Machine learning

regression

Mohammad-Reza A. Dehaqani

dehaqani@ut.ac.ir

Slides mainly adopted form Andrew Ng and Aarti course

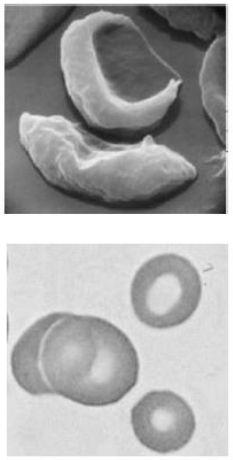
Supervised Learning Tasks

Classification

X = Document

Sports
Science
News

Y = Topic

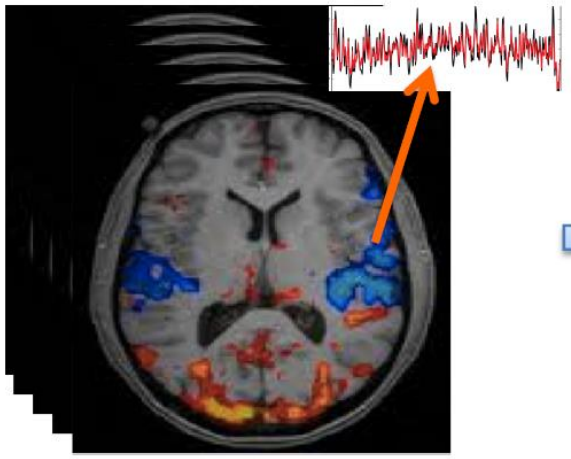


X = Cell Image

Anemic cell
Healthy cell

Y = Diagnosis

Regression

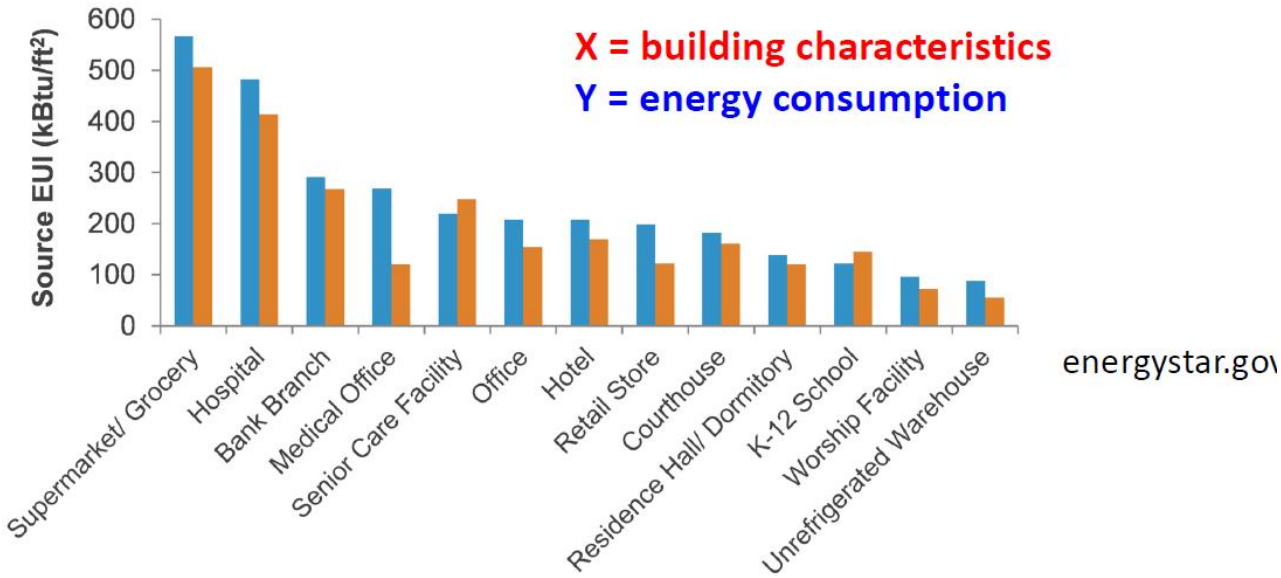


X = Brain Scan

Y = Age of a subject

Regression Tasks

Estimating Energy Usage



Estimating Contamination

Performance Measures

Performance Measure: Quantifies knowledge gained

$\text{loss}(Y, f(X))$ - Measure of closeness between true label *Y* and prediction *f*(*X*)

Don't just want label of one test data (cell image), but any cell image $X \in \mathcal{X}$

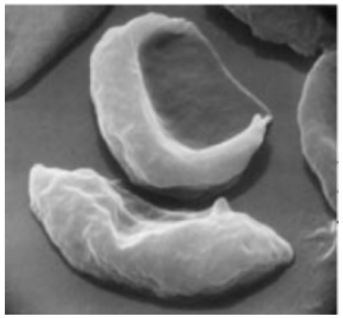
$$(X, Y) \sim P_{XY}$$

Given a cell image drawn randomly from the collection of all cell images, how well does the predictor perform on average?

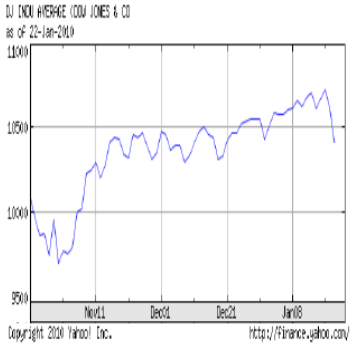
$$\text{Risk } R(f) \equiv \mathbb{E}_{XY}\left[\text{loss}(Y, f(X))\right]$$

Performance Measures

Performance Measure: Risk $R(f) \equiv \mathbb{E}_{XY}\left[\text{loss}(Y, f(X))\right]$



"Anemic cell"



Share Price
"$ 24.50"

loss$(Y, f(X))$	Risk $R(f)$
$1_{\{f(X)\neq Y\}}$	$P(f(X) \neq Y)$
0/1 loss	**Probability of Error**
$(f(X) - Y)^2$	$\mathbb{E}[(f(X) - Y)^2]$
square loss	**Mean Square Error**

Bayes Optimal Rule

Ideal goal: Construct **prediction rule** $f^* : \mathcal{X} \to \mathcal{Y}$

$$f^* = \arg\min_f \mathbb{E}_{XY}\left[\text{loss}(Y, f(X))\right]$$

Bayes optimal rule

Best possible performance:

Bayes Risk $\quad R(f^*) \leq R(f)$ for all f

BUT… Optimal rule is not computable - depends on unknown P_{XY} !

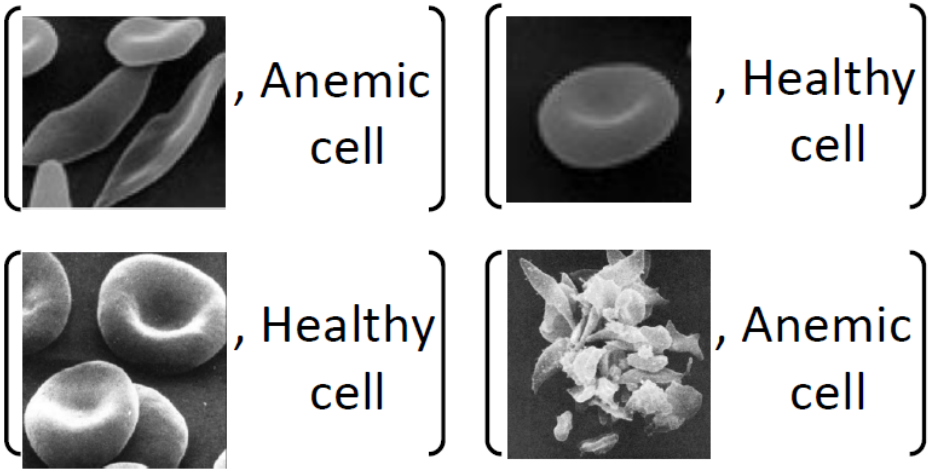
Experience - Training Data

Can't minimize risk since P_{XY} unknown!

Training data (experience) provides a glimpse of P_{XY}

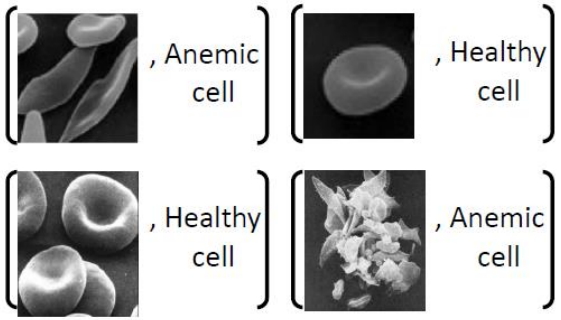
(observed) $\{(X_i, Y_i)\}_{i=1}^{n} \overset{i.i.d.}{\sim} P_{XY}$ (unknown)

independent, identically distributed



Provided by expert, measuring device, some experiment, ...

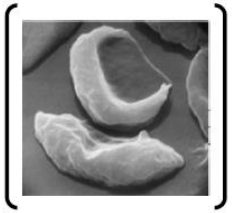
Machine Learning Algorithm



$\widehat{f}_n$

Data $\{(X_i, Y_i)\}_{i=1}^n$

$\widehat{f}_n$ is a mapping from $\mathcal{X} \to \mathcal{Y}$

$\widehat{f}_n$  = "Anemic cell"

Test data X

Empirical Risk Minimization

Optimal predictor:

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

Empirical Minimizer:

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

Empirical mean

Law of Large Numbers:

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

Restrict class of predictors

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

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

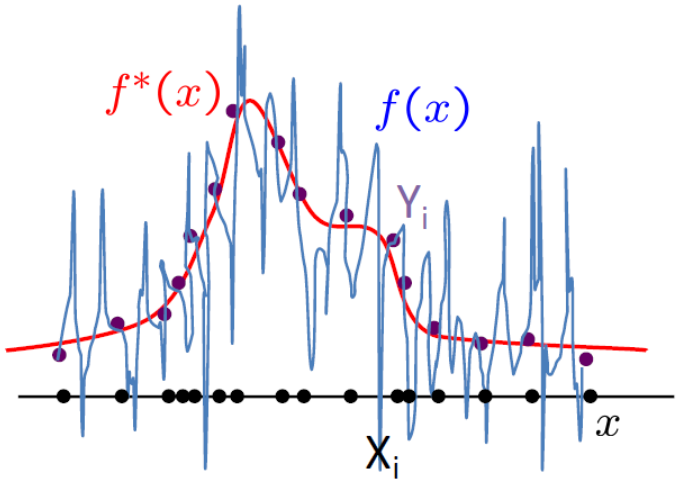
Class of predictors

Why?

Overfitting!

