of M
- Suggested Approach:
- Training set
  - to determine coefficients **w**
  - For different values of (M or λ)
- Validation set (holdout)
  - to optimize model complexity (M or λ)



$M=9$ polynomial

Linear Regression

The (general) regression task

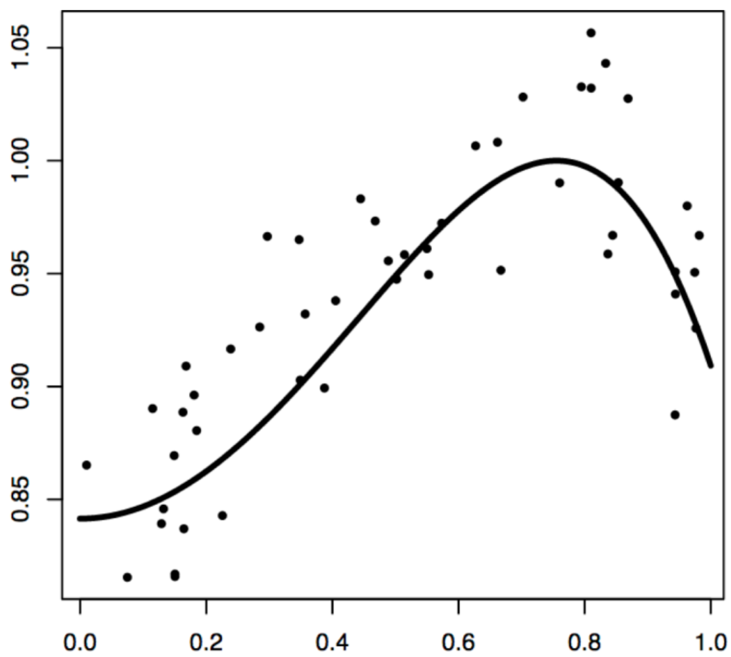
- It is a supervised learning task
- Goal of regression:
  - predict value of one or more <u>target</u> variables t
  - given <u>d-dimensional</u> vector $\mathbf{x}$ of input variables
  - With dataset of known inputs and outputs
    - $(\mathbf{x}_1, t_1), ..(\mathbf{x}_N, t_N)$
    - Where x_i is an input (possibly a vector)
    - t_i is the target output (or response) for case i which is real-valued
  - Goal is to predict t from $\mathbf{x}$ for some future test case

Regression v.s. Classification

- Regression
  - Predict a numerical value t given some input
    - Learning algorithm has to output function $f: \mathrm{R}^n \rightarrow \mathrm{R}$
      - where n = no of input variables
- Classification
  - If t value is a label (categories): $f: \mathrm{R}^n \rightarrow \{1,..,k\}$

An example problem

- Fifty points generated (one-dimensional problem)
  - With x uniform from $(0,1)$
  - y generated from formula $y = \sin(1+x^2) + \text{noise}$
    - Where noise has $N(0, 0.03^2)$ distribution
    - Noise-free true function and data points are as shown



Polynomial Curve Fitting with a Scalar

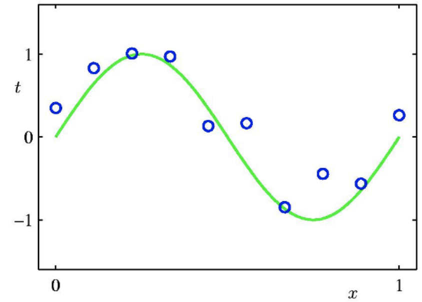
– With a single input variable x

– $y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$

M is the order of the polynomial,

x^j denotes x raised to the power j,

Coefficients $w_0,\ldots,w_M$ are collectively denoted by vector $\mathbf{w}$



Training data set
N=10, Input x, target t

– Task: Learn $\mathbf{w}$ from training data $D = \{(x_i, t_i)\},\ i = 1,..,N$

•Can be done by minimizing an error function that minimizes the misfit between $y(x,\mathbf{w})$ for any given $\mathbf{w}$ and training data

•One simple choice of error function is sum of squares of error between predictions $y(x_n,\mathrm{w})$ for each data point x_n and corresponding target values t_n so that we minimize

$$E(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{y(x_n,\mathrm{w}) - t_n\right\}^2$$

•It is zero when function $y(x,\mathrm{w})$ passes exactly through each training data point

Regression with multiple inputs

- Generalization
  - Predict value of continuous target variable t given value of D input variables $\mathrm{x}=[x_1,..x_D]$
  - t can also be a set of variables (multiple regression)
  - Linear functions of adjustable parameters
- Polynomial curve fitting is good only for:
  - Single input variable scalar x
  - It cannot be easily generalized to several variables

