# Revisiting
# Logistic Regression & Naïve Bayes

Aarti Singh

Machine Learning 10-701/15-781
Jan 27, 2010

MACHINE LEARNING DEPARTMENT
Carnegie Mellon
School of Computer Science

# Generative and Discriminative Classifiers

Training classifiers involves learning a mapping f: X -> Y, or P(Y|X)

Generative classifiers (e.g. Naïve Bayes)

- Assume some functional form for P(X,Y) (or P(X|Y) and P(Y))
- Estimate parameters of P(X|Y), P(Y) directly from training data
- Use Bayes rule to calculate P(Y|X)

Discriminative classifiers (e.g. Logistic Regression)

- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

# Logistic Regression

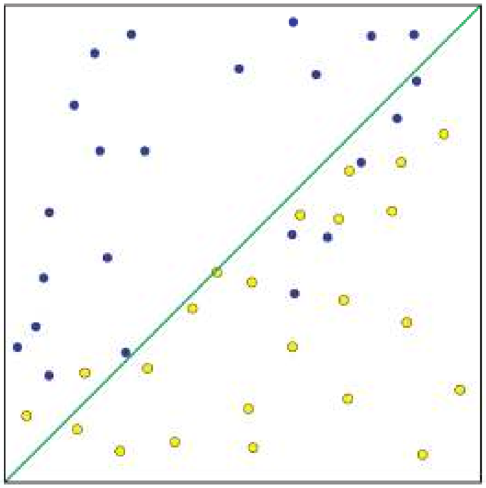
Assumes the following functional form for P(Y|X):

$$P(Y=1|X) = \frac{1}{1+\exp(w_0 + \sum_i w_i X_i)}$$

Alternatively,

$$\log \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

(Linear Decision Boundary)



**DOES NOT require any conditional independence assumptions**

# Connection to Gaussian Naïve Bayes

There are several distributions that can lead to a linear decision boundary.

As another example, consider a generative model:

$$Y \sim \mathrm{Bernoulli}(\pi)$$

$$P(X|Y=y) \propto e^{\phi_y(X)}$$

Exponential family

$$\phi_y(X) = a_y + \sum_i b_{i,y} X_i + \sum_{ij} c_{ij,y} X_i X_j + \ldots$$

Observe that Gaussian is a special case

If coefficients of all non-linear terms are same for $y=0$ and $y=1$, e.g. $c_{ij,0} = c_{ij,1}$, we have a linear decision boundary:

$$\log \frac{P(X|Y=0)}{P(X|Y=1)} = \log P(X|Y=0) - \log P(X|Y=1) = (a_0 - a_1) + \sum_i (b_{i,0} - b_{i,1}) X_i$$

# Connection to Gaussian Naïve Bayes

$$\log \frac{P(Y=0|X)}{P(Y=1|X)} = \log \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)} = \log \frac{1-\pi}{\pi} + \log \frac{P(X|Y=0)}{P(X|Y=1)}$$

$$= \underbrace{\log \frac{1-\pi}{\pi} + (a_0 - a_1)}_{\text{Constant term}} + \underbrace{\sum_i (b_{i,0} - b_{i,1}) X_i}_{\text{First-order term}}$$

$$=: w_0 + \sum_i w_i X_i$$

**Special case:** P(X|Y=y) ~ Gaussian( $\mu_y, \Sigma_y$ ) where $\Sigma_0 = \Sigma_1$ ($c_{ij,0} = c_{ij,1}$)

Conditionally independent $c_{ij,y} = 0$, $i \neq j$

(Gaussian Naïve Bayes)

# Generative vs Discriminative

Given **infinite data** (asymptotically),

If conditional independence assumption holds,
Discriminative and generative perform similar.

$$\epsilon_{\mathrm{Dis},\infty} \sim \epsilon_{\mathrm{Gen},\infty}$$

If conditional independence assumption does NOT holds,
Discriminative outperforms generative.

$$\epsilon_{\mathrm{Dis},\infty} < \epsilon_{\mathrm{Gen},\infty}$$

# Generative vs Discriminative

Given **finite data** (n data points, p features),

$$\epsilon_{\mathrm{Dis},n} \leq \epsilon_{\mathrm{Dis},\infty} + O\left(\sqrt{\frac{p}{n}}\right)$$

$$\epsilon_{\mathrm{Gen},n} \leq \epsilon_{\mathrm{Gen},\infty} + O\left(\sqrt{\frac{\log p}{n}}\right)$$

Ng-Jordan paper

Naïve Bayes (generative) requires n = O(log p) to converge to its asymptotic error, whereas Logistic regression (discriminative) requires n = O(p).

Why? "Independent class conditional densities"

* smaller classes are easier to learn
* parameter estimates not coupled – each parameter is learnt independently, not jointly, from training data.

# Naïve Bayes vs Logistic Regression

## Verdict

Both learn a linear decision boundary.

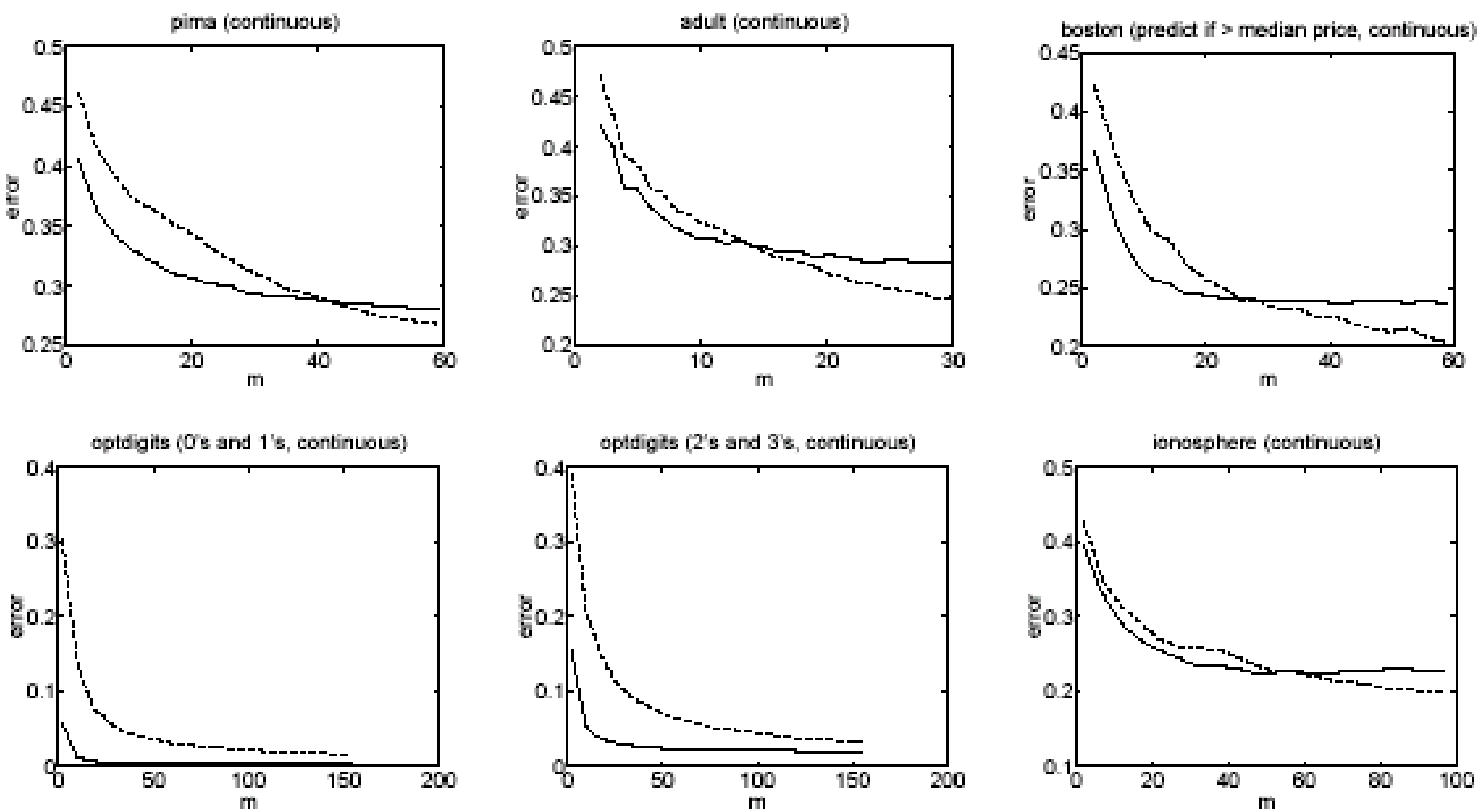
Naïve Bayes makes more restrictive assumptions and has higher asymptotic error,

