Linear and Non-Linear Regression

Dr. Jianlin Cheng

Department of Electrical Engineering and Computer Science
University of Missouri, Columbia

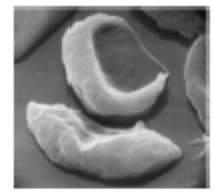
Fall, 2019

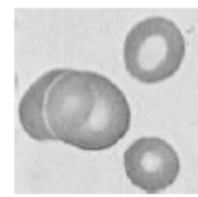
Slides Adapted from Book, CMU, Stanford Machine Learning Courses, and my presentations

Discrete to Continuous Labels

Classification

Sports
Science
News



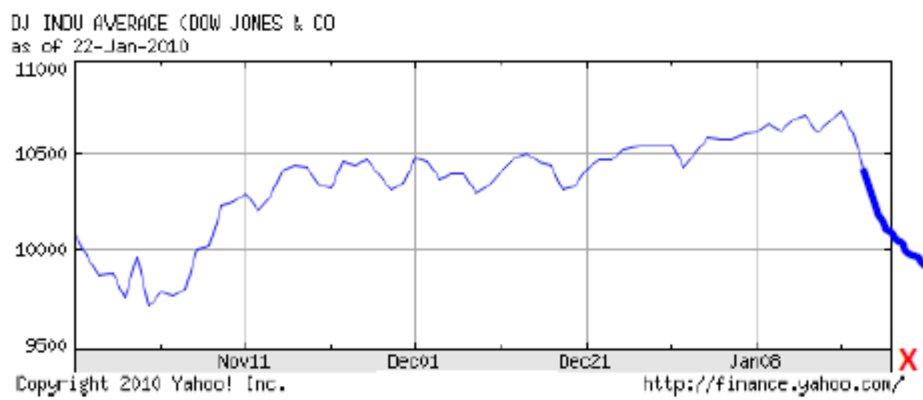


Anemic cell
Healthy cell

X = Document   Y = Topic   X = Cell Image   Y = Diagnosis

Regression

Stock Market Prediction

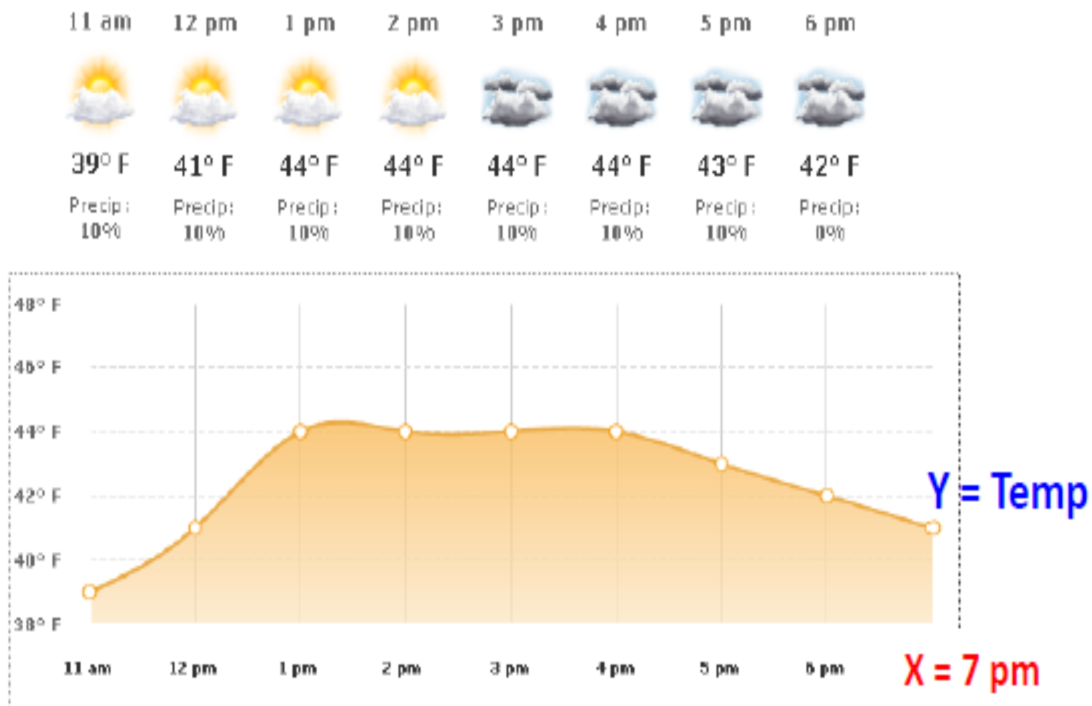


Y = ?