Empiricial loss minimized by any function of the form

$$f(x) = \begin{cases} Y_i, & x = X_i \text{ for } i = 1, \ldots, n \\ \text{any value}, & \text{otherwise} \end{cases}$$



Restrict class of predictors

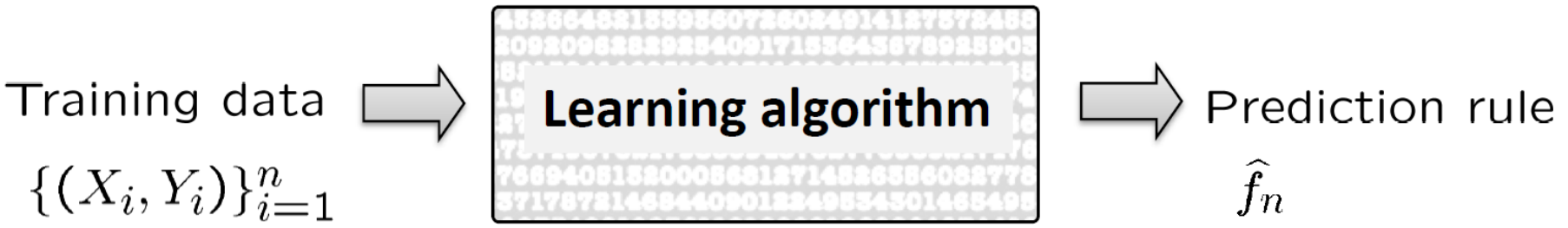
Optimal predictor: $$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

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

Class of predictors

$\mathcal{F}$ -
- Class of Linear functions
- Class of Polynomial functions
- Class of nonlinear functions

Regression algorithms



Linear Regression
Regularized Linear Regression – Ridge regression, Lasso
Polynomial Regression
Kernelized Ridge Regression
Gaussian Process Regression
Kernel regression, Regression Trees, Splines, Wavelet estimators, …

Linear Regression

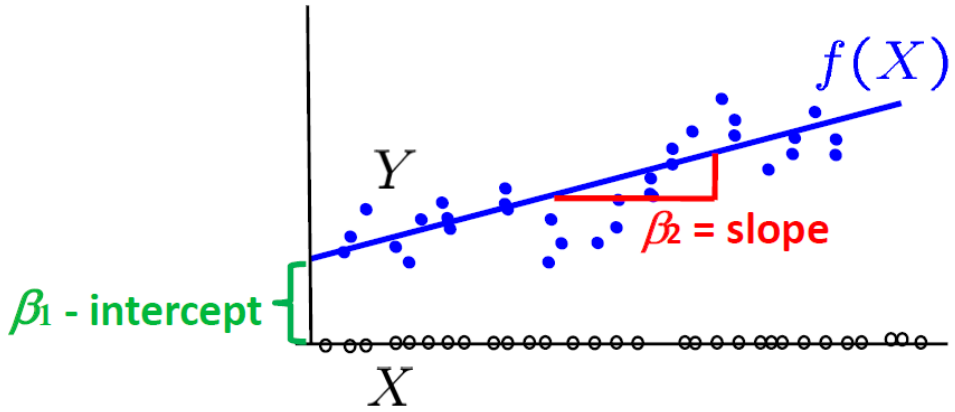
$$\hat{f}_n^L = \arg\min_{f \in \mathcal{F}_L} \frac{1}{n} \sum_{i=1}^{n} (f(X_i) - Y_i)^2$$

Least Squares Estimator

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

Uni-variate case:

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



Multi-variate case:

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

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

Least Squares Estimator

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

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

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

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

Least Squares Estimator

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

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

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

Least Square solution satisfies Normal Equations

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

p x p p x1 p x1

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

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

Least Square solution satisfies Normal Equations

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

p x p p x1 p x1

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

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

Later: When is $(\mathbf{A}^T\mathbf{A})$ invertible ?

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

Now: What if $(\mathbf{A}^T\mathbf{A})$ is invertible but expensive (p very large)?

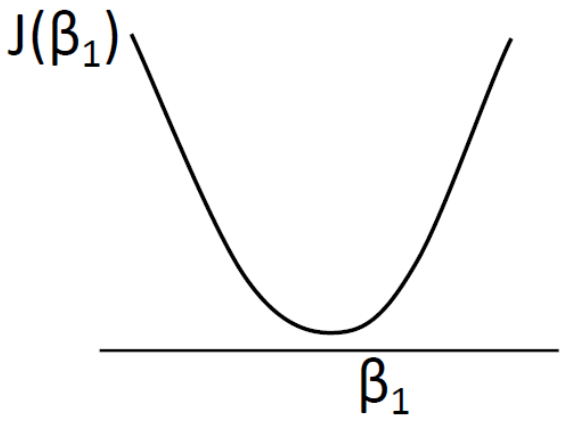
Gradient Descent

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

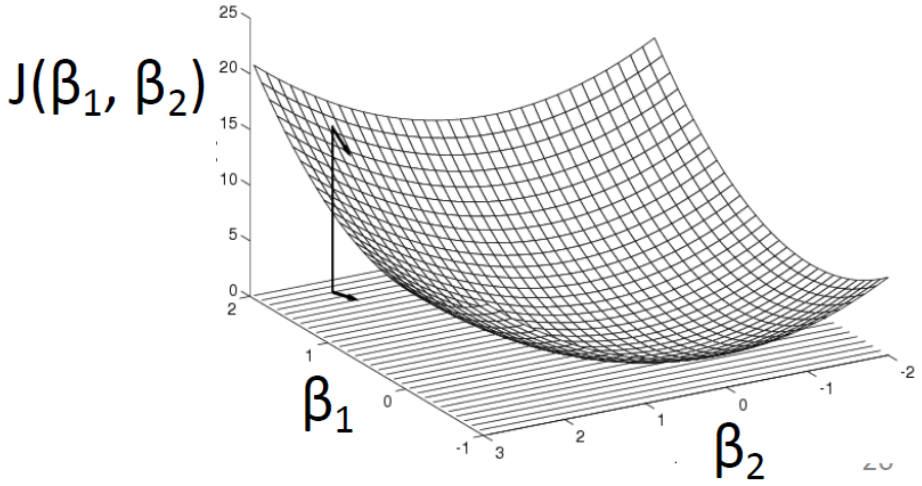
$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

Treat as optimization problem

Observation: J(β) is convex in β.



How to find the minimizer?



Gradient Descent

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

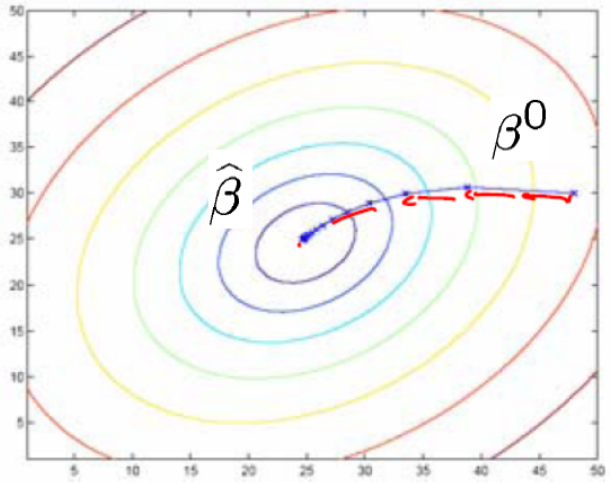
$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n}(\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) = \arg\min_{\beta} J(\beta)$$

Since J(β) is convex, move along negative of gradient

Initialize: β^0

step size

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



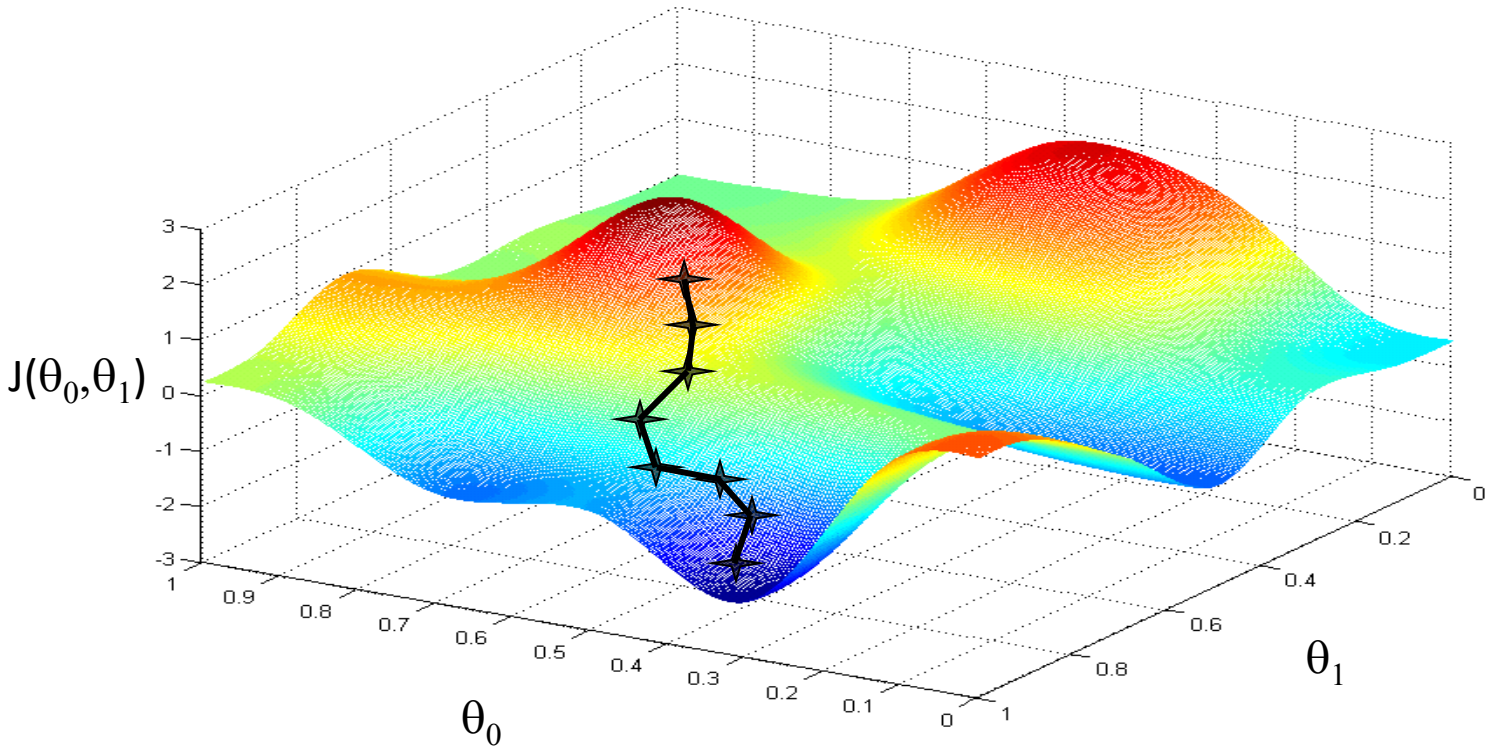
Stop: when some criterion met e.g. fixed # iterations, or $\frac{\partial J(\beta)}{\partial \beta}\Big|_{\beta^t} < \varepsilon$.

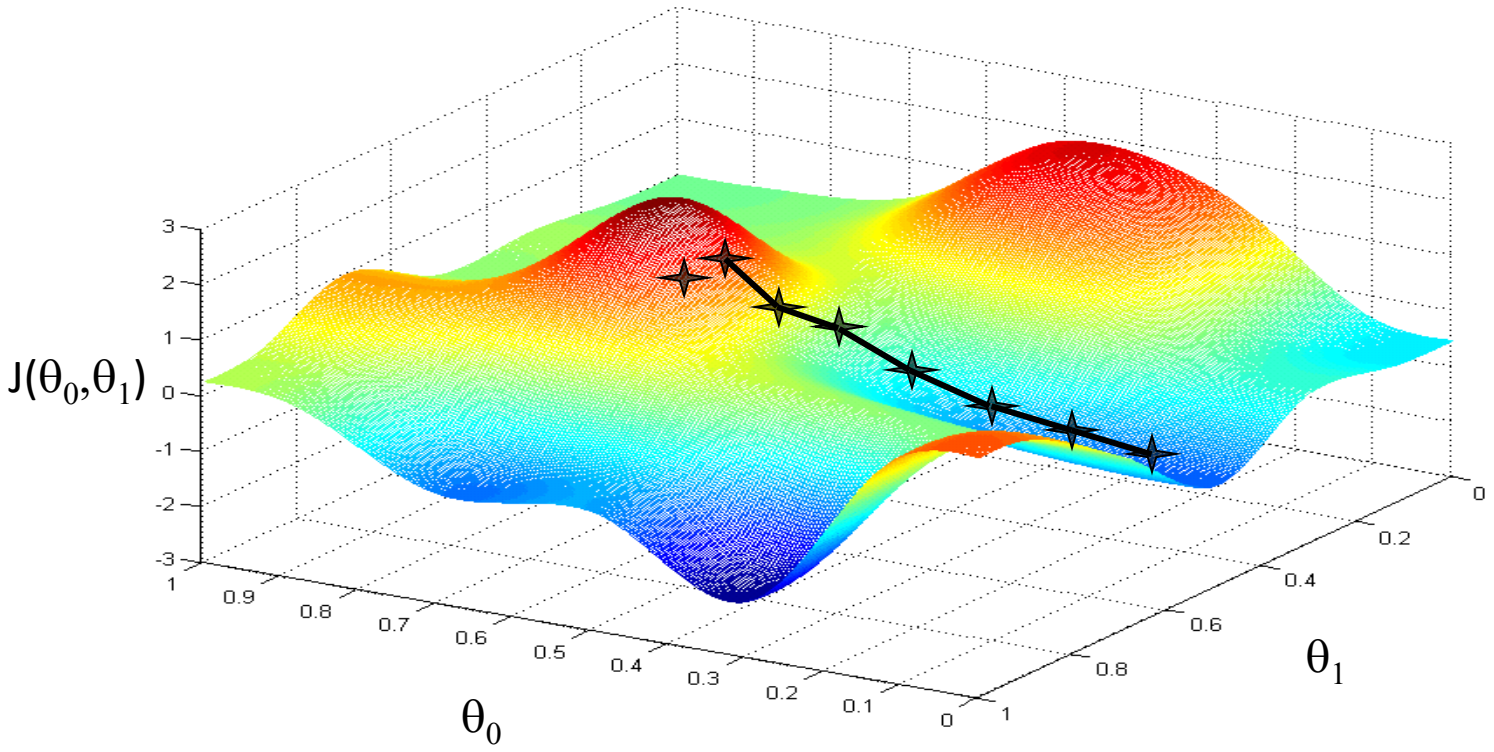
"Batch" Gradient Descent

"Batch": Each step of gradient descent uses all the training examples.