BUT

converges faster to its less accurate asymptotic error.

# Experimental Comparison (Ng-Jordan'01)

UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features




More in Paper…

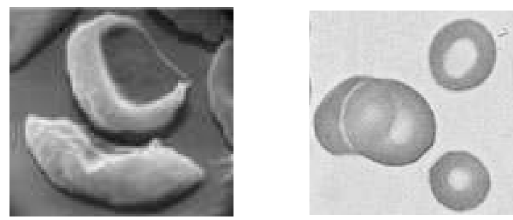
—— Naïve Bayes ----- Logistic Regression

# Classification so far … (Recap)

# Classification Tasks

**Features, X** | **Labels, Y**

Diagnosing sickle cell anemia



⇨ Anemic cell
Healthy cell

Tax Fraud Detection

| Refund | Marital Status | Taxable Income |
| --- | --- | --- |
| No | Married | 80K |

⇨

| Cheat |
| --- |
| ? |

Web Classification


⇨ Sports
Science
News

Predict squirrel hill resident

Drive to CMU, Rachel's fan, Shop at SH Giant Eagle

⇨ Resident
Not resident

# Classification

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

Sports
Science
News

**Features, X**

**Labels, Y**

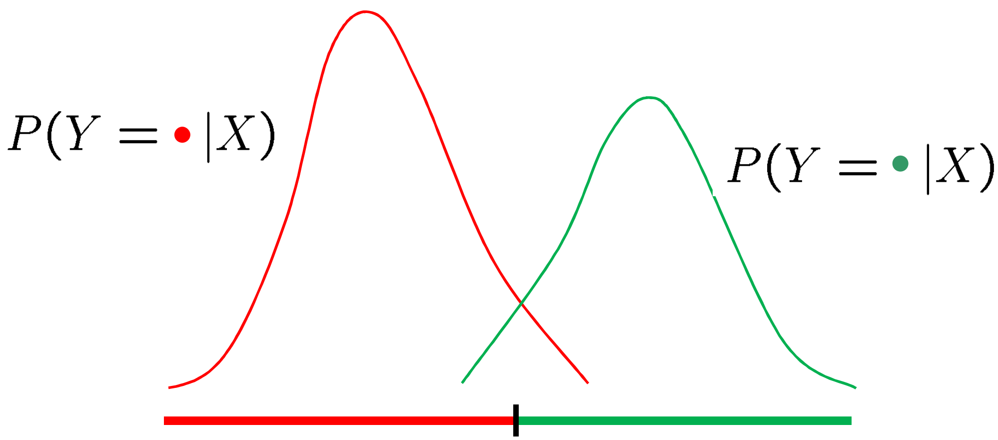
$$R(f) = P(f(X) \neq Y)$$

**Probability of Error**

# Classification

Optimal predictor:
(Bayes classifier)

$$f^* = \arg\min_f P(f(X) \neq Y)$$




$$f^*(X) = \begin{cases} {\color{red}\bullet} & P(Y = {\color{red}\bullet} \mid X) > P(Y = {\color{green}\bullet} \mid X) \\ {\color{green}\bullet} & \text{otherwise} \end{cases}$$

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

# Classification algorithms

However, we can **learn** a good prediction rule from **training data**

$$\{(X_i, Y_i)\}_{i=1}^n \overset{\text{iid}}{\sim} P_{XY} \text{(unknown)}$$

Independent and identically distributed

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

**So far …**

Decision Trees
K-Nearest Neighbor
Naïve Bayes
Logistic Regression

# Linear Regression

Aarti Singh

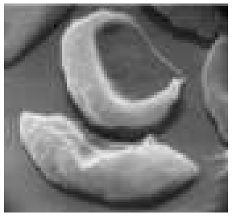
Machine Learning 10-701/15-781
Jan 27, 2010

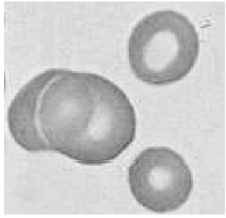
MACHINE LEARNING DEPARTMENT
Carnegie Mellon.
School of Computer Science