X = Feb01

Regression Tasks

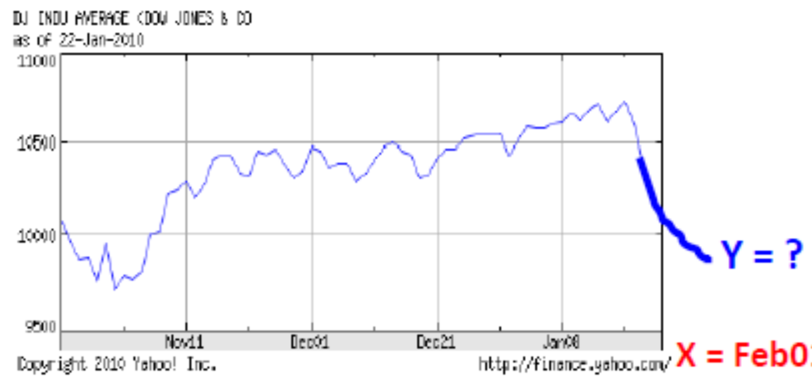
Weather Prediction



Supervised Learning

Goal: Construct a **predictor** $f : X \to Y$ to minimize a risk (performance measure) $R(f)$

Sports
Science
News



Classification:

$$R(f) = P(f(X) \neq Y)$$

Probability of Error

Regression:

$$R(f) = \mathbb{E}[(f(X) - Y)^2]$$

Mean Squared Error

Regression

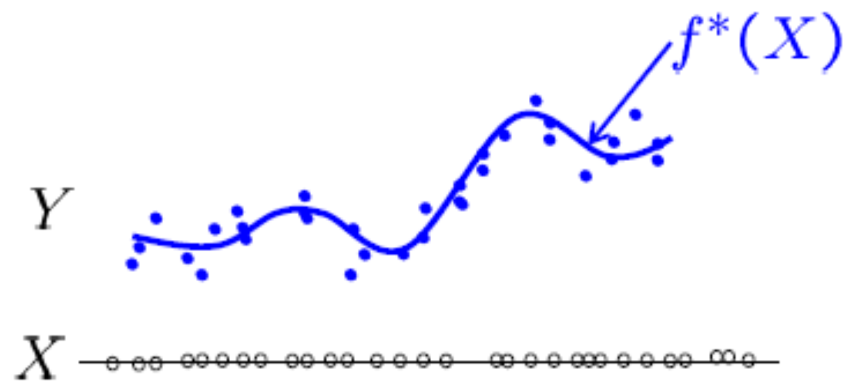
Optimal predictor:

$$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

$$= \mathbb{E}[Y|X] \qquad \text{(Conditional Mean)}$$

Intuition: Signal plus (zero-mean) Noise model

$$Y = f^*(X) + \epsilon$$



Regression

Optimal predictor: $f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2] = \mathbb{E}[Y|X]$

Proof Strategy: $R(f) \geq R(f^*)$ for any prediction rule f

$$R(f) = \mathbb{E}_{XY}[(f(X) - Y)^2] = \mathbb{E}_X[\mathbb{E}_{Y|X}[(f(X) - Y)^2|X]]$$

Dropping subscripts for notational convenience

$$= E\left[E\left[(\underbrace{f(X) - E[Y|X]} + \underbrace{E[Y|X] - Y})^2|X\right]\right]$$

$$= \begin{aligned} E[\quad & E[(f(X) - E[Y|X])^2|X] \\ & +2E\left[(f(X) - E[Y|X])(E[Y|X] - Y)|X\right] \\ & +E[(E[Y|X] - Y)^2|X]] \end{aligned}$$

$$= \begin{aligned} E[\quad & E[(f(X) - E[Y|X])^2|X] \\ & +2(f(X) - E[Y|X]) \times 0 \\ & +E[(E[Y|X] - Y)^2|X]] \end{aligned}$$

$$= \underbrace{E\left[(f(X) - E[Y|X])^2\right]}_{\geq 0} + R(f^*).$$

Regression

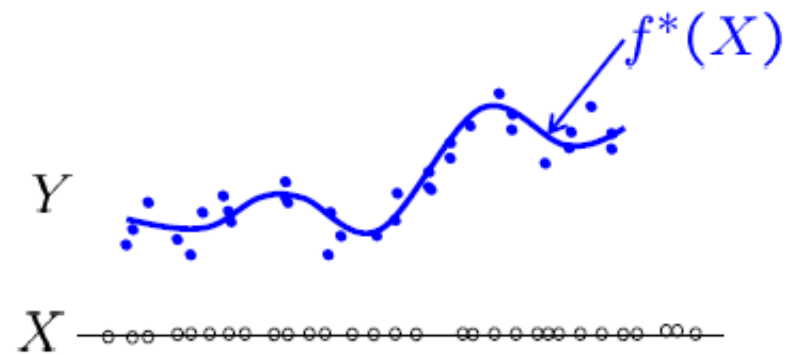
Optimal predictor:

$$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

$$= \mathbb{E}[Y|X] \qquad \text{(Conditional Mean)}$$

Intuition: Signal plus (zero-mean) Noise model

$$Y = f^*(X) + \epsilon$$



Depends on **unknown** distribution P_{XY}

Regression algorithms

Training data $\{(X_i, Y_i)\}_{i=1}^n$ → **Learning algorithm** → Prediction rule $\widehat{f}_n$