Solution II: stochastic gradient descent

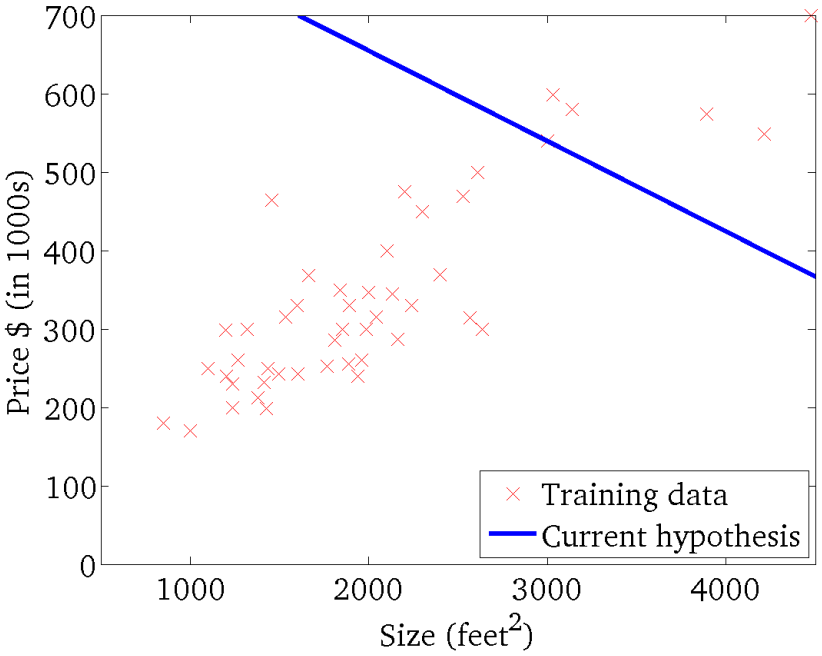
To update the parameters according to the **gradient of the error** with respect to that s**ingle training example only**.


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

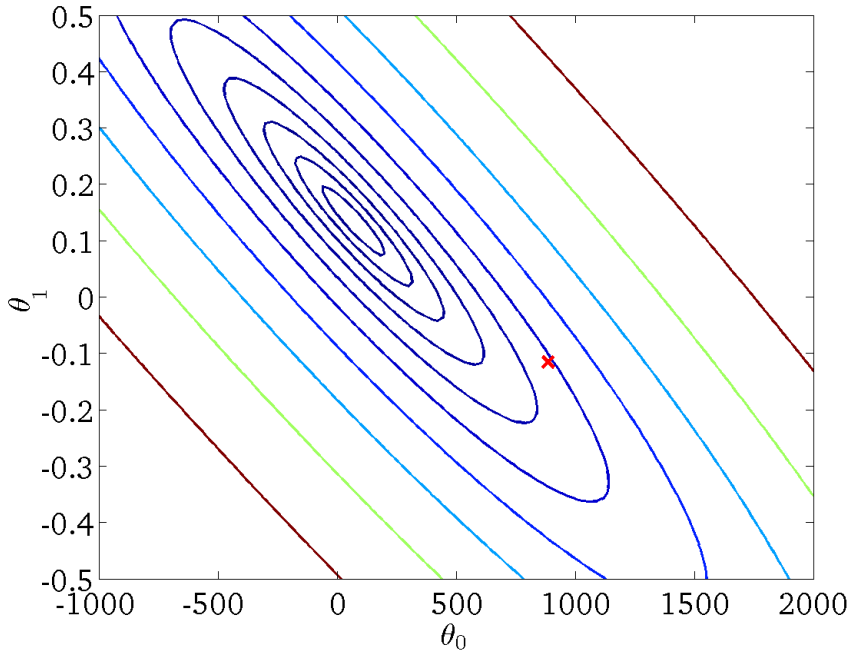
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



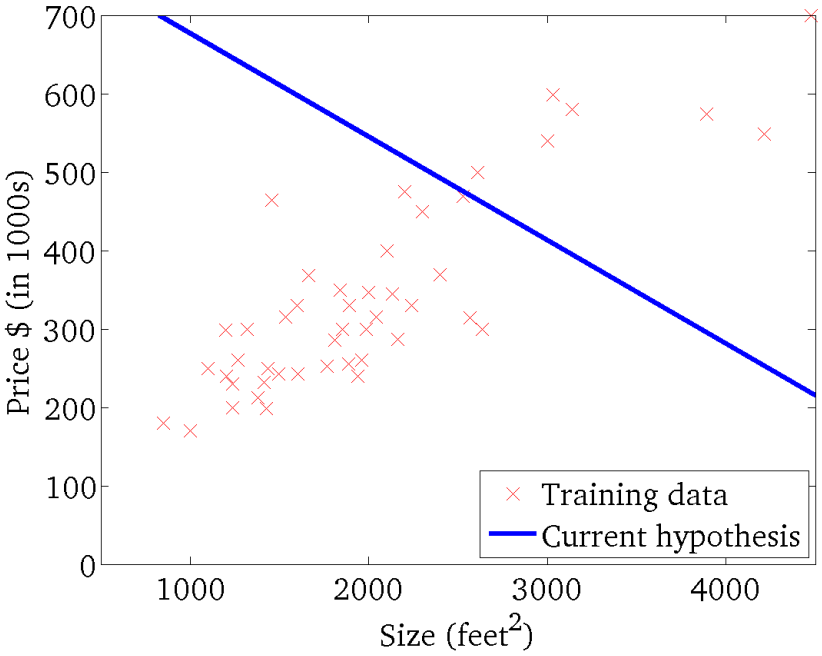
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



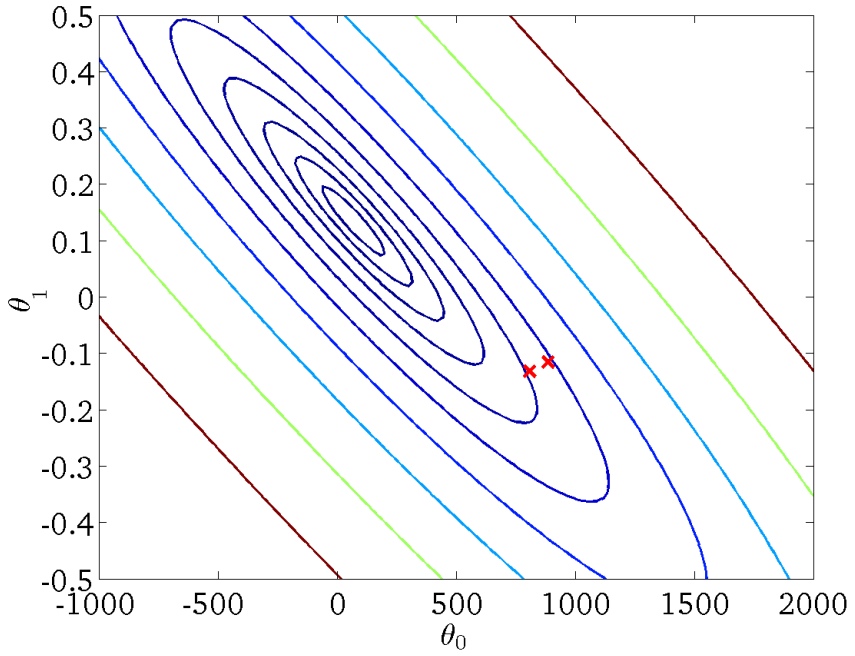
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



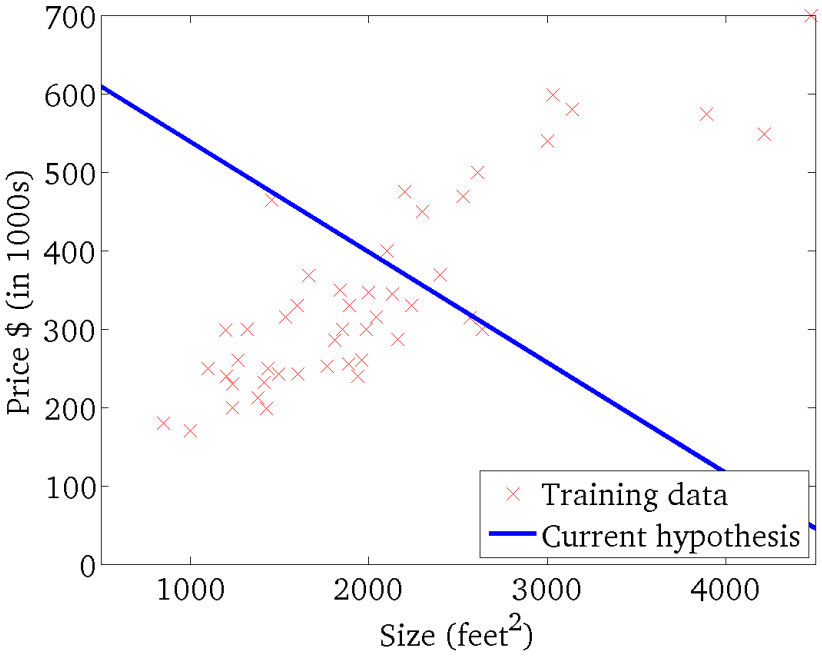
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



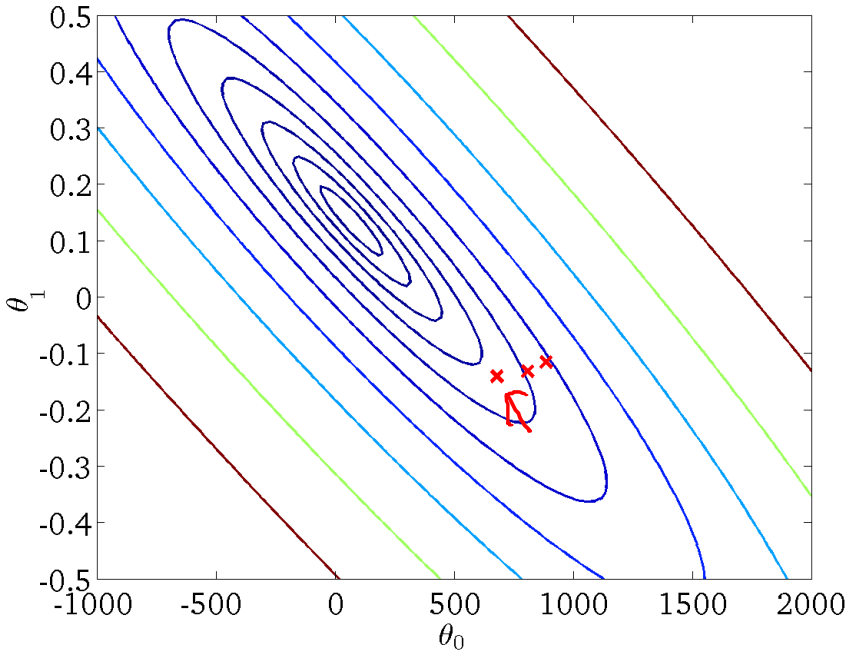
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