# Discrete to Continuous Labels

## Classification

Sports
Science
News





Anemic cell
Healthy cell

X = Document

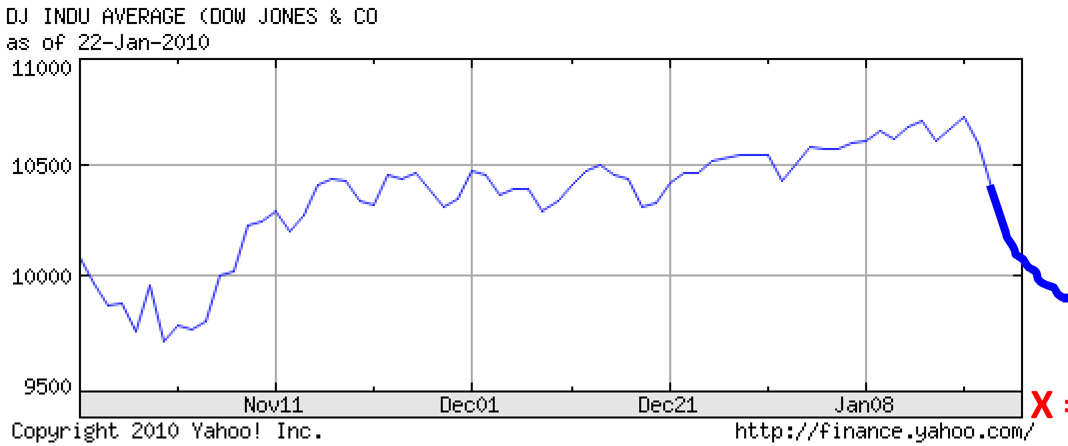
Y = Topic

X = Cell Image

Y = Diagnosis

## Regression

Stock Market Prediction

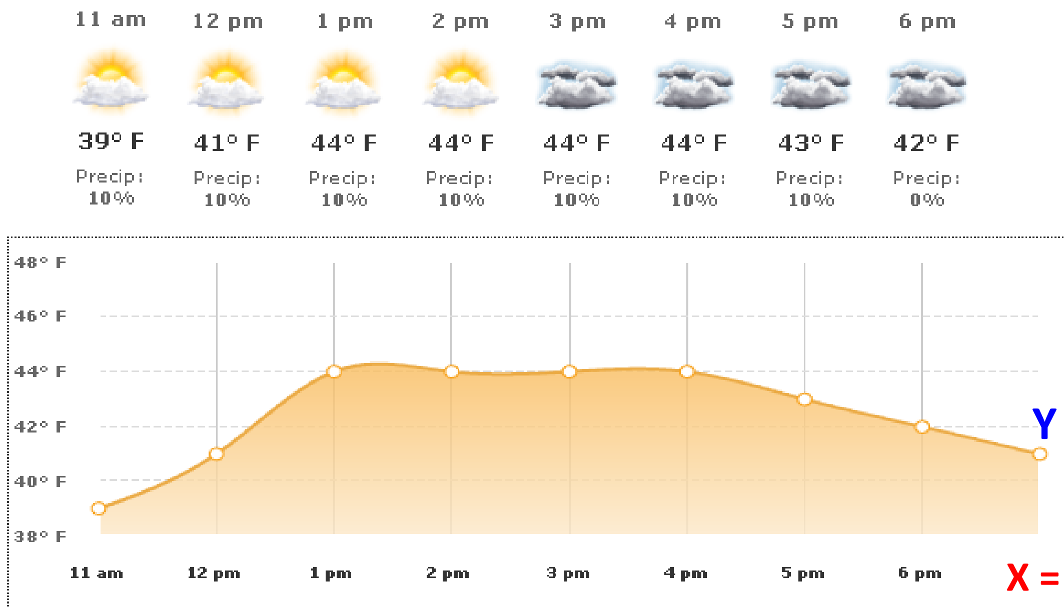



Y = ?

X = Feb01

# Regression Tasks

Weather Prediction

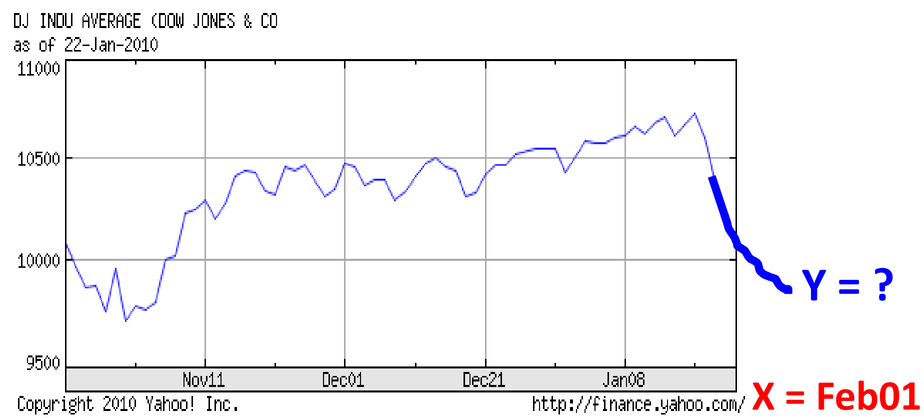



Estimating Contamination

# 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:
(Conditional Mean)

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

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

**Dropping subscripts**
**for notational convenience**

$$\begin{aligned}
&= E\left[E\left[(f(X) - E[Y|X] + E[Y|X] - Y)^2|X\right]\right] \\
&= E\big[\; E[(f(X) - E[Y|X])^2|X] \\
&\qquad +2E\left[(f(X) - E[Y|X])(E[Y|X] - Y)|X\right] \\
&\qquad +E[(E[Y|X] - Y)^2|X]\big] \\
&= E\big[\; E[(f(X) - E[Y|X])^2|X] \\
&\qquad +2(f(X) - E[Y|X]) \times 0 \\
&\qquad +E[(E[Y|X] - Y)^2|X]\big] \\
&= E\left[(f(X) - E[Y|X])^2\right] + R(f^*).
\end{aligned}$$

Thus $R(f) \geq R(f^*)$ for any prediction rule $f$, and therefore $R^* = R(f^*)$.

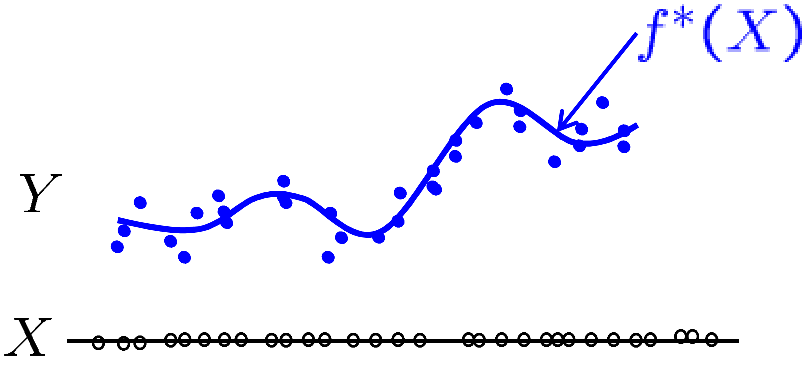
# Regression

Optimal predictor:
(Conditional Mean)

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

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 Minimizer: $$\widehat{f}_n = \arg\min_f \underbrace{\frac{1}{n}\sum_{i=1}^n}_{\text{Empirical mean}} (f(X_i) - Y_i)^2$$

# Linear Regression

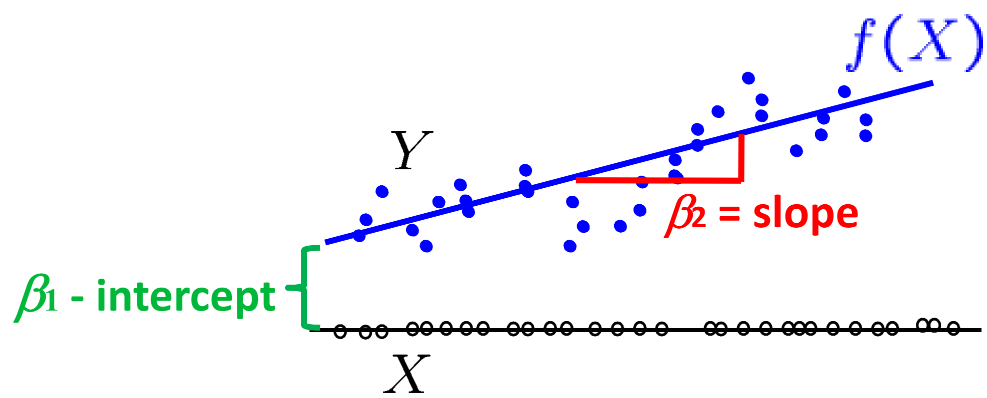
$$\widehat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Least Squares Estimator

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

Uni-variate case:

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




Multi-variate case:

1

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

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

# Least Squares Estimator

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

$$\Downarrow$$

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

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

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

# Least Squares Estimator

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

$$J(\beta) =$$

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0$$

# 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 ? (Homework 2)
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 ? (Homework 2)
Regularization (later)

# Geometric Interpretation

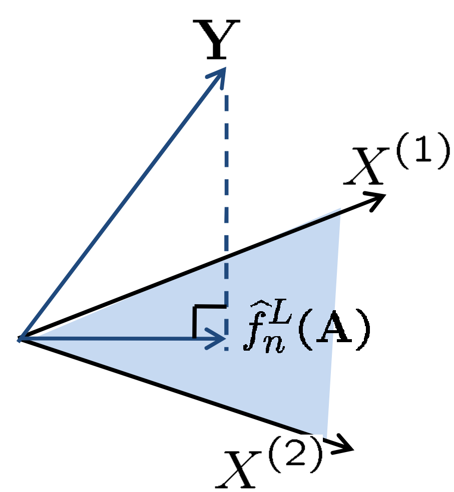
$$\widehat{f}_n^L(X) = X\widehat{\beta} = X(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y}$$

Difference in prediction on training set:

$$\widehat{f}_n^L(\mathbf{A}) - \mathbf{Y} =$$

$$\mathbf{A}^T(\widehat{f}_n^L(\mathbf{A}) - \mathbf{Y}) = 0$$

$\widehat{f}_n^L(\mathbf{A})$ is the orthogonal projection of $\mathbf{Y}$ onto the linear subspace spanned by the columns of $\mathbf{A}$



# Revisiting Gradient Descent

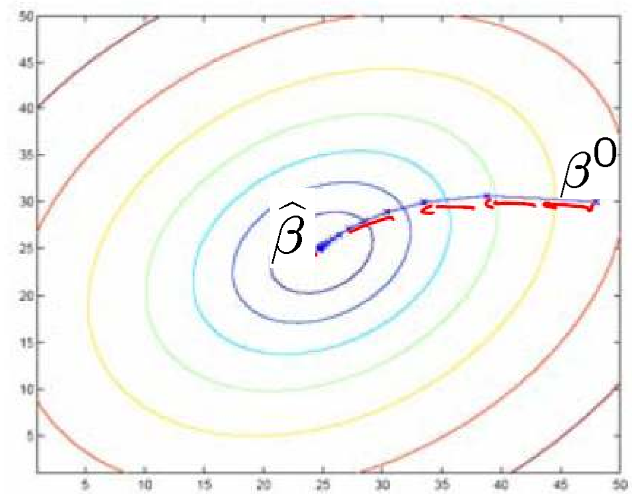
Even when $(\mathbf{A}^T\mathbf{A})$ is invertible, might be computationally expensive if **A** is huge.