Linear Regression

Lasso, Ridge regression (Regularized Linear Regression)

Nonlinear Regression

Kernel Regression

Regression Trees, Splines, Wavelet estimators, …

Empirical Risk Minimization (ERM)

Optimal predictor: $$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

Empirical Risk Minimizer: $$\widehat{f}_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}(f(X_i) - Y_i)^2$$

Class of predictors **Empirical mean**

$$\boxed{\frac{1}{n}\sum_{i=1}^{n}[\text{loss}(Y_i, f(X_i))] \xrightarrow[\text{Numbers}]{\text{Law of Large}} \mathbb{E}_{XY}[\text{loss}(Y, f(X))]}$$

ERM – you saw it before!

- Learning Distributions

  Max likelihood = Min -ve log likelihood empirical risk

$$\max_\theta P(D|\theta) = \min_\theta \frac{1}{n}\sum_{i=1}^{n} \underbrace{-\log P(X_i|\theta)}_{\text{loss}(X_i,\theta)}$$

Negative log Likelihood loss

What is the class $\mathcal{F}$?

Class of parametric distributions

Bernoulli (θ)

Gaussian (μ, σ^2)

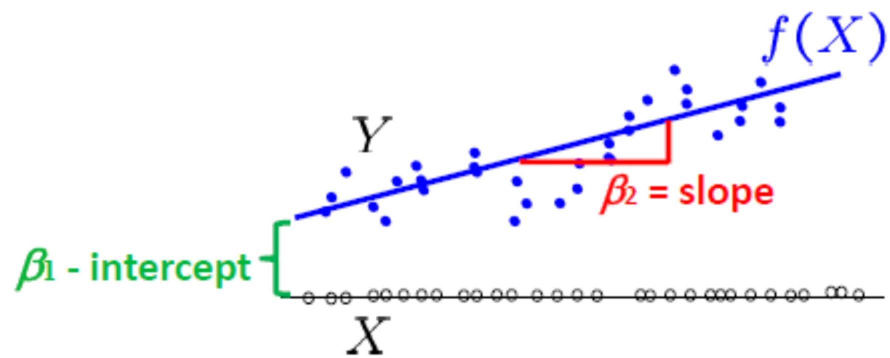
Linear Regression

$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$ Least Squares Estimator

$\mathcal{F}_L$ - Class of Linear functions

Uni-variate case:

$$f(X) = \beta_1 + \beta_2 X$$



Multi-variate case:

$$f(X) = f(X^{(1)}, \ldots, X^{(p)}) = \beta_1 \cancelto{1}{X^{(1)}} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)}$$

$$= X\beta$$ where $X = [X^{(1)} \ldots X^{(p)}]$, $\beta = [\beta_1 \ldots \beta_p]^T$

Least Squares Estimator

$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

$$\Downarrow$$

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (X_i \beta - Y_i)^2 \qquad \widehat{f}_n^L(X) = X\widehat{\beta}$$

$$= \arg\min_{\beta} \frac{1}{n} (\mathbf{A}\beta - \mathbf{Y})^T (\mathbf{A}\beta - \mathbf{Y})$$

$$\mathbf{A} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} X_1^{(1)} & \dots & X_1^{(p)} \\ \vdots & \ddots & \vdots \\ X_n^{(1)} & \dots & X_n^{(p)} \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix}$$

Least Squares Estimator

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

$$J(\beta) = (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y})$$

$$= \boldsymbol{A^T A \beta \beta^T - 2\beta^T A^T Y + Y^T Y}$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0 \quad \boldsymbol{= 2A^T A\beta - 2A^T Y = 0}$$

Normal Equations

$$(\mathbf{A}^T\mathbf{A})\widehat{\beta} = \mathbf{A}^T\mathbf{Y}$$

p xp p x1 p x1

If $(\mathbf{A}^T\mathbf{A})$ is invertible,

$$\boxed{\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \widehat{f}_n^L(X) = X\widehat{\beta}}$$

When is $(\mathbf{A}^T\mathbf{A})$ invertible ?
Recall: Full rank matrices are invertible. What is rank of $(\mathbf{A}^T\mathbf{A})$?