Linear Model with D inputs

- Regression with D input variables

$$y(\mathrm{x},\mathrm{w}) = w_0 + w_1 x_1 + .. + w_d x_D = \mathrm{w}^T\mathrm{x}$$

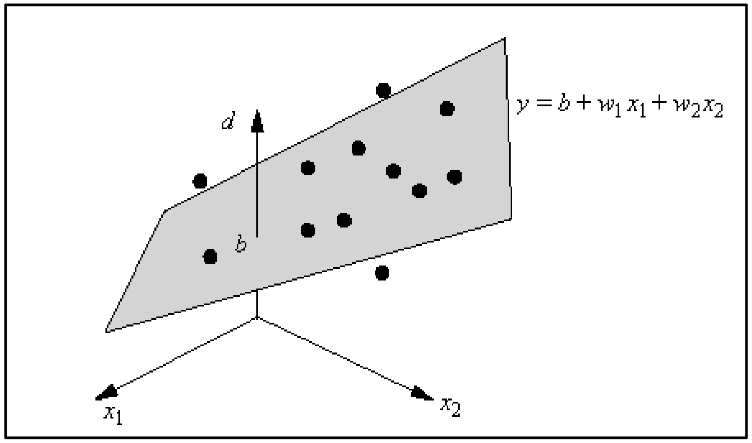
This differs from
Linear Regression with <u>one</u> variable
and Polynomial Reg with <u>one</u> variable

where $\mathrm{x}=(x_1,..,x_D)^{\mathrm{T}}$ are the input variables

- Called Linear Regression since it is a linear function of
  - parameters $w_0,..,w_D$
  - input variables $x_1,..,x_D$
- In the one-dimensional case this amounts a straight-line fit (degree-one polynomial)
  - $y(x,\mathrm{w}) = w_0 + w_1 x$

Fitting a Regression Plane

- Assume t is a function of inputs $x_1, x_2, \ldots x_D$
  Goal: find best linear regressor of t on all inputs
  - Fitting a hyperplane through N input samples
  - For $D = 2$:



x_1	x_2	t
1	2	2
2	5	1
2	3	2
2	2	2
3	4	1
3	5	3
4	6	2
5	5	3
5	6	4
5	7	3
6	8	4
7	6	2
8	4	4
8	9	3
9	8	4

- Being a linear function of input variables imposes limitations on the model
  - Can extend class of models by considering fixed nonlinear functions of input variables

Basis Functions

- In many applications, we apply some form of fixed-preprocessing, or feature extraction, to the original data variables
- If the original variables comprise the vector $\mathbf{x}$, then the features can be expressed in terms of basis functions $\{\varphi_j(\mathbf{x})\}$
  - By using nonlinear basis functions we allow the function $y(\mathbf{x},\mathbf{w})$ to be a nonlinear function of the input vector $\mathbf{x}$
    - They are linear functions of parameters (gives them simple analytical properties), yet are nonlinear wrt input variables

Linear Regression with M Basis Functions

- Extended by considering nonlinear functions of input variables

$$y(\mathbf{x},\mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

  - where $\varphi_j(\mathrm{x})$ are called Basis functions
  - We now need M weights for basis functions instead of D weights for features
  - With a dummy basis function $\varphi_0(\mathbf{x})=1$ corresponding to the bias parameter w_0, we can write

$$y(\mathbf{x},\mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

  - where $\mathbf{w}=(w_0,w_1,..,w_{M-1})$ and $\Phi=(\varphi_0,\varphi_1,..,\varphi_{M\text{-}1})^{\mathrm{T}}$

- Basis functions allow non-linearity with D input variables

Learning

A regression model is a linear one when the model comprises a linear combination of the parameters, i.e.,

$$f(x, \boldsymbol{\beta}) = \sum_{j=1}^{m} \beta_j \phi_j(x),$$

Letting $X_{ij} = \phi_j(x_i)$ Y is the vector of output values D = (X, Y) is the set of all data

$$L(D, \boldsymbol{\beta}) = \|X\boldsymbol{\beta} - Y\|^2 = (X\boldsymbol{\beta} - Y)^\mathsf{T}(X\boldsymbol{\beta} - Y) = Y^\mathsf{T}Y - Y^\mathsf{T}X\boldsymbol{\beta} - \boldsymbol{\beta}^\mathsf{T}X^\mathsf{T}Y + \boldsymbol{\beta}^\mathsf{T}X^\mathsf{T}X\boldsymbol{\beta}$$