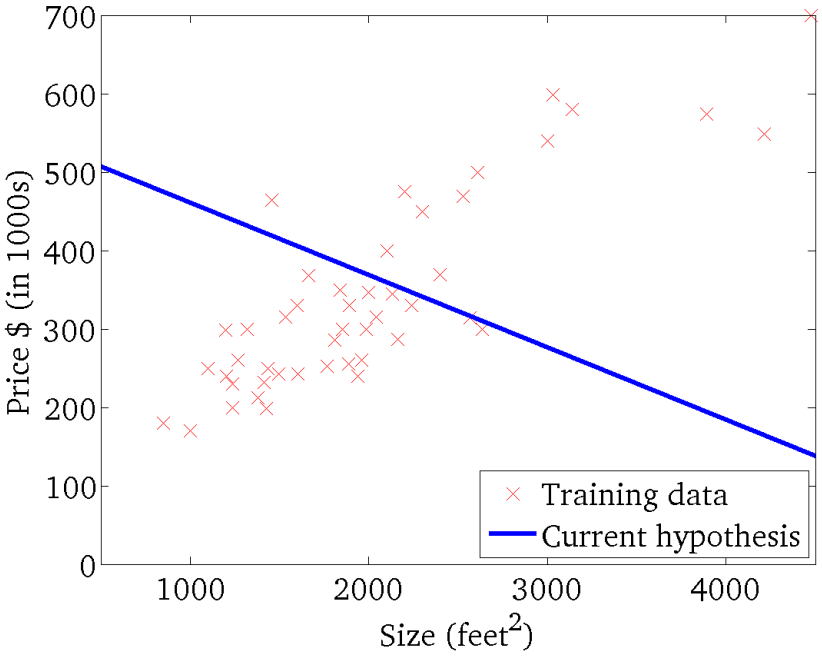
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



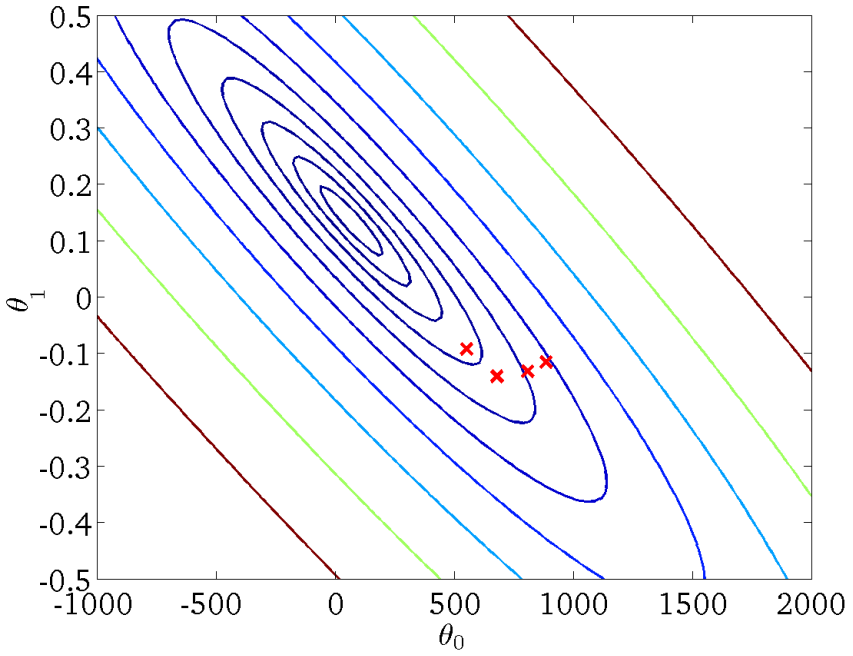
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



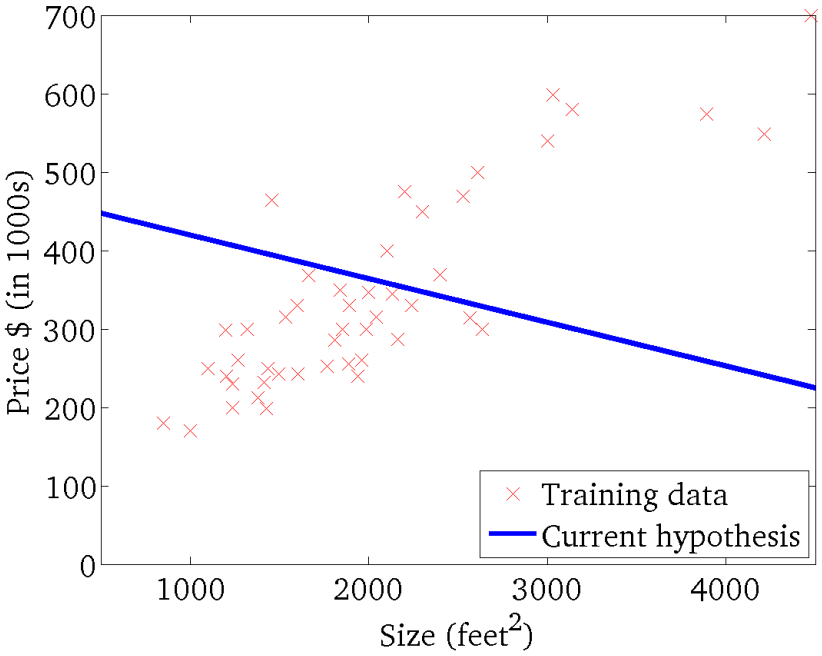
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



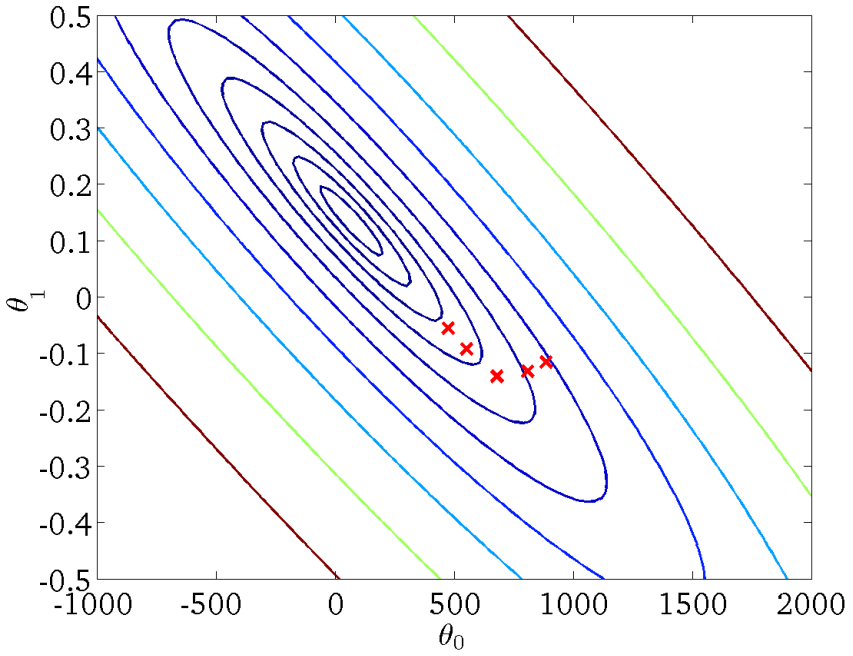
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



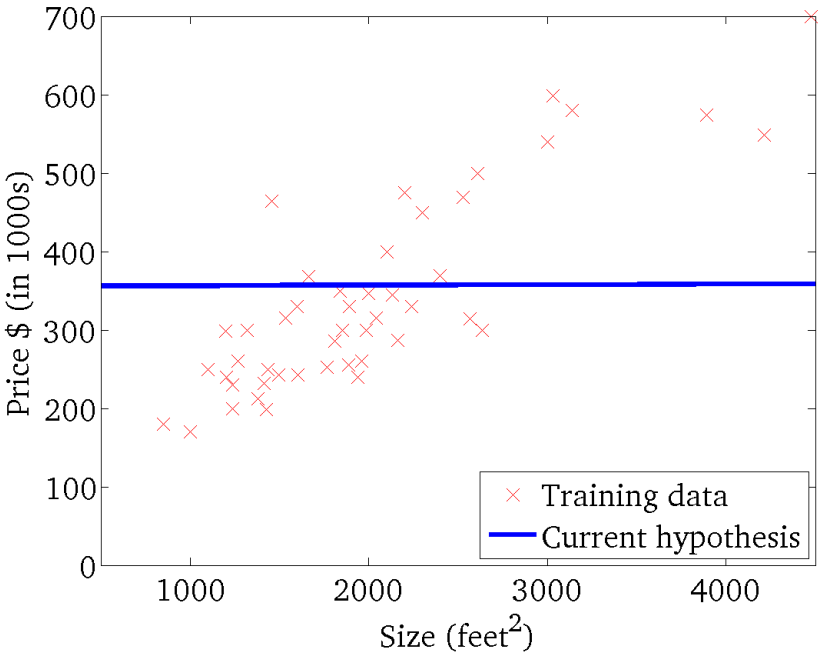
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



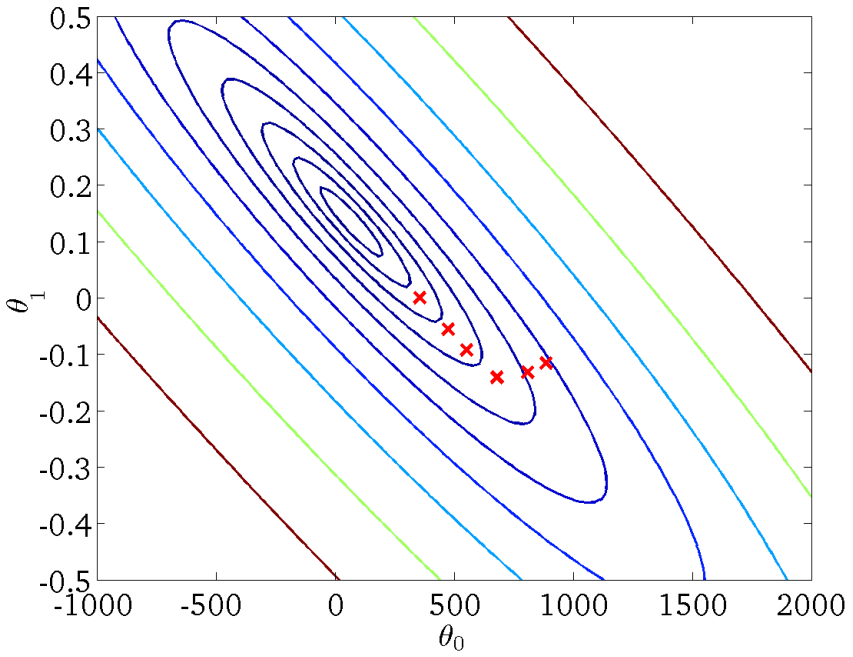
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