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?
Regularization (later)

Revisiting Gradient Descent

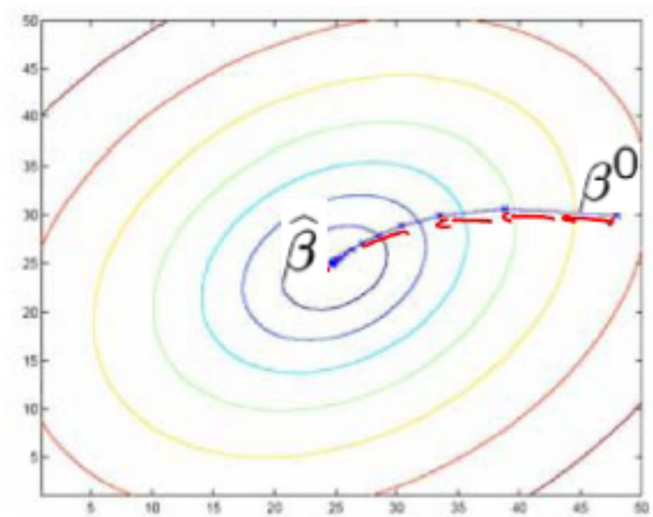
Even when $(\mathbf{A}^T\mathbf{A})$ is invertible, might be computationally expensive if **A** is huge.

$$\widehat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

Gradient Descent since J(β) is convex

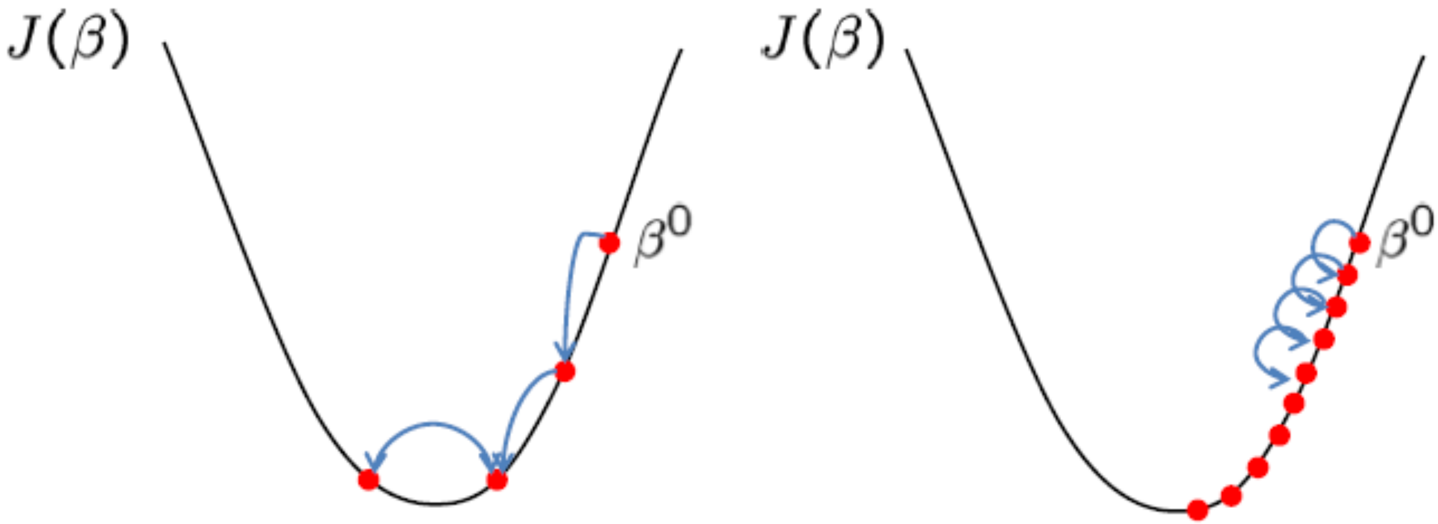
Initialize: β^0

Update:
$$\begin{aligned}\beta^{t+1} &= \beta^t - \frac{\alpha}{2}\frac{\partial J(\beta)}{\partial \beta}\bigg|_t \\ &= \beta^t - \alpha\ \mathbf{A}^T\underbrace{(\mathbf{A}\beta^t - Y)}_{0 \text{ if } \beta^t = \widehat{\beta}}\end{aligned}$$



Stop: when some criterion met e.g. fixed # iterations, or $\frac{\partial J(\beta)}{\partial \beta}\Big|_{\beta^t} < \varepsilon$.

Effect of step-size α



Large α => Fast convergence but larger residual error
Also possible oscillations

Small α => Slow convergence but small residual error

Least Squares and MLE

Intuition: Signal plus (zero-mean) Noise model

$$Y = f^*(X) + \epsilon = X\beta^* + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$$

$$Y \sim \mathcal{N}(X\beta^*, \sigma^2\mathbf{I})$$

$$P(Y_i|X_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(Y_i - X_i\beta)^2}{-2\sigma^2}}$$

$$\widehat{\beta}_{\mathrm{MLE}} = \arg\max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^n | \beta, \sigma^2)}_{\text{log likelihood}}$$

$$= \arg\min_{\beta} \sum_{i=1}^{n} (X_i\beta - Y_i)^2 = \widehat{\beta}$$

Least Square Estimate is same as Maximum Likelihood Estimate under a Gaussian model !