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

## Gradient Descent

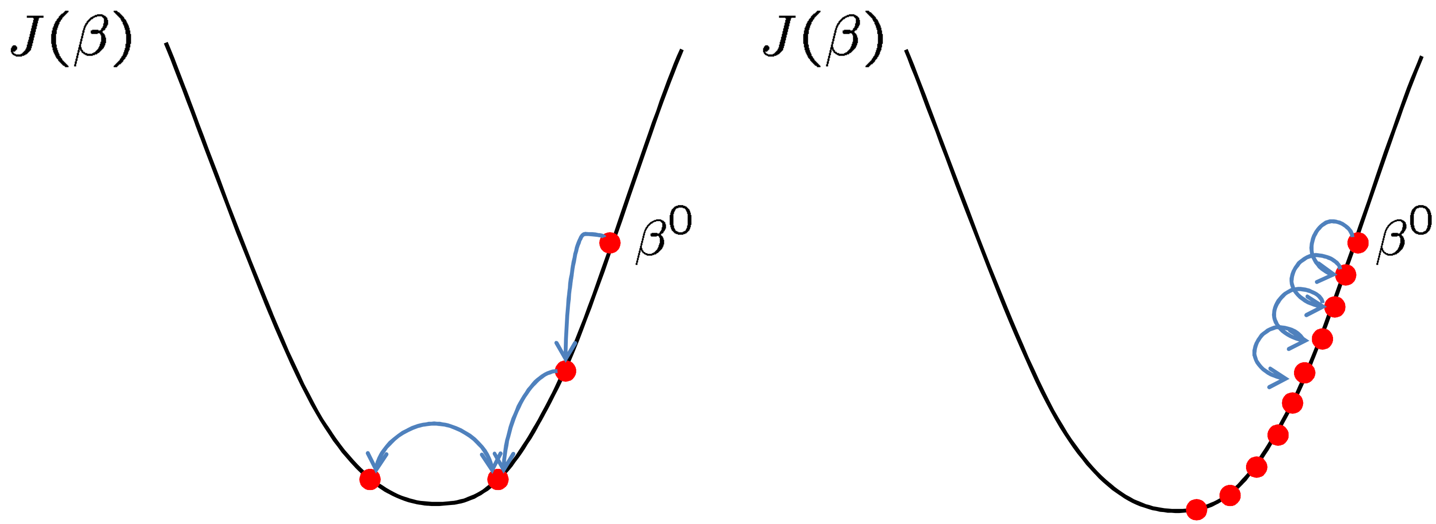
Initialize: $\beta^0$

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




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

# Effect of step-size α

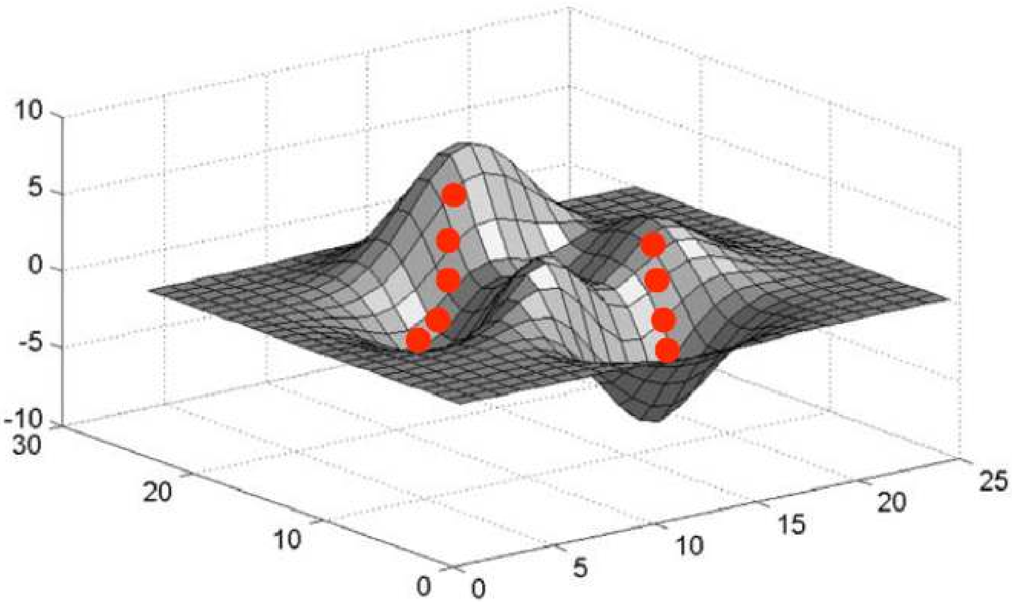



Large α => Fast convergence but larger residual error
Also possible oscillations

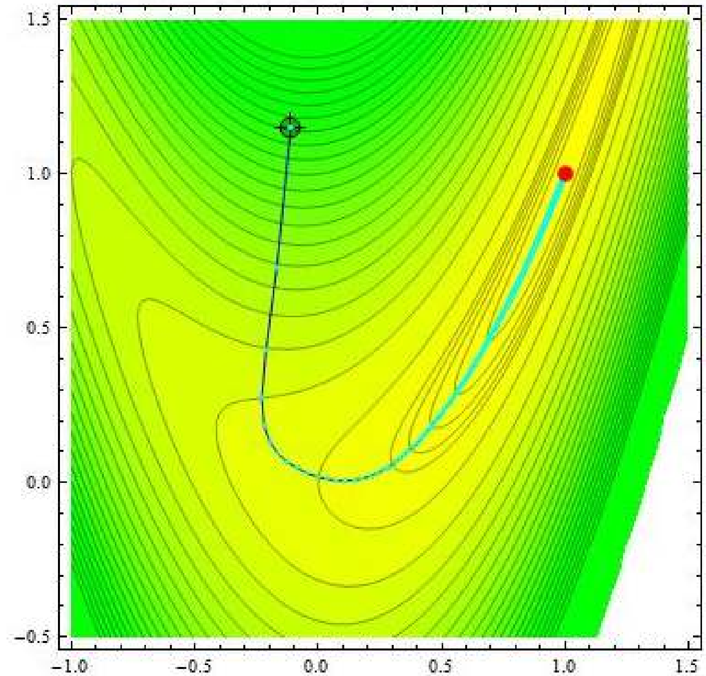
Small α => Slow convergence but small residual error

# When does Gradient Descent succeed?

View of the algorithm is myopic.




http://www.ce.berkeley.edu/~bayen/




http://demonstrations.wolfram.com

Guaranteed to converge to local minima if

$$0 < \alpha < \frac{2}{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}$$

Converges as $(1 - \alpha\lambda_j)^t$ in jth direction
Convergence depends on eigenvalue spread

# Least Squares and MLE

Intuition: Signal plus (zero-mean) Noise model

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

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

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

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

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

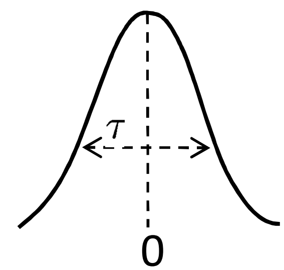
# Regularized Least Squares and MAP

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

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




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

Ridge Regression

Closed form: HW

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

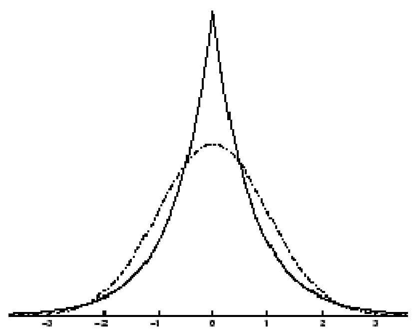
# Regularized Least Squares and MAP

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

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

II) Laplace Prior

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



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

Lasso

Closed form: HW

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

# Ridge Regression vs Lasso

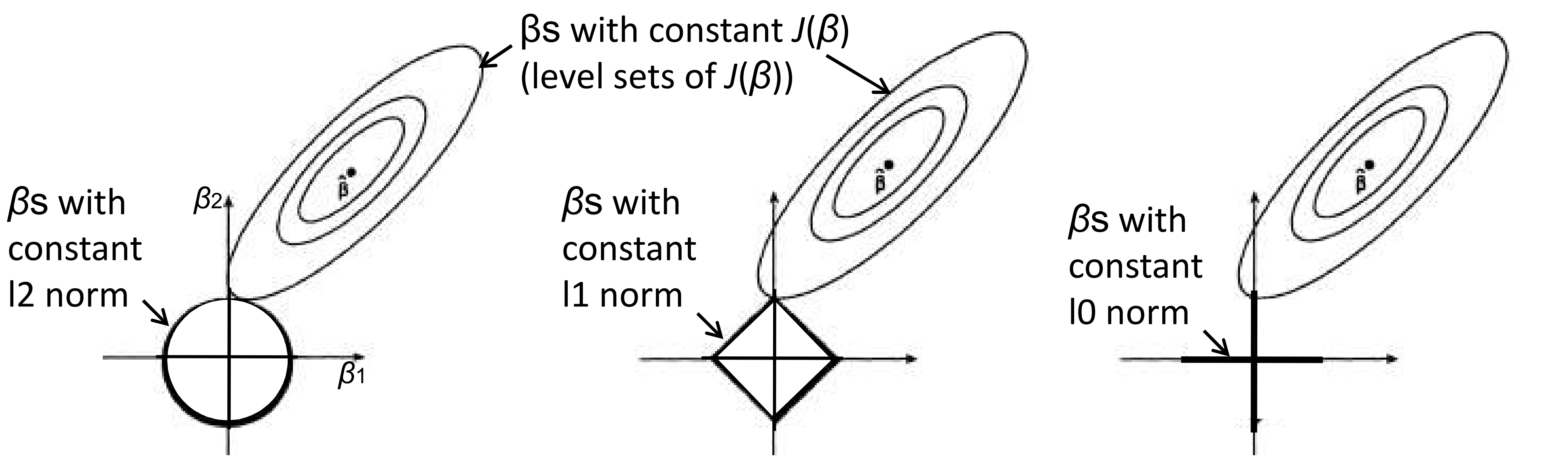
$$\min_{\beta}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) + \lambda \mathrm{pen}(\beta) = \min_{\beta} J(\beta) + \lambda \mathrm{pen}(\beta)$$

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

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

HOT!

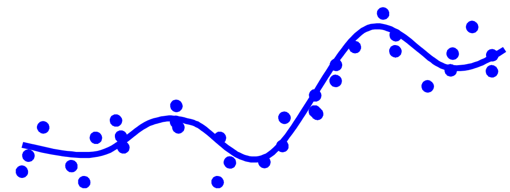
Ideally l0 penalty, but optimization becomes non-convex



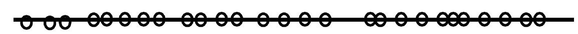

**Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates**
**Good for high-dimensional problems – don't have to store all coordinates!**

# Beyond Linear Regression

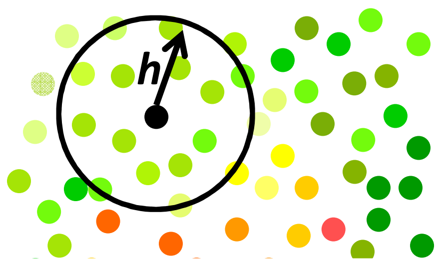
Polynomial regression



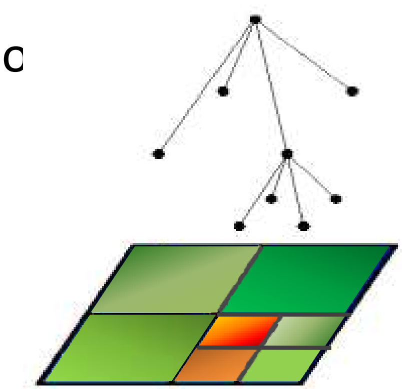
Regression with nonlinear features/basis functions