Finding the minimum can be achieved through setting the gradient of the loss to zero and solving for $\vec{\beta}$

$$\frac{\partial L(D, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{\partial \left(Y^\mathsf{T}Y - Y^\mathsf{T}X\boldsymbol{\beta} - \boldsymbol{\beta}^\mathsf{T}X^\mathsf{T}Y + \boldsymbol{\beta}^\mathsf{T}X^\mathsf{T}X\boldsymbol{\beta}\right)}{\partial \boldsymbol{\beta}} = -2X^\mathsf{T}Y + 2X^\mathsf{T}X\boldsymbol{\beta}$$

Finally setting the gradient of the loss to zero and solving for $\boldsymbol{\beta}$ we get:

$$-2X^\mathsf{T}Y + 2X^\mathsf{T}X\boldsymbol{\beta} = 0 \Rightarrow X^\mathsf{T}Y = X^\mathsf{T}X\boldsymbol{\beta}$$

$$\hat{\boldsymbol{\beta}} = \left(X^\mathsf{T}X\right)^{-1}X^\mathsf{T}Y$$

Vector derivatives

Scalar derivative			Vector derivative		
$f(x)$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}x}$	$f(\mathbf{x})$	$\rightarrow$	$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{B}$	$\rightarrow$	$\mathbf{B}$
bx	$\rightarrow$	b	$\mathbf{x}^T\mathbf{b}$	$\rightarrow$	$\mathbf{b}$
x^2	$\rightarrow$	$2x$	$\mathbf{x}^T\mathbf{x}$	$\rightarrow$	$2\mathbf{x}$
bx^2	$\rightarrow$	$2bx$	$\mathbf{x}^T\mathbf{B}\mathbf{x}$	$\rightarrow$	$2\mathbf{B}\mathbf{x}$

http://www.gatsby.ucl.ac.uk/teaching/courses/sntn/sntn-2017/resources/Matrix_derivatives_cribsheet.pdf

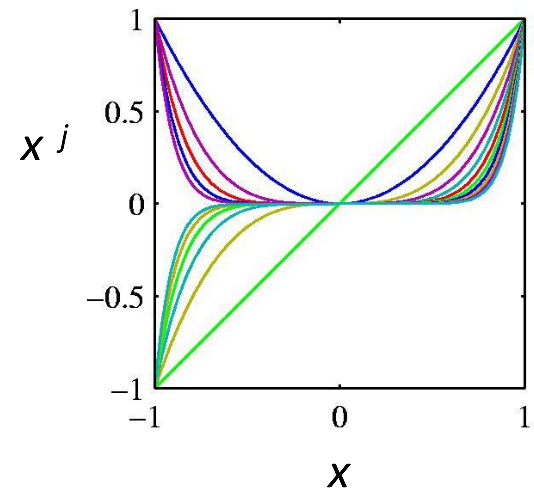
Choice of Basis Functions

- Many possible choices for basis function:
  1. Polynomial regression
     - Good only if there is only one input variable
  2. Gaussian basis functions
  3. Sigmoidal basis functions
  4. Fourier basis functions
  5. Wavelets

1. Polynomial Basis for one variable

- Linear Basis Function Model

$$y(x,\mathbf{w}) = \sum_{j=0}^{M-1} w_j \varphi_j(x) = \mathbf{w}^T \boldsymbol{\varphi}(x)$$



- Polynomial Basis (for single variable x)

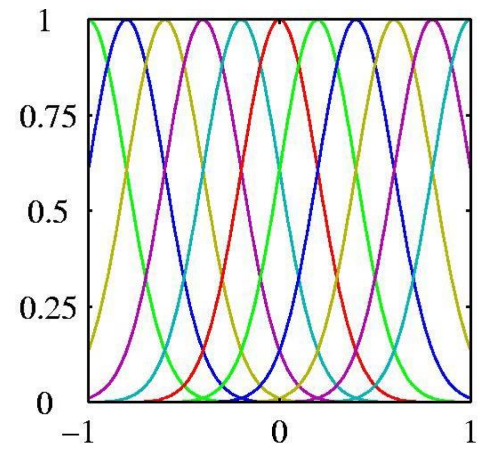
  $\varphi_j(x)=x^j$ with degree M-1 polynomial