An early demonstration of the strength of **Gauss**'s method came when it was used to predict the future location of the newly discovered **asteroid Ceres**. On January 1, 1801, the Italian astronomer **Giuseppe Piazzi** discovered Ceres and was able to track its path for 40 days before it was lost in the glare of the sun. Based on this data, astronomers desired to **determine the location of Ceres** after it emerged from behind the sun without solving the complicated Kepler's nonlinear equations of planetary motion. The only predictions that successfully allowed Hungarian astronomer **Franz Xaver von Zach** to relocate Ceres were those performed by the **24-year-old Gauss** using **least-squares analysis**.

Source: Wikipedia

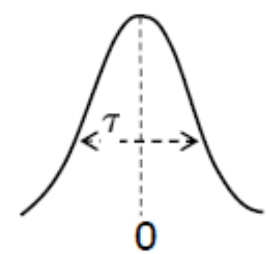
Regularized Least Squares and MAP

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?

$$\widehat{\beta}_{\text{MAP}} = \arg\max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^n | \beta, \sigma^2)}_{\text{log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

I) Gaussian Prior

$$\beta \sim \mathcal{N}(0, \tau^2\mathbf{I}) \qquad p(\beta) \propto e^{-\beta^T\beta/2\tau^2}$$



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^n (Y_i - X_i\beta)^2 + \underset{\substack{\downarrow \\ \text{constant}(\sigma^2, \tau^2)}}{\lambda} \|\beta\|_2^2$$

Ridge Regression

Prior belief that β is Gaussian with zero-mean biases solution to “small” β

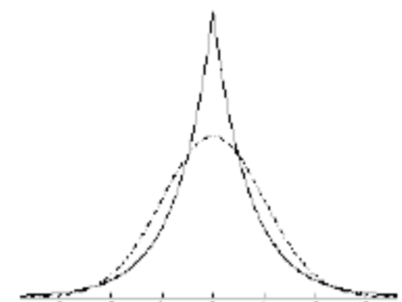
Regularized Least Squares and MAP

What if $(\mathbf{A}^T\mathbf{A})$ is not invertible ?

$$\widehat{\beta}_{\mathsf{MAP}} = \arg\max_{\beta} \underbrace{\log p(\{(X_i, Y_i)\}_{i=1}^n | \beta, \sigma^2)}_{\text{log likelihood}} + \underbrace{\log p(\beta)}_{\text{log prior}}$$

II) Laplace Prior

$$\beta_i \overset{iid}{\sim} \text{Laplace}(0, t) \qquad\qquad p(\beta_i) \propto e^{-|\beta_i|/t}$$



$$\widehat{\beta}_{\mathsf{MAP}} = \arg\min_{\beta} \sum_{i=1}^n (Y_i - X_i\beta)^2 + \underset{\substack{\downarrow \\ \text{constant}(\sigma^2, t)}}{\lambda} \|\beta\|_1$$

Lasso

Prior belief that β is Laplace with zero-mean biases solution to “small” β

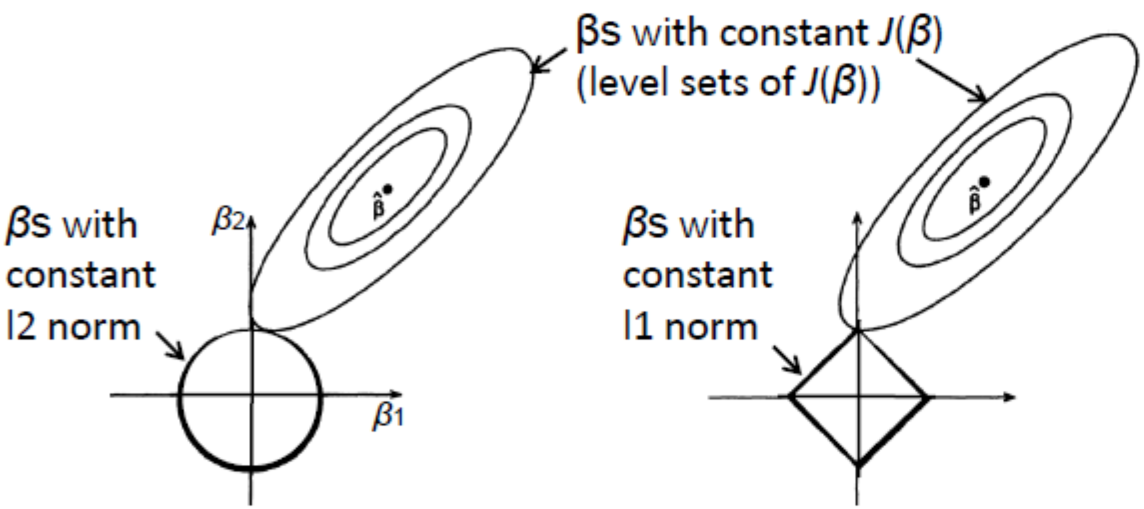
Ridge Regression vs Lasso

$$\min_{\beta}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) + \lambda \text{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \text{pen}(\beta)$$