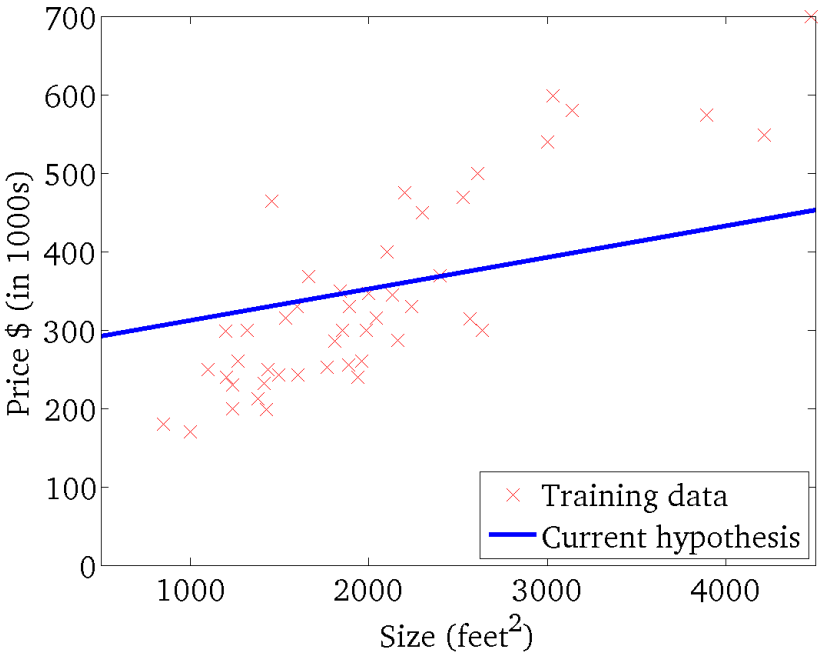
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



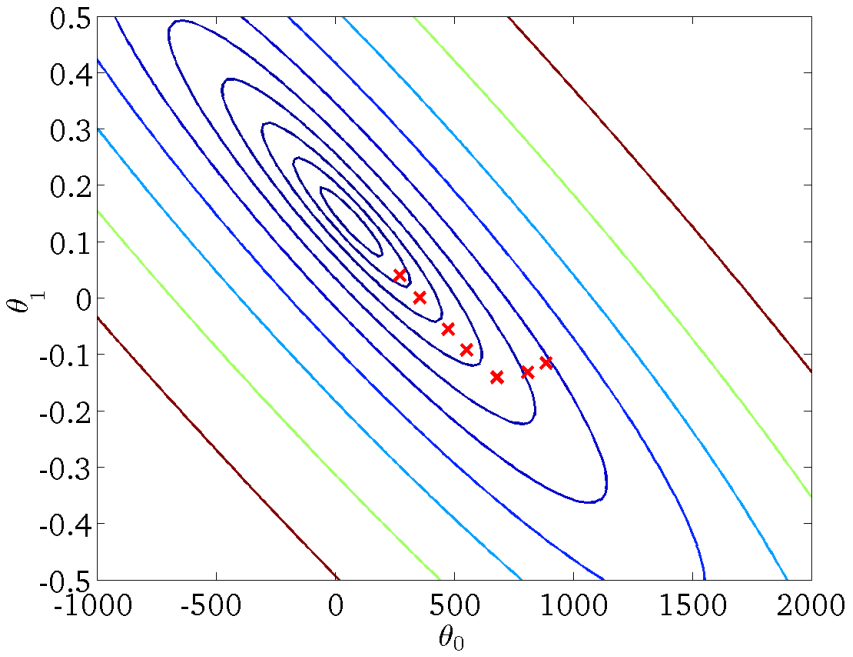
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



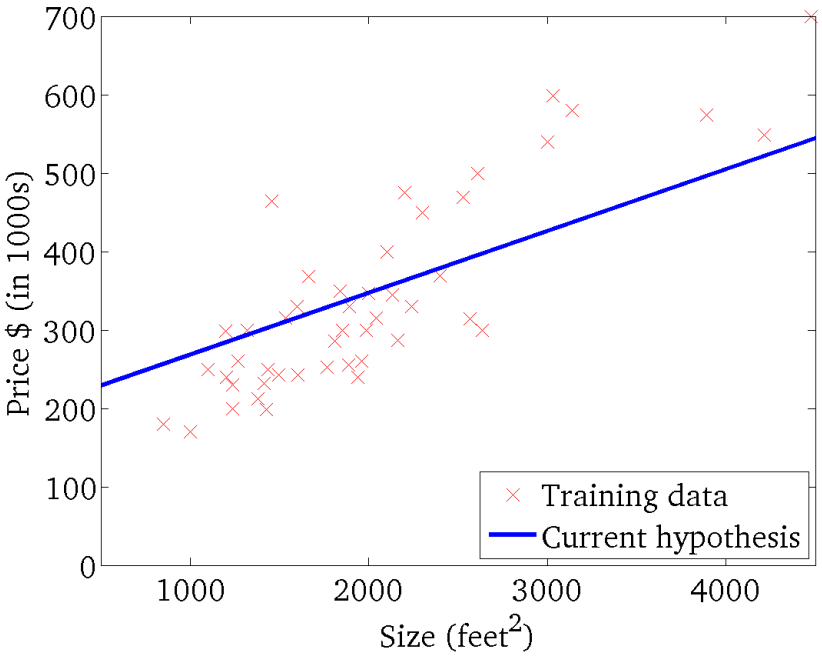
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



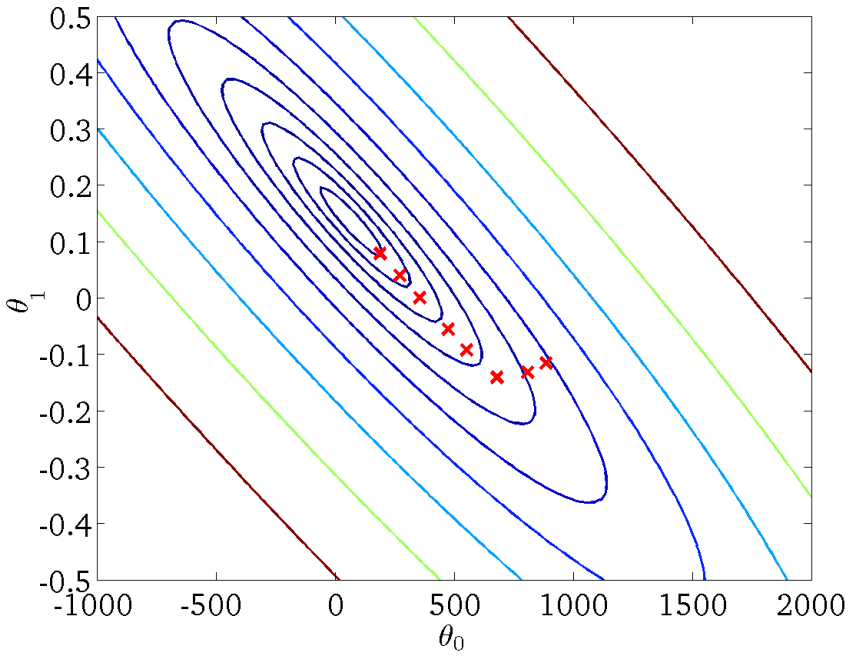
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



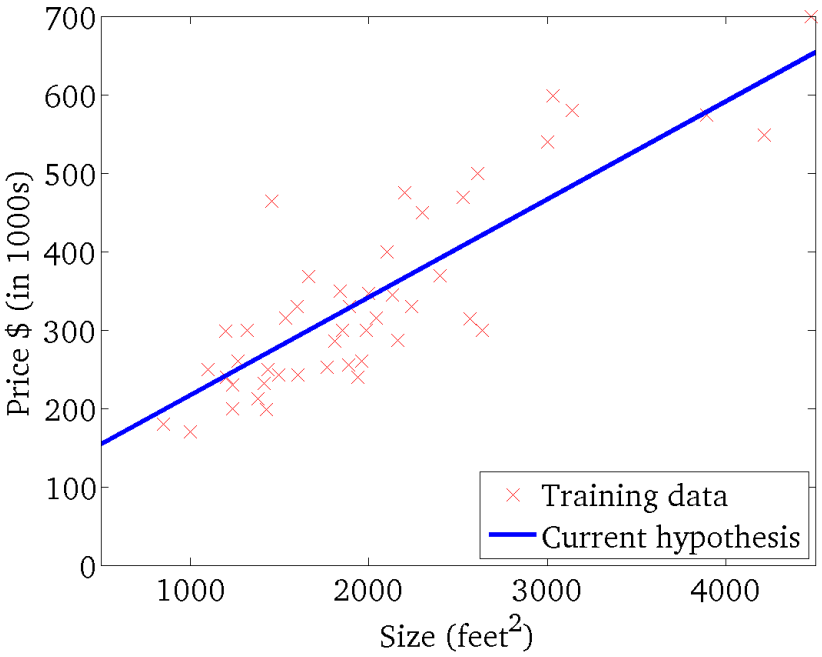
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



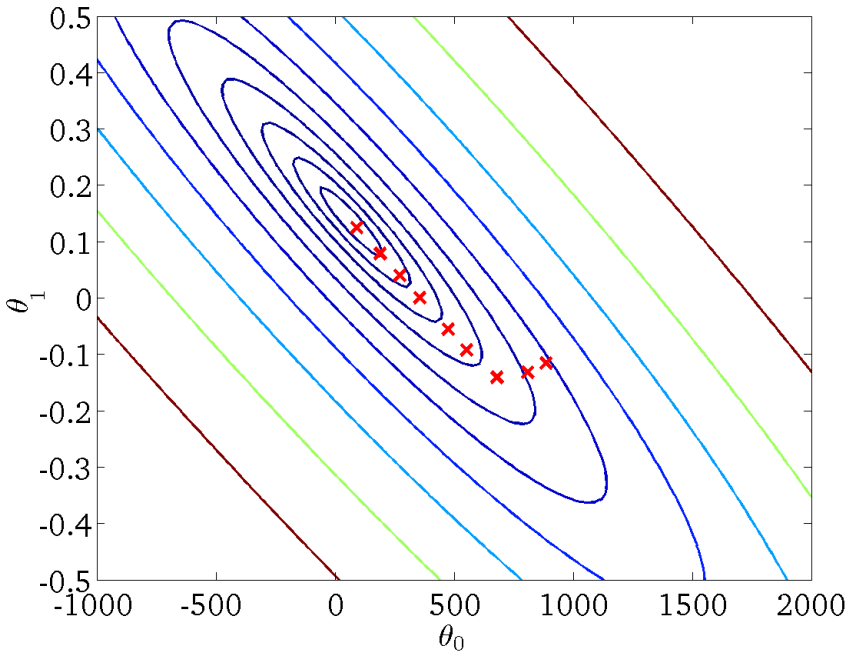
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Regularized regression

Slides adopted from Aarti course from cmu

Geometric interpretation

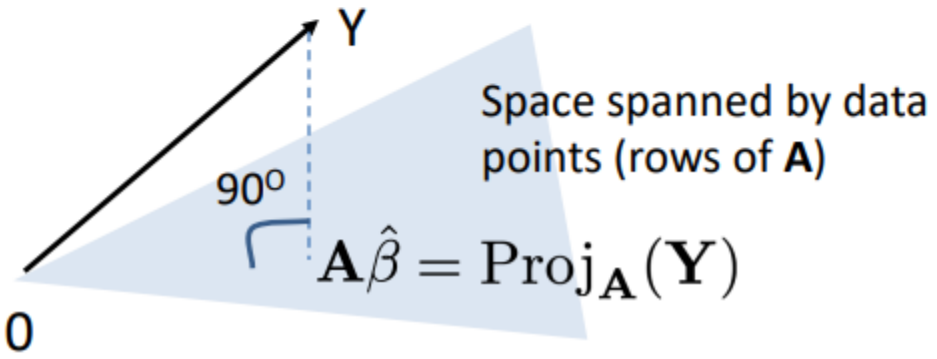
$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0 \qquad \text{gives} \qquad (\mathbf{A}^T\mathbf{A})\widehat{\beta} = \mathbf{A}^T\mathbf{Y}$$

p x p p x1 p x1

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

1) If dimension p not too large, analytical solution:

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



Gradient descent in case of large dimension

$$\left.\frac{\partial J(\beta)}{\partial \beta}\right|_{\widehat{\beta}} = 0 \qquad \text{gives} \qquad (\mathbf{A}^T\mathbf{A})\widehat{\beta} = \mathbf{A}^T\mathbf{Y}$$

p x p p x1 p x1

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

1) If dimension p not too large, analytical solution:

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

2) If dimension p is large, computing inverse is expensive $O(p^3)$
Gradient descent since objective is convex ($\mathbf{A}^T\mathbf{A} \succeq 0$)

$$\begin{aligned} \beta^{t+1} &= \beta^t - \frac{\alpha}{2}\left.\frac{\partial J(\beta)}{\partial \beta}\right|_t \\ &= \beta^t - \alpha\, \mathbf{A}^T(\mathbf{A}\beta^t - Y) \end{aligned}$$

Ranke of A^TA

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

p x p p x1 p x1

When is $(\mathbf{A}^T\mathbf{A})$ invertible ?