Kernel regression - Local/Weighted regression




Regression trees – Spatially adaptive regressic



# Polynomial Regression

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

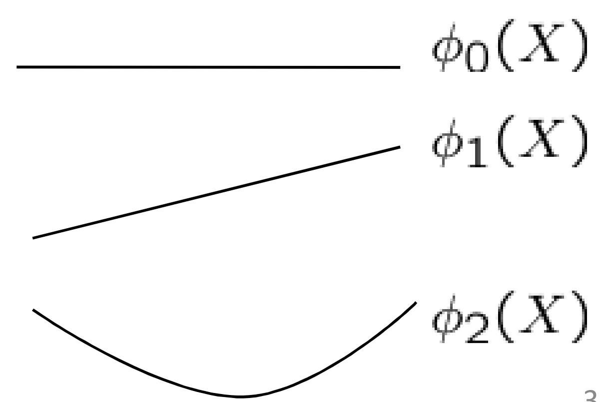
where $\mathbf{X} = [1 \ X \ X^2 \ldots X^m]$, $\beta = [\beta_1 \ldots \beta_m]^T$

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

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

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

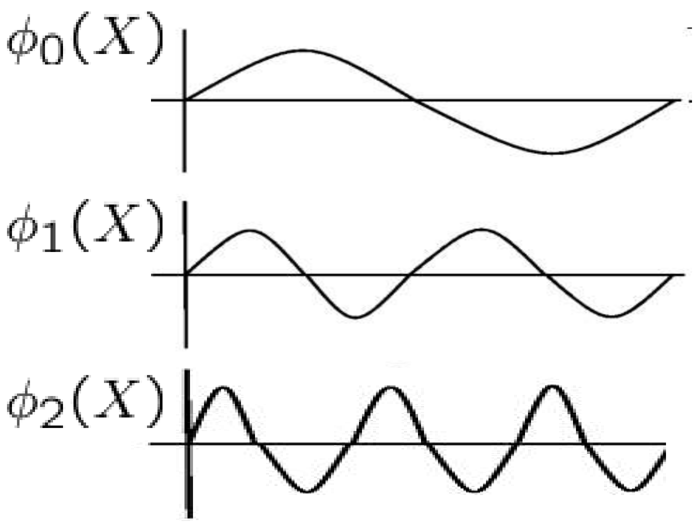
$$f(X) = \textstyle\sum_{j=0}^{m} \beta_j X^j = \sum_{j=0}^{m} \underbrace{\beta_j}_{\text{Weight of each feature}} \underbrace{\phi_j(X)}_{\text{Nonlinear features}}$$



# Nonlinear Regression

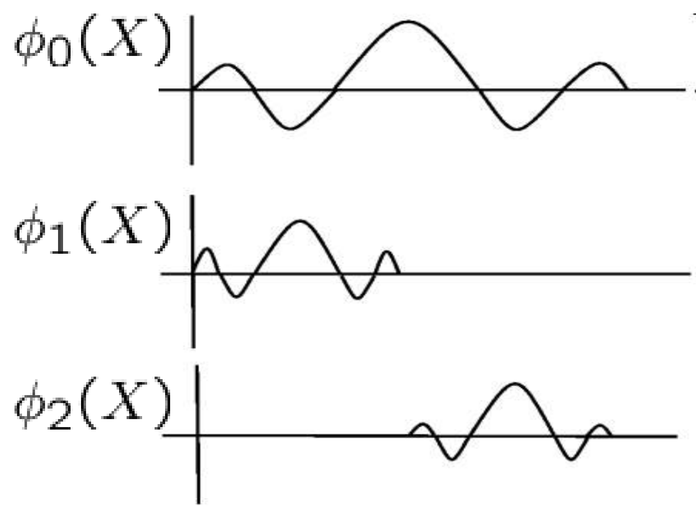
$$f(X) = \sum_{j=0}^{m} \underbrace{\beta_j}_{\text{Basis coefficients}} \underbrace{\phi_j(X)}_{\text{Nonlinear features/basis functions}}$$