Ridge Regression:
$\text{pen}(\beta) = \|\beta\|_2^2$

Lasso:
$\text{pen}(\beta) = \|\beta\|_1$

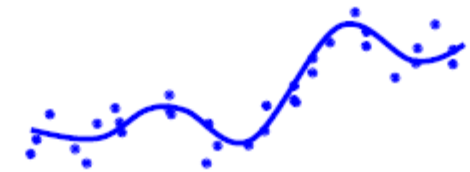
HOT!



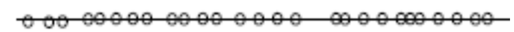
Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates!

Beyond Linear Regression

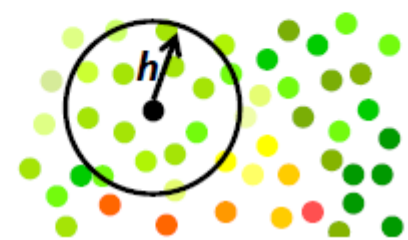
Polynomial regression



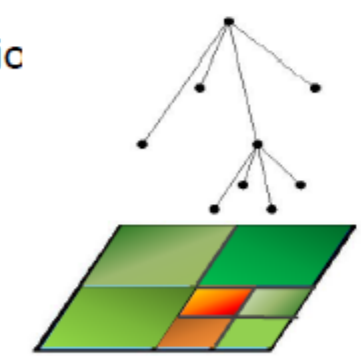
Regression with nonlinear features/basis functions



Kernel regression - Local/Weighted regression



Regression trees – Spatially adaptive regressic



Polynomial Regression

Univariate (1-d) case: $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_m X^m = \mathbf{X}\beta$

where $\mathbf{X} = [1 \;\; X \;\; X^2 \ldots X^m],\, \beta = [\beta_1 \ldots \beta_m]^T$

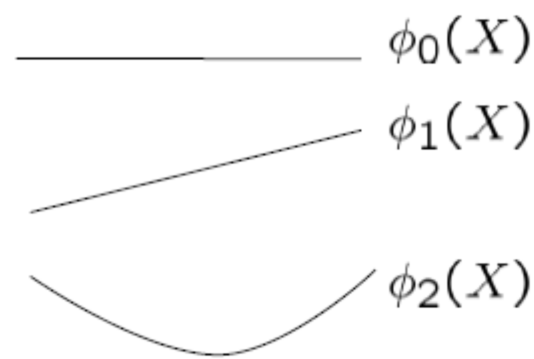
$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}$$

$$\widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$$

$$\mathbf{A} = \begin{bmatrix} 1 & X_1 & X_1^2 & \ldots & X_1^m \\ \vdots & & & \ddots & \vdots \\ 1 & X_n & X_n^2 & \ldots & X_n^m \end{bmatrix}$$

$$f(X) = \sum_{j=0}^{m} \beta_j X^j = \sum_{j=0}^{m} \beta_j \underbrace{\phi_j(X)}_{\text{Nonlinear features}}$$

Weight of each feature



A Regression Example

Average height and weight of American women aged 30 - 39

Height/ m	1.47	1.5	1.52	1.55	1.57	1.60	1.63	1.65	1.68	1.7	1.73	1.75	1.78	1.8	1.83
Weight/kg	52.21	53.12	54.48	55.84	57.2	58.57	59.93	61.29	63.11	64.47	66.28	68.1	69.92	72.19	74.46

Weight is not linear with height, so add a quadratic term into regression

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$$

$$\hat{f}(X) = X\hat{\beta}$$

$$A = \begin{bmatrix} 1 & 1.47 & 2.16 \\ 1 & 1.50 & 2.25 \\ 1 & 1.52 & 2.31 \\ 1 & 1.55 & 2.40 \\ 1 & 1.57 & 2.46 \\ 1 & 1.60 & 2.56 \\ 1 & 1.63 & 2.66 \\ 1 & 1.65 & 2.72 \\ 1 & 1.68 & 2.82 \\ 1 & 1.70 & 2.89 \\ 1 & 1.73 & 2.99 \\ 1 & 1.75 & 3.06 \\ 1 & 1.78 & 3.17 \\ 1 & 1.81 & 3.24 \\ 1 & 1.83 & 3.35 \end{bmatrix} \qquad Y = \begin{matrix} 52.21 \\ 53.12 \\ 54.48 \\ 55.84 \\ 57.2 \\ 58.57 \\ 59.93 \\ 61.29 \\ 63.11 \\ 64.47 \\ 66.28 \\ 68.1 \\ 69.92 \\ 72.19 \\ 74.46 \end{matrix}$$