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

Rank$(\mathbf{A}^T\mathbf{A})$ = number of non-zero eigenvalues of $(\mathbf{A}^T\mathbf{A})$= number of non-zero singular values of **A** <= min(n,p) since **A** is n x p

So, rank$(\mathbf{A}^T\mathbf{A})$, r <= min(n,p) not invertible if r < p (e.g. n < p i.e. high-dimensional setting)

Under-determined; no unique solution

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

p x p p x1 p x1

When is $(\mathbf{A}^T\mathbf{A})$ invertible ?

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

If $\mathbf{A} = \underset{S\text{ - r x r}}{\mathbf{U}}\mathbf{S}\mathbf{V}^\top$, then normal equations $(\underset{r\,x\,p}{\mathbf{S}\mathbf{V}^\top})\underset{p\,x\,1}{\hat{\beta}} = (\underset{r\,x\,1}{\mathbf{U}^\top\mathbf{Y}})$

r equations in p unknowns. Under-determined if r < p, hence no unique solution.

Regularized Least Squares; constrain the solution further

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

r equations , p unknowns – underdetermined system of linear equations
many feasible solutions

Need to constrain solution further

e.g. bias solution to "small" values of β (small changes in input don't translate to large changes in output)

$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

Ridge Regression
(l2 penalty)

$$= \arg\min_{\beta} \; (\mathbf{A}\beta - \mathbf{Y})^T(\mathbf{A}\beta - \mathbf{Y}) + \lambda\|\beta\|_2^2 \qquad \lambda \geq 0$$

$$\hat{\beta}_{\text{MAP}} = (\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^\top\mathbf{Y}$$

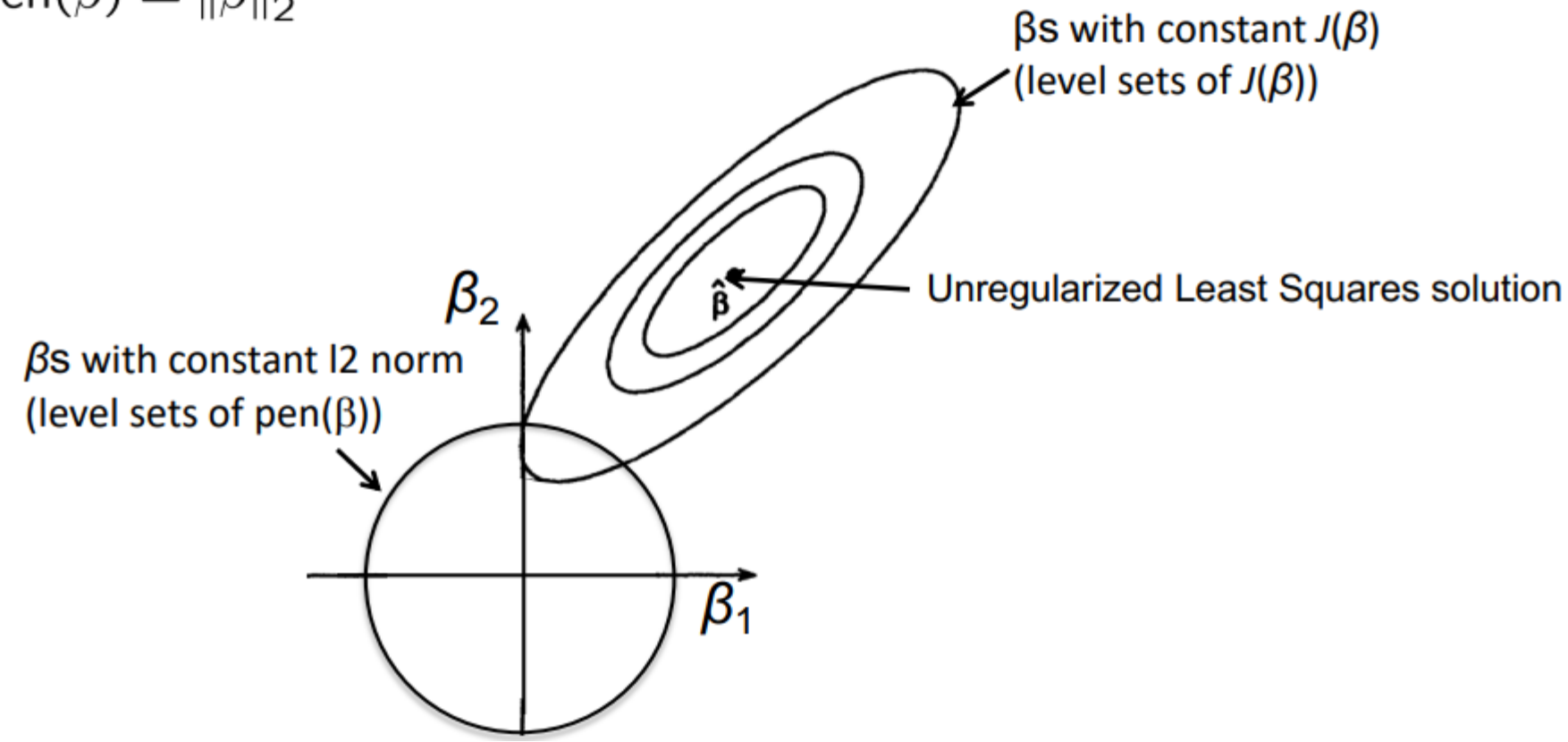
Is $(\mathbf{A}^\top\mathbf{A} + \lambda\mathbf{I})$ invertible ?

Understanding regularized Least Squares

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

Ridge Regression:

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



Lasso

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

r equations , p unknowns – underdetermined system of linear equations
many feasible solutions

Need to constrain solution further

e.g. bias solution to "small" values of β (small changes in input don't translate to large changes in output)

$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

Ridge Regression
(l2 penalty)

$$\hat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_1 \qquad \lambda \geq 0$$

Lasso
(l1 penalty)

Many β can be zero – many inputs are irrelevant to prediction in high-dimensional settings (typically intercept term not penalized)

No closed form solution, but can optimize using sub-gradient descent (packages available)

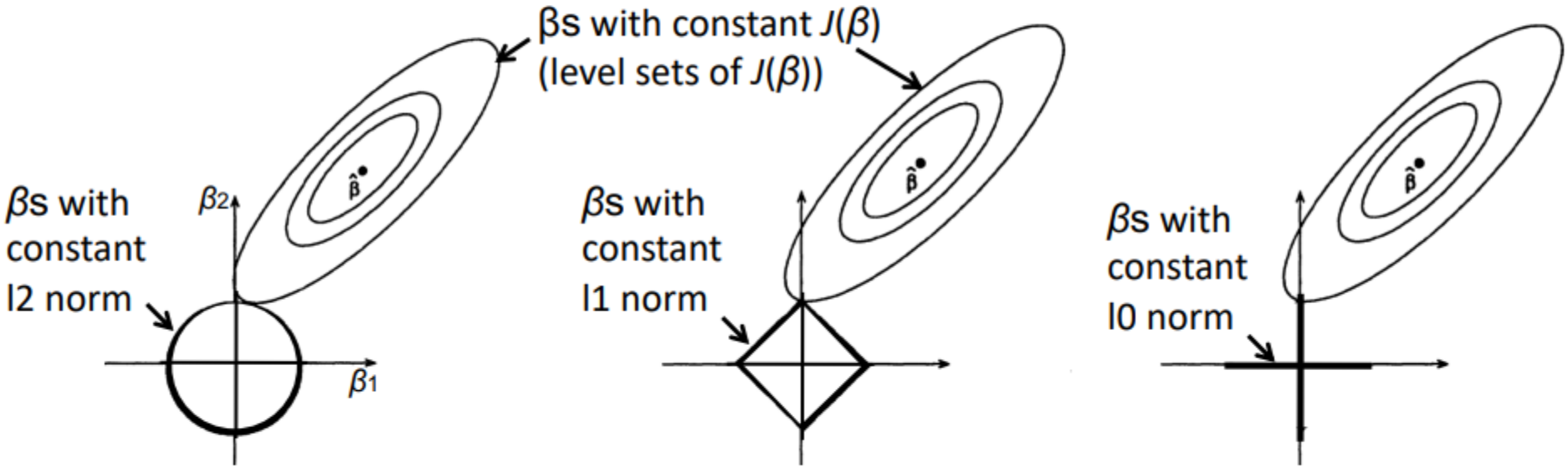
Ridge Regression vs Lasso

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

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

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

Ideally l0 penalty, but optimization becomes non-convex



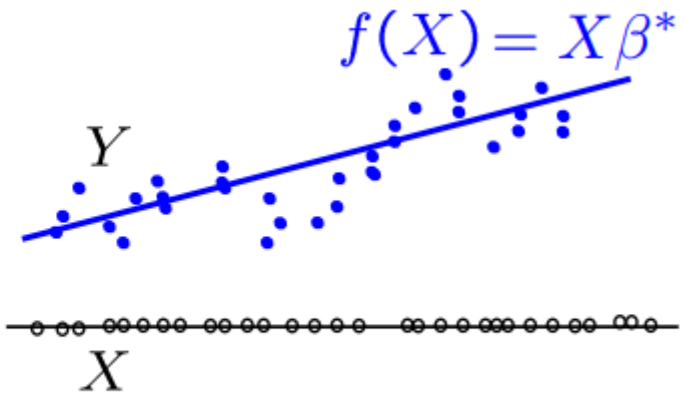
Lasso (l1 penalty) results in sparse solutions – vector with more zero coordinates
Good for high-dimensional problems – don't have to store all coordinates, interpretable solution!