Fourier Basis




Good representation for oscillatory functions

Wavelet Basis

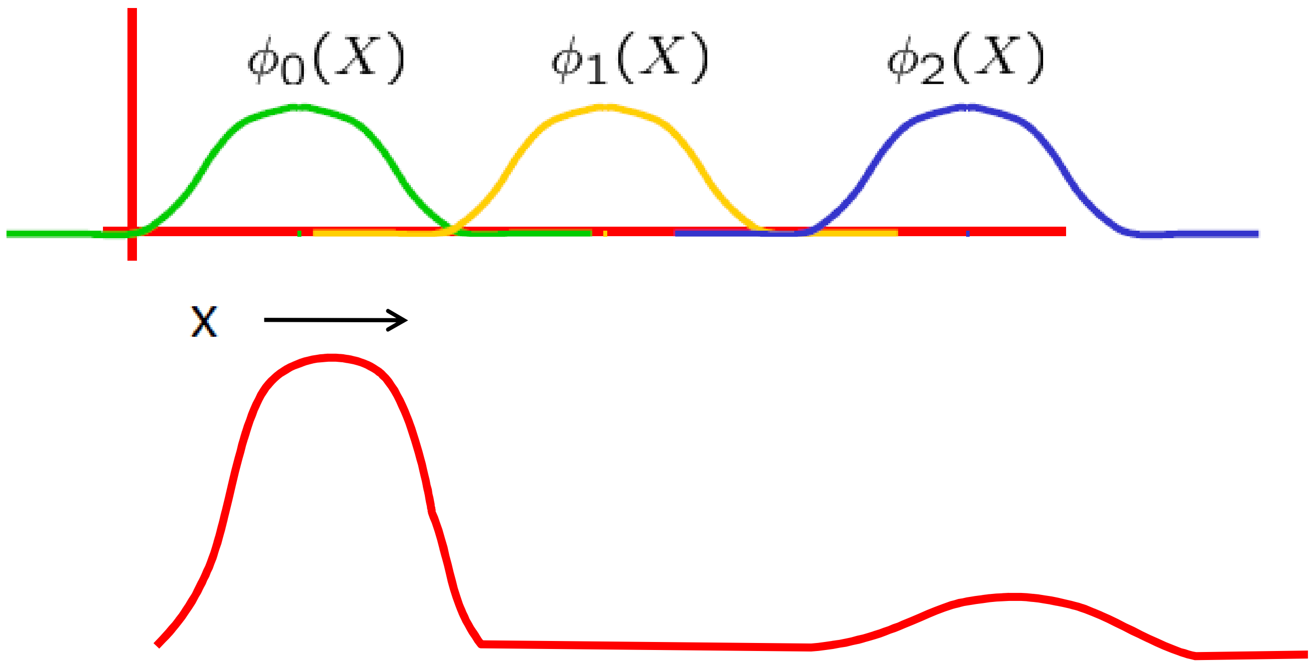



Good representation for functions localized at multiple scales

# Local Regression

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

Basis coefficients ← Nonlinear features/basis functions

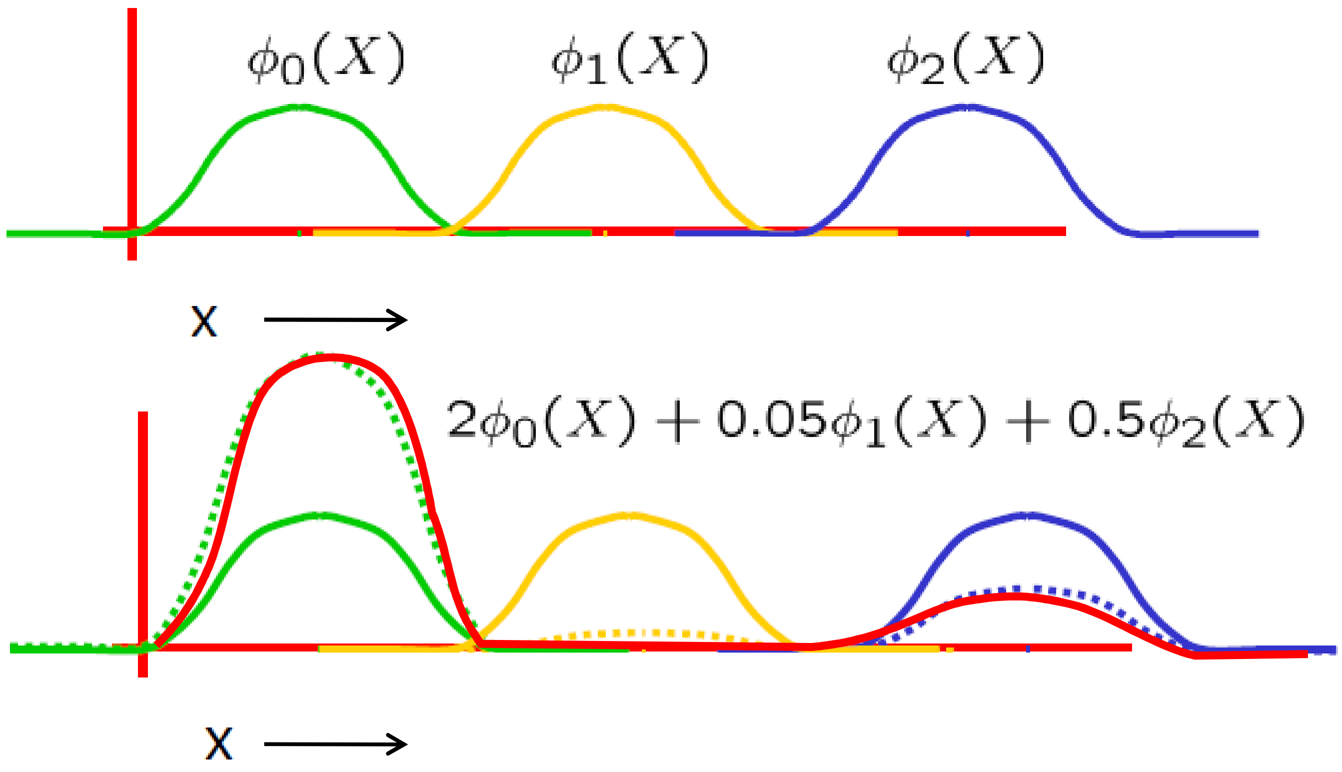



Globally supported basis functions (polynomial, fourier) will not yield a good representation

# 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

# Kernel Regression (Local)

$$\min_f \frac{1}{n} \sum_{i=1}^{n} w_i (f(X_i) - Y_i)^2 \qquad \frac{1}{n} \sum_{i=1}^{n} w_i = 1$$

Weighted Least Squares

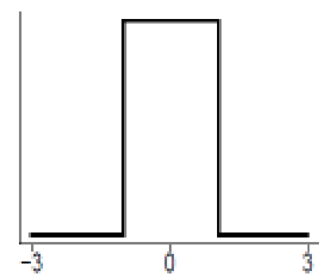
Weigh each training point based on distance to test point

$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

*K* – Kernel
*h* – Bandwidth of kernel

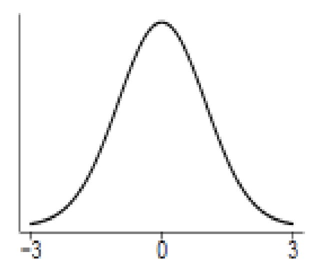
boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$




Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



# Nadaraya-Watson Kernel Regression

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} w_i(\beta - Y_i)^2$$

constant

$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

$$\frac{\partial J(\beta)}{\partial \beta} = 2 \sum_{i=1}^{n} w_i(\beta - Y_i) = 0$$

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

# Nadaraya-Watson Kernel Regression

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} w_i(\beta - Y_i)^2$$

constant

$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

boxcar kernel :

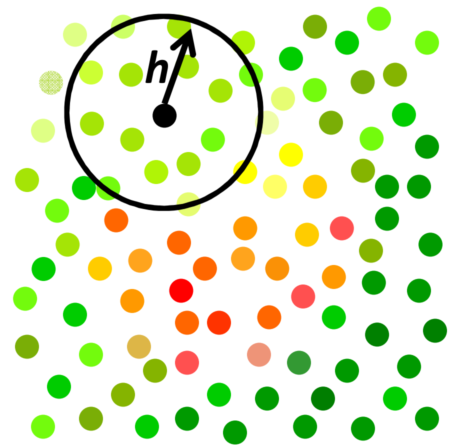
$$K\left(\frac{X-X_i}{h}\right) = \mathbf{1}_{|X-X_i|\le h}$$

$$\frac{\partial J(\beta)}{\partial \beta} = 2\sum_{i=1}^{n} w_i(\beta - Y_i) = 0$$

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

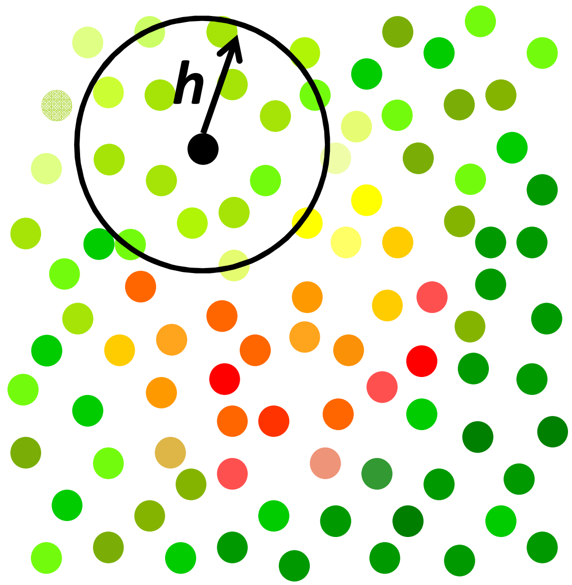
with box-car kernel $= \dfrac{1}{n_X^h} \underbrace{\sum_{i=1}^{n} Y_i \ \mathbf{1}_{|X-X_i|\le h}}_{\text{Sum of Ys in h ball around X}}$

#pts in h ball around X




Recall: NN classifier
Average <-> majority vote

# Choice of Bandwidth




Should depend on n, # training data
(determines variance)