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}$$

$$\hat{\beta}_0 = ?$$

$$\hat{\beta}_1 = ?$$

$$\hat{\beta}_2 = ?$$

Source: Wikipedia

Assignment 3 – Programming 1

- Write programs in Matlab, R, C/C++, Java, Perl, or Python to implement the analytical (e.g. matrix-based) **or** iterative (e.g. gradient descent) linear regression algorithm and test it on the problem in the previous slide. Don't directly call linear regression functions in any software
- Turn in the programs and execution results

Assignment 3 – Programming 2
Due Sept. 28, 2015

Iris

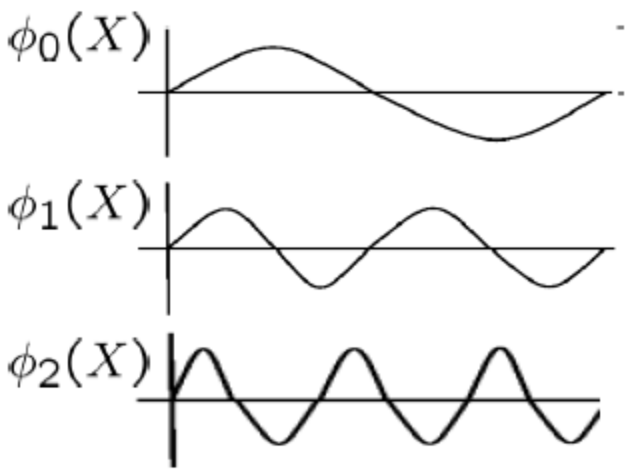
Wikipedia

- Write a program to implement the iterative (e.g. gradient ascent / descent) logistic regression algorithm for binary classification and apply it to the Iris classification data set
- Iris data set: http://archive.ics.uci.edu/ml/datasets/Iris
- Only select data points of two highlighted classes (**Iris Setosa, Iris Versicolour**, Iris Virginica)
- Submit programs and execution results

Nonlinear Regression

$$f(X) = \sum_{j=0}^{m} \underbrace{\beta_j}_{\text{Basis coefficients}} \underbrace{\phi_j(X)}_{\text{Nonlinear features/basis functions}}$$

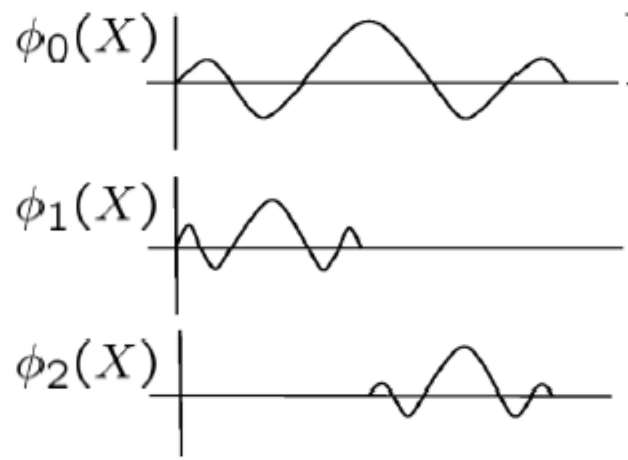
Fourier Basis



Good representation for oscillatory functions

$$\frac{1}{\sqrt{2\pi a^2}} \cdot \operatorname{sinc}\left(\frac{\omega}{2\pi a}\right)$$

Wavelet Basis



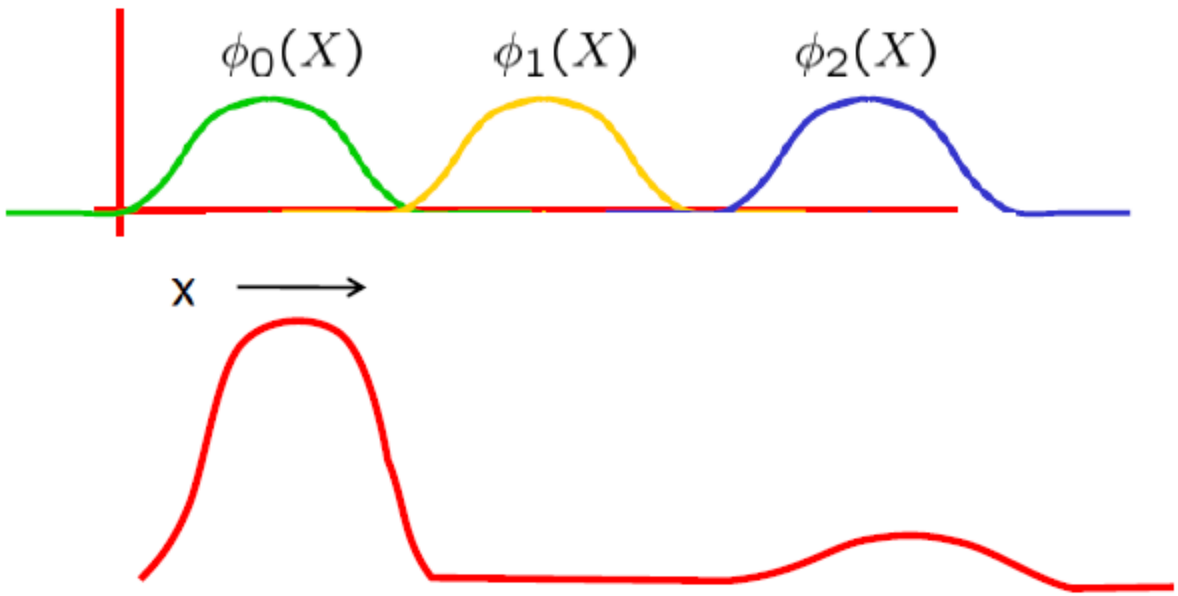
Good representation for functions localized at multiple scales