Regularized Least Squares – connection to MLE and MAP (Model-based approaches)

Least Squares and MLE

Intuition: Signal plus (zero-mean) Noise model

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



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

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

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

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

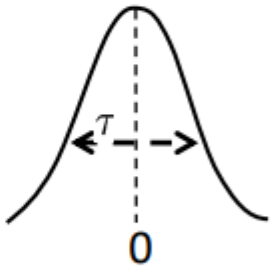
Regularized Least Squares and MAP

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

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

I) Gaussian Prior

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



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n}(Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

$\lambda \downarrow$ constant(σ^2,τ^2)

Ridge Regression

$$\widehat{\beta}_{\text{MAP}} = (\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^\top\boldsymbol{Y}$$

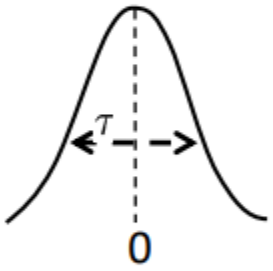
Regularized Least Squares and MAP

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

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

I) Gaussian Prior

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



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

λ : constant(σ^2, τ^2)

Ridge Regression

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

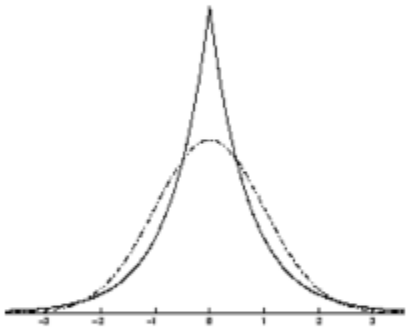
Regularized Least Squares and MAP

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

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

II) Laplace Prior

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



$$\widehat{\beta}_{\text{MAP}} = \arg\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda \|\beta\|_1$$

$\lambda \|\beta\|_1 \downarrow$ constant(σ^2, t)

Lasso

Prior belief that β is Laplace with zero-mean biases solution to "sparse" β

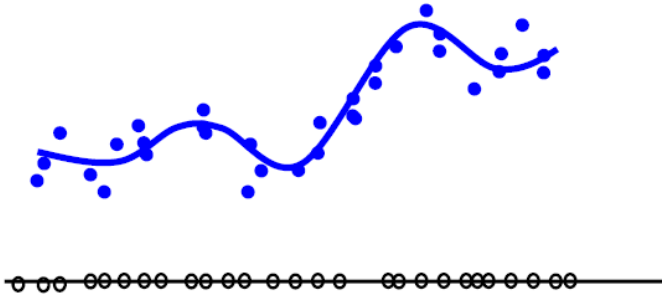
Nonlinear Regression
Bias Variance trade-off

dehaqani@ut.ac.ir

Slides are mainly adopted form cmu Aarti course and Tibshirani

Beyond Linear Regression

Polynomial regression
Regression with nonlinear features



Kernelized Ridge Regression

Local Kernel Regression

Polynomial Regression

degree m

Univariate (1-dim) case:

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

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

$$\widehat{\beta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{Y} \text{ or } (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y} \qquad \widehat{f}_n(X) = \mathbf{X}\widehat{\beta}$$

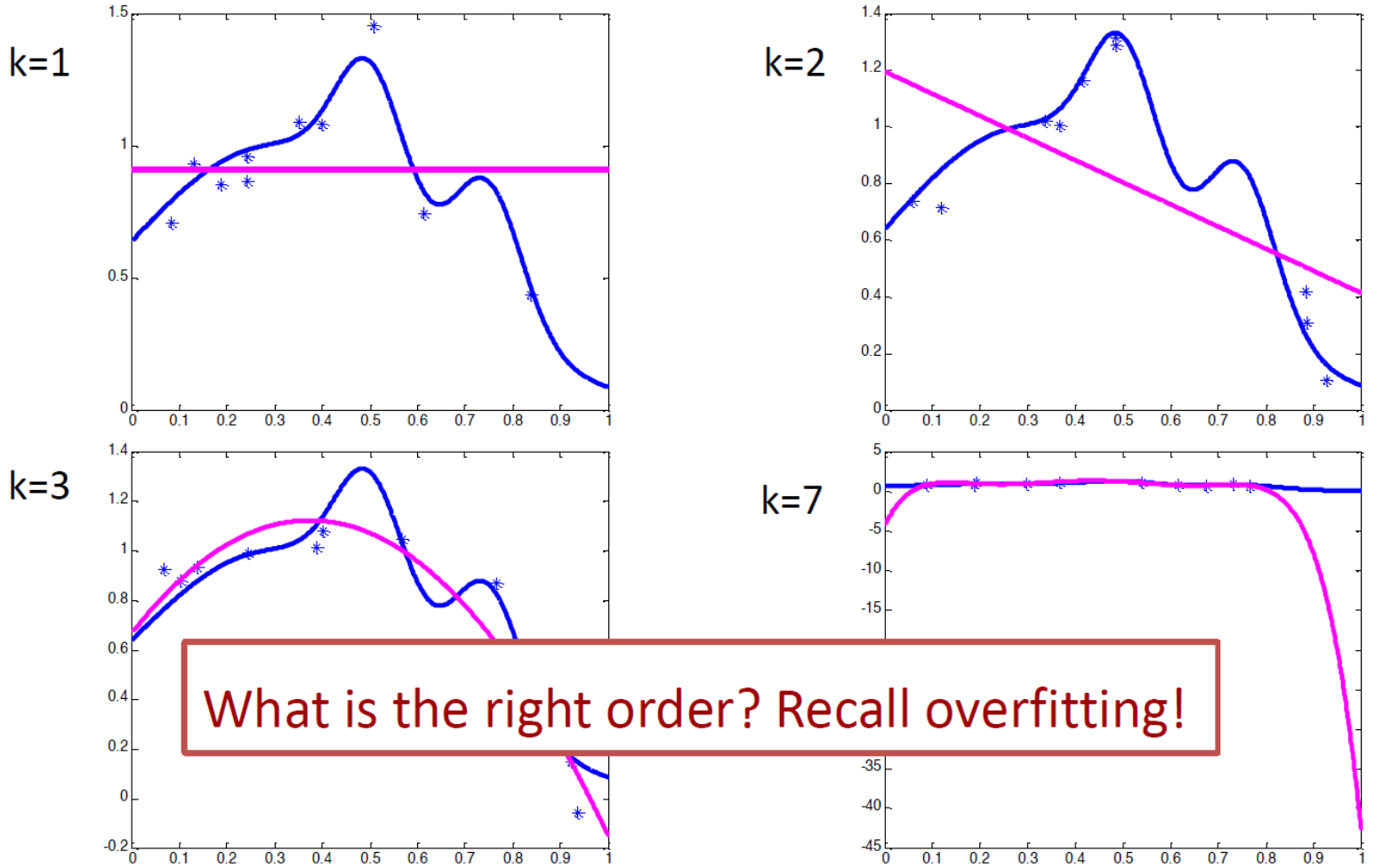
where $\mathbf{A} = \begin{bmatrix} 1 & X_1 & X_1^2 & \ldots & X_1^m \\ \vdots & & & \ddots & \vdots \\ 1 & X_n & X_n^2 & \ldots & X_n^m \end{bmatrix}$

Multivariate (p-dim) case:

$$\begin{aligned} f(X) \;=\; & \beta_0 + \beta_1 X^{(1)} + \beta_2 X^{(2)} + \cdots + \beta_p X^{(p)} \\ & + \sum_{i=1}^{p}\sum_{j=1}^{p} \beta_{ij} X^{(i)} X^{(j)} + \sum_{i=1}^{p}\sum_{j=1}^{p}\sum_{k=1}^{p} X^{(i)} X^{(j)} X^{(k)} \\ & + \ldots \text{terms up to degree m} \end{aligned}$$

Polynomial Regression

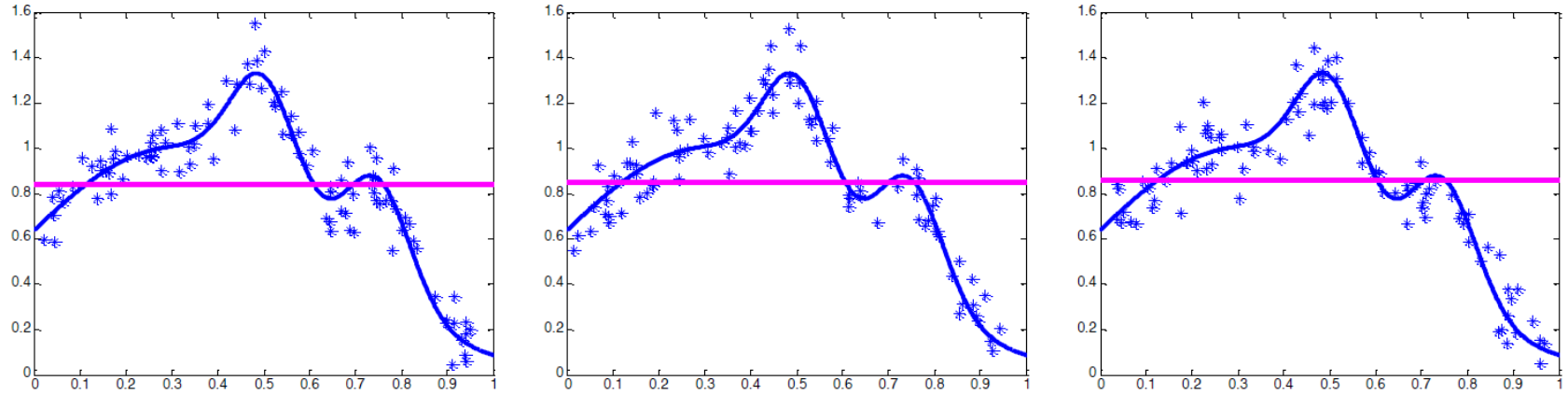
Polynomial of order k, equivalently of degree up to k-1