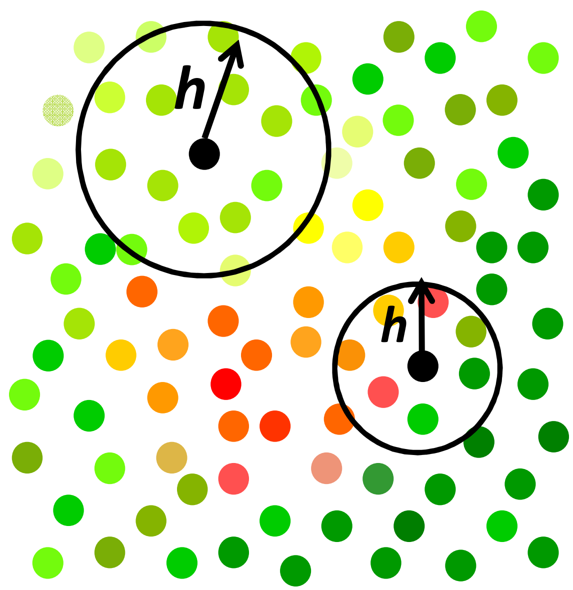
Should depend on smoothness of function
(determines bias)

Large Bandwidth – average more data points, reduce noise (Lower variance)

Small Bandwidth – less smoothing, more accurate fit (Lower bias)

Bias – Variance tradeoff : More to come in later lectures

# Spatially adaptive regression




If function smoothness varies spatially, we want to allow bandwidth h to depend on X

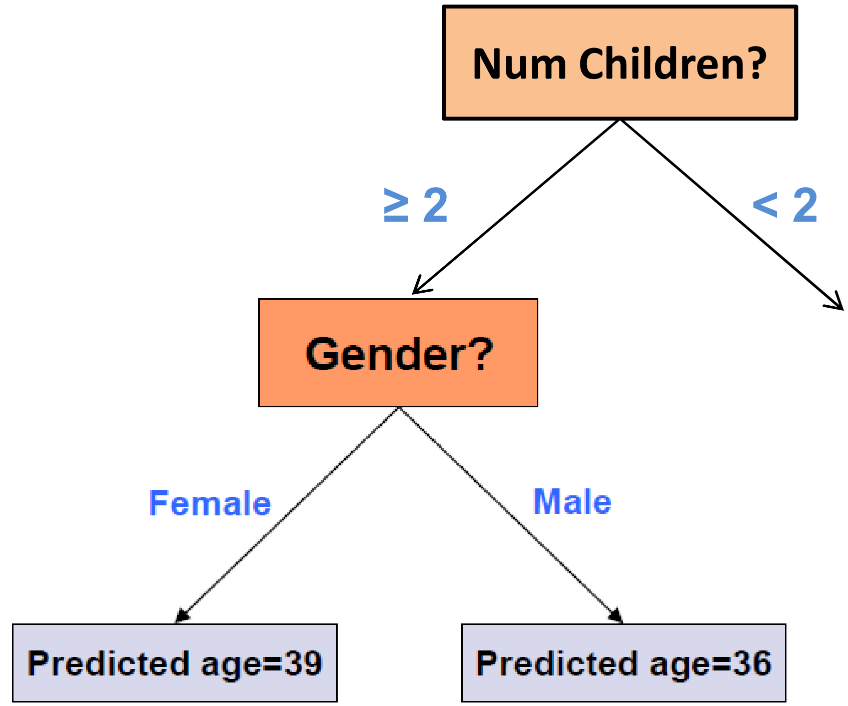
Local polynomials, splines, wavelets, regression trees …

# Regression trees

$X^{(1)}$ .... $X^{(p)}$ $Y$

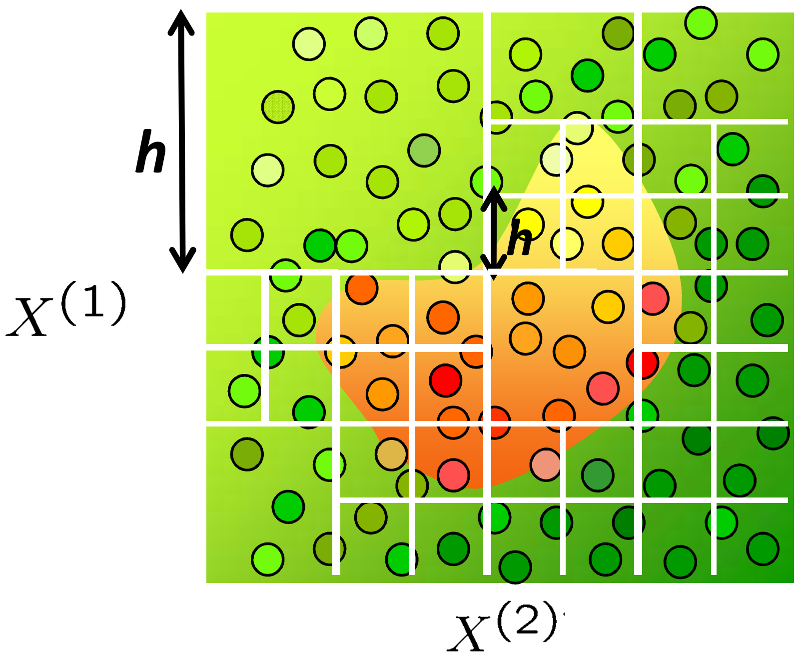
| Gender | Rich? | Num. Children | # travel per yr. | Age |
|---|---|---|---|---|
| F | No | 2 | 5 | 38 |
| M | No | 0 | 2 | 25 |
| M | Yes | 1 | 0 | 72 |
| : | : | : | : | : |

Binary Decision Tree

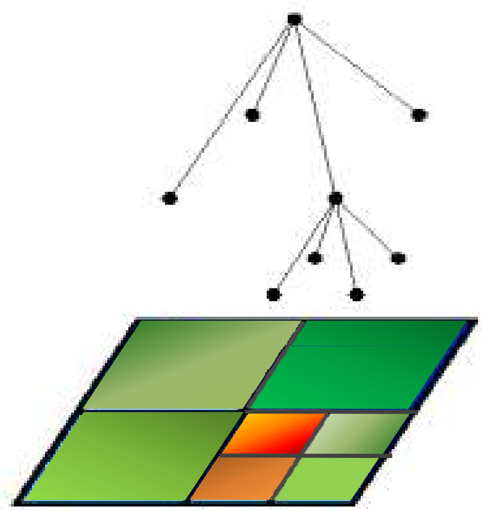



Average (fit a constant ) on the leaves

# Regression trees




Quad Decision Tree



$f$ - Polynomial fit on each leaf

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

If $\displaystyle \sum_{\text{small cells}, L} \sum_{i \in L} (\widehat{f}(X_i) - Y_i)^2 < \sum_{i \in \text{merged cell}} (\widehat{f}(X_i) - Y_i)^2$, then split

Else stop

**Compare residual error with and without split**

# Summary

Discriminative vs Generative Classifiers

- Naïve Bayes vs Logistic Regression

Regression

- Linear Regression
  - Least Squares Estimator
  - Normal Equations
  - Gradient Descent
  - Geometric Interpretation
  - Probabilistic Interpretation (connection to MLE)
- Regularized Linear Regression (connection to MAP)
  - Ridge Regression, Lasso
- Polynomial Regression, Basis (Fourier, Wavelet) Estimators
- Kernel Regression (Localized)
- Regression Trees