- Disadvantage
  - Good only if there is only one input variable

2. Gaussian Radial Basis Functions

- Gaussian

$$\phi_j(x) = \exp\left(\frac{(x-\mu_j)^2}{2\sigma^2}\right)$$



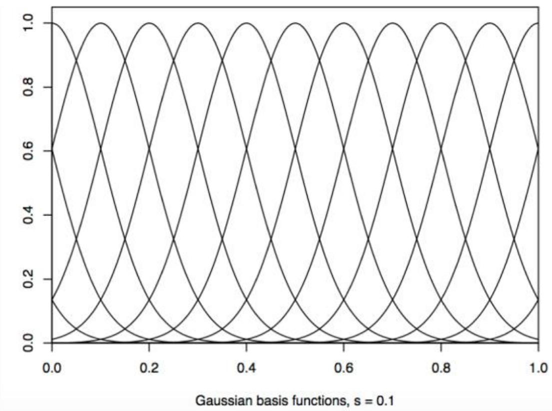
- Choice of parameters
  - μ_j govern the locations of the basis functions
    - Can be an arbitrary set of points within the range of the data
      - Can choose some representative data points
  - σ governs the spatial scale
    - Could be chosen from the data set e.g., average variance
- Several variables
  - A Gaussian kernel would be chosen for each dimension
  - For each j a different set of means would be needed– perhaps chosen from the data

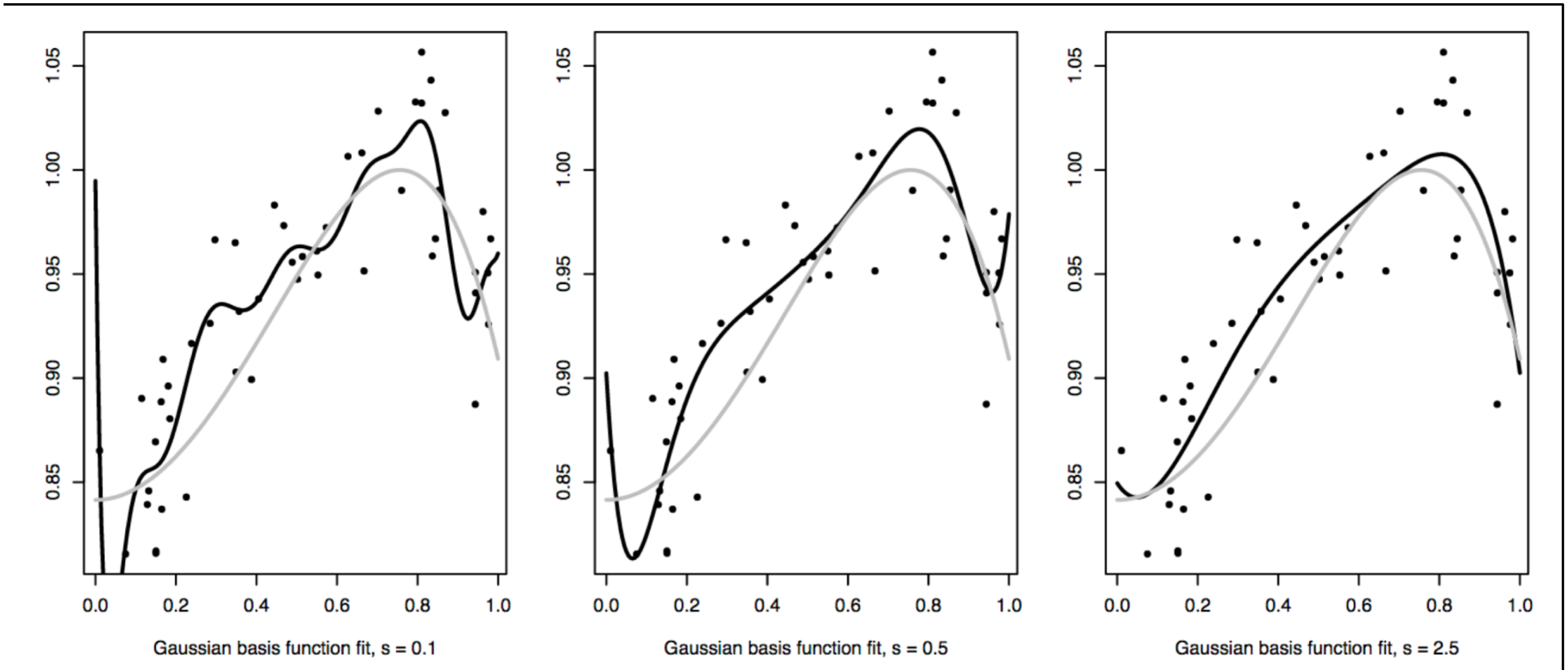
$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right)$$

Result with Gaussian Basis Functions

$$\phi_j(x) \quad = \quad \exp(-(x-\mu_j)^2/2s^2)$$

Basis functions for $s=0.1$, with the μ_j on a grid with spacing s





w_j s for middle model:

6856.5
-3544.1
-2473.7
-2859.8
-2637.7
-2861.5
-2468.0
-3558.4

3. Other Basis Functions

- Fourier
  - Expansion in sinusoidal functions
  - Infinite spatial extent
- Signal Processing
  - Functions localized in time and frequency
  - Called *wavelets*
    - Useful for lattices such as images and time series