$$\psi(t) = 2\operatorname{sinc}(2t) - \operatorname{sinc}(t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t}$$

Local Regression

$$f(X) = \sum_{j=0}^{m} \beta_j \underbrace{\phi_j(X)}_{\text{Nonlinear features/basis functions}}$$

Basis coefficients

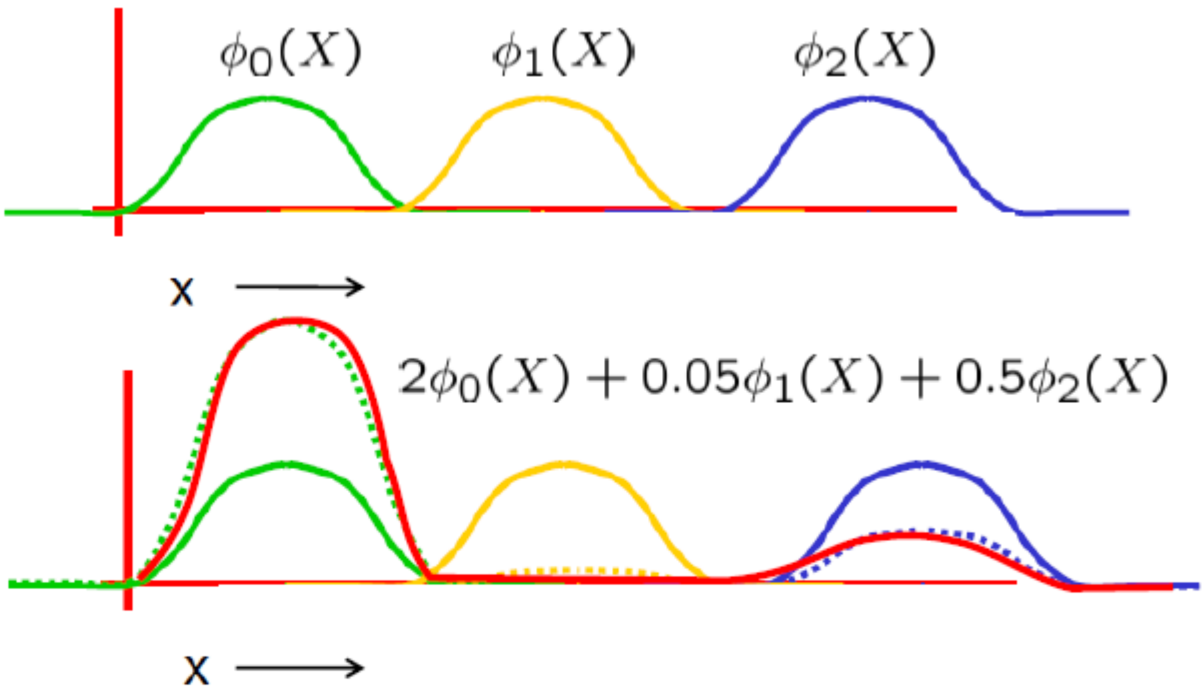


Globally supported basis functions (polynomial, fourier) will not yield a good representation

Local Regression

$$f(X) = \Sigma_{j=0}^{m} \beta_j \underbrace{\phi_j(X)}_{\text{Nonlinear features/basis functions}}$$

Basis coefficients



Globally supported basis functions (polynomial, fourier) will not yield a good representation

What you should know

Linear Regression

- Least Squares Estimator
- Normal Equations
- Gradient Descent

Regularized Linear Regression (connection to MAP)

- Ridge Regression, Lasso

Polynomial Regression, Basis (Fourier, Wavelet) Estimators

Next time

- Kernel Regression (Localized)
- Regression Trees