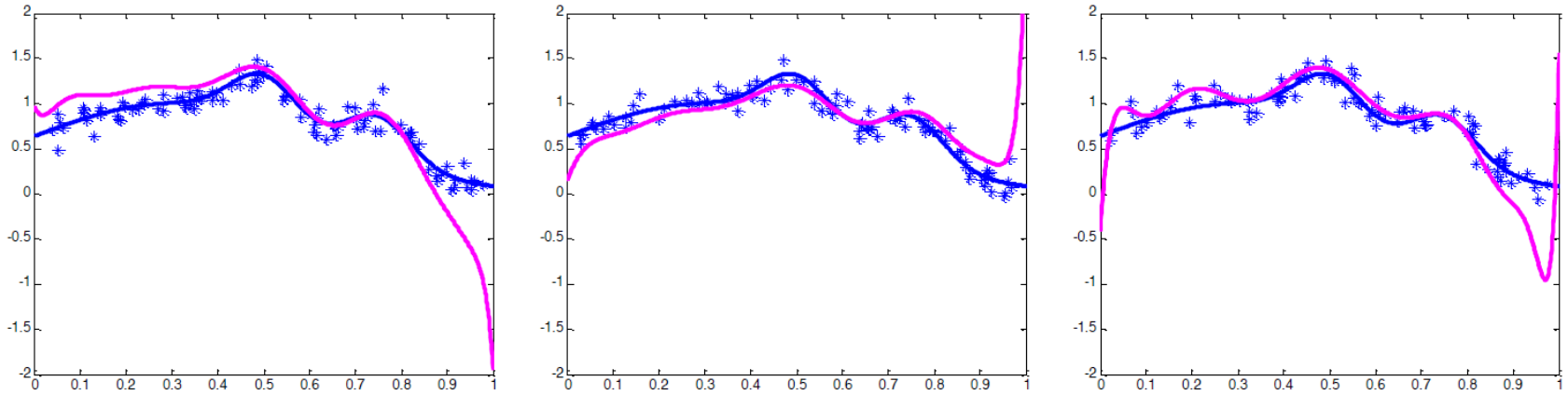
Bias – Variance Tradeoff

3 Independent training datasets

Large bias, Small variance – poor approximation but robust/stable



Small bias, Large variance – good approximation but unstable



Bias – Variance Decomposition

It can be shown that

$$E[(f(X) - f^*(X))^2] = \text{Bias}^2 + \text{Variance}$$

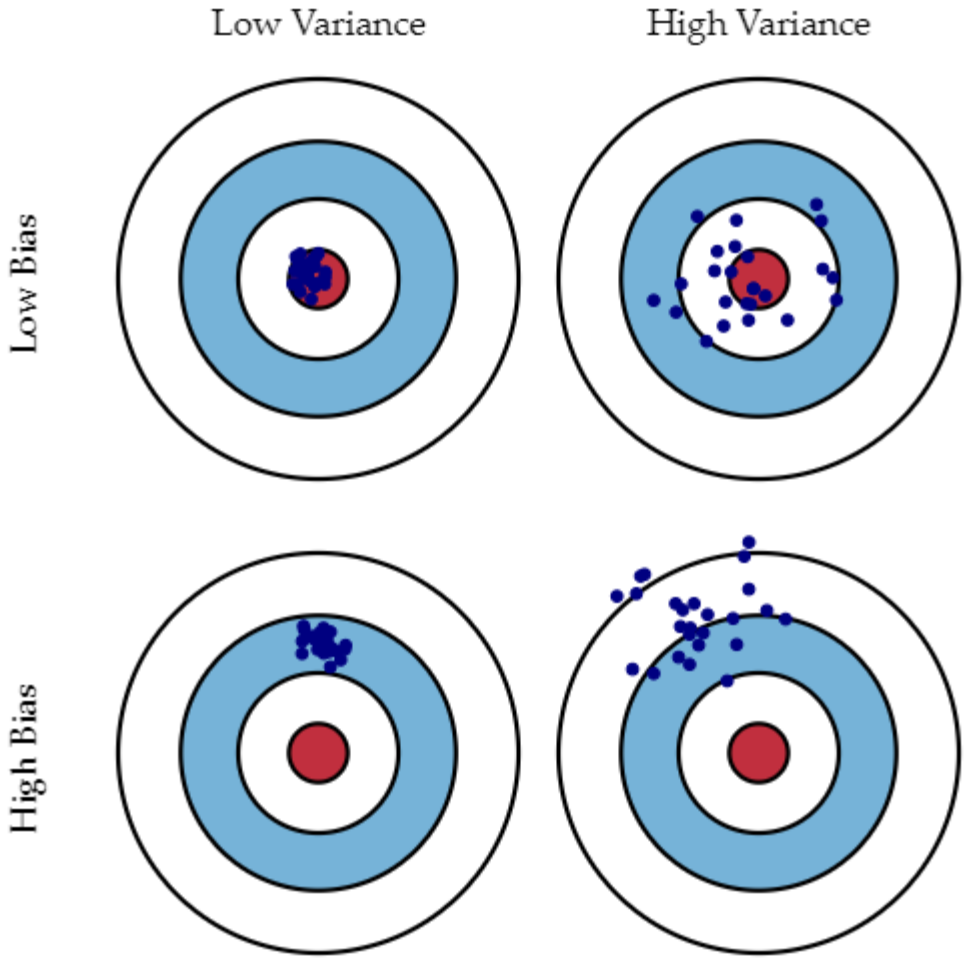
$$\text{Bias} = E[f(X)] - f^*(X)$$ How far is the model from best model on average

$$\text{Variance} = E[(f(X) - E[f(X)])^2]$$ How variable is the model

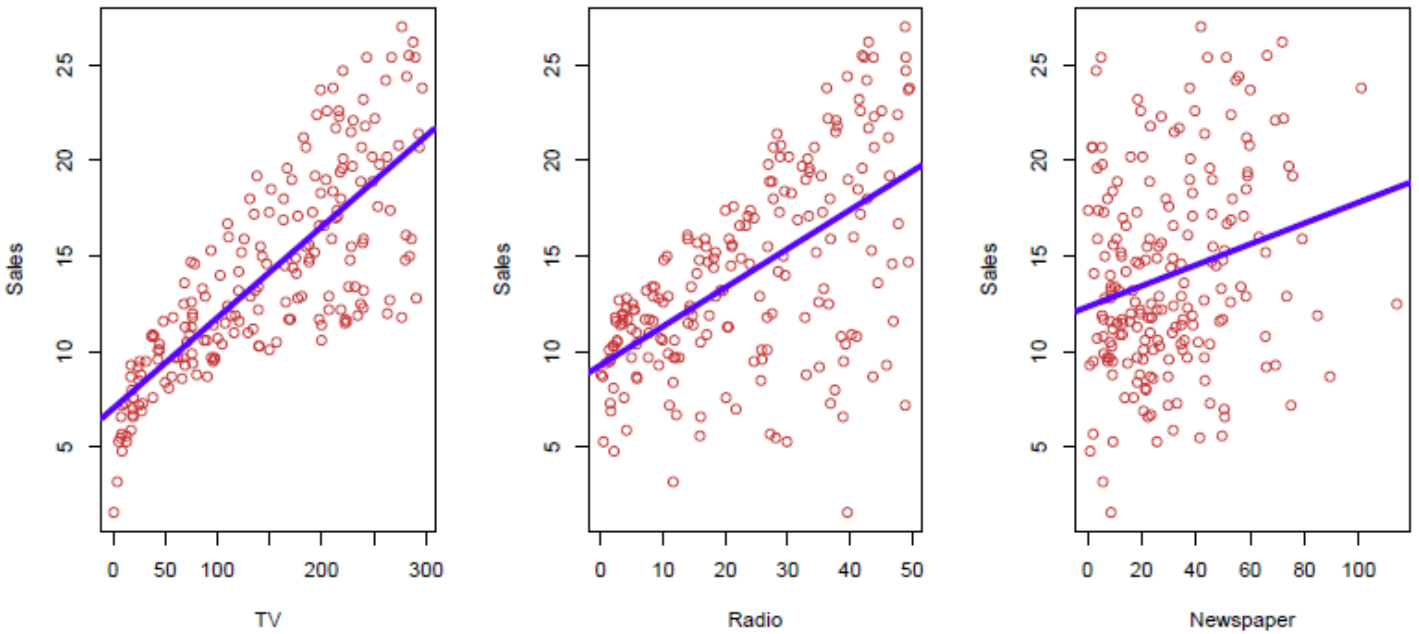
Underfitting vs. overfitting

- The **bias** error: (**underfitting**)
  - From **erroneous assumptions** in the learning algorithm.
  - High bias can cause an algorithm to **miss the relevant relations** between features and target outputs.
- The **variance** error: (**overfitting**).
  - From **sensitivity** to **small fluctuations** in the training set.
  - High variance can cause an algorithm to **model the random noise** in the training data, rather than the intended outputs

Graphical illustration of bias and variance



Mathematical Definition; Statistical learning



Shown are Sales vs TV, Radio and Newspaper, with a blue linear-regression line fit separately to each. Can we predict Sales using these three? Perhaps we can do better using a model

$$\text{Sales} \approx f(\text{TV}, \text{Radio}, \text{Newspaper})$$

Notation

Here Sales is a response or target that we wish to predict. We generically refer to the response as Y .TV is a feature, or input, or predictor; we name it X_1. Likewise name Radio as X_2, and so on. We can refer to the input vector collectively as

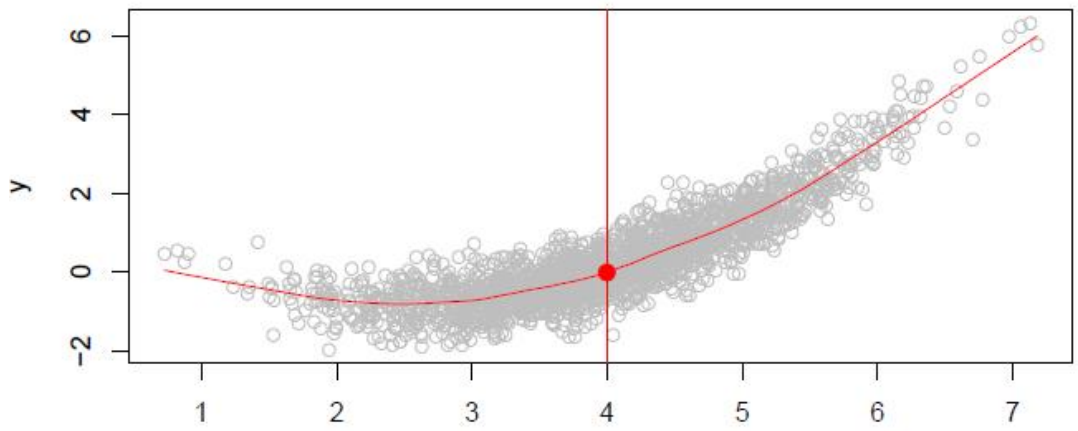
$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix}$$

Now we write our model as

$$Y = f(X) + \varepsilon$$

where ε captures measurement errors and other discrepancies.

What is f(X) good for?



Is there an ideal $f(X)$? In particular, what is a good value for $f(X)$ at any selected value of X, say $X = 4$? There can be many Y values at $X = 4$. A good value is

$$f(4) = E(Y|X = 4)$$

$E(Y|X = 4)$ means *expected value* (average) of Y given $X = 4$.

This ideal $f(x) = E(Y|X = x)$ is called the *regression function*.

The regression function f(x)

- Is also defined for vector X; e.g.
  $f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$

- Is the *ideal* or *optimal* predictor of Y with regard to mean-squared prediction error: $f(x) = E(Y|X = x)$ is the function that minimizes $E[(Y - g(X))^2|X = x]$ over all functions g at all points $X = x$.

- $\epsilon = Y - f(x)$ is the *irreducible* error — i.e. even if we knew $f(x)$, we would still make errors in prediction, since at each $X = x$ there is typically a distribution of possible Y values.

- For any estimate $\hat{f}(x)$ of $f(x)$, we have

$$E[(Y - \hat{f}(X))^2|X = x] = \underbrace{[f(x) - \hat{f}(x)]^2}_{Reducible} + \underbrace{\mathrm{Var}(\epsilon)}_{Irreducible}$$

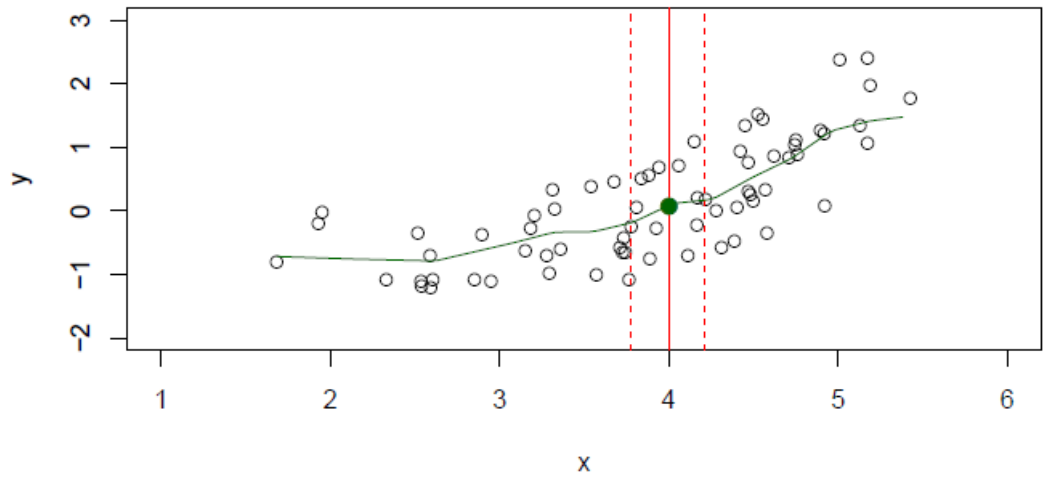
How to estimate f

- Typically we have few if any data points with $X = 4$ exactly.
- So we cannot compute $E(Y|X = x)$!

- Relax the definition and let

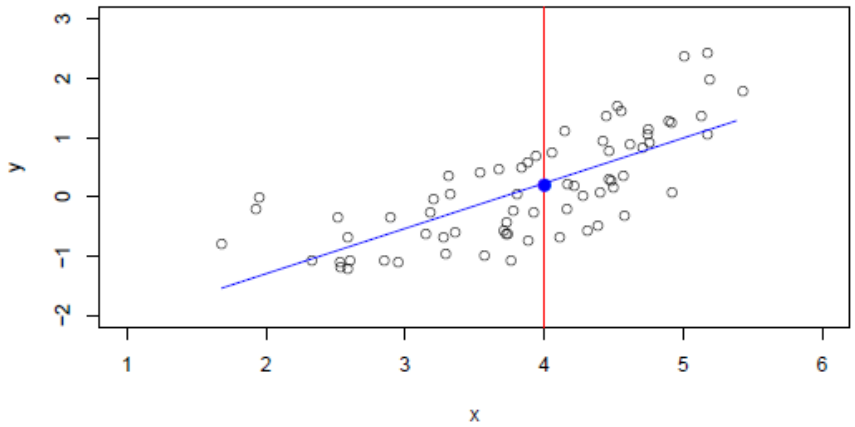
$$\hat{f}(x) = \text{Ave}(Y|X \in \mathcal{N}(x))$$

where $\mathcal{N}(x)$ is some *neighborhood* of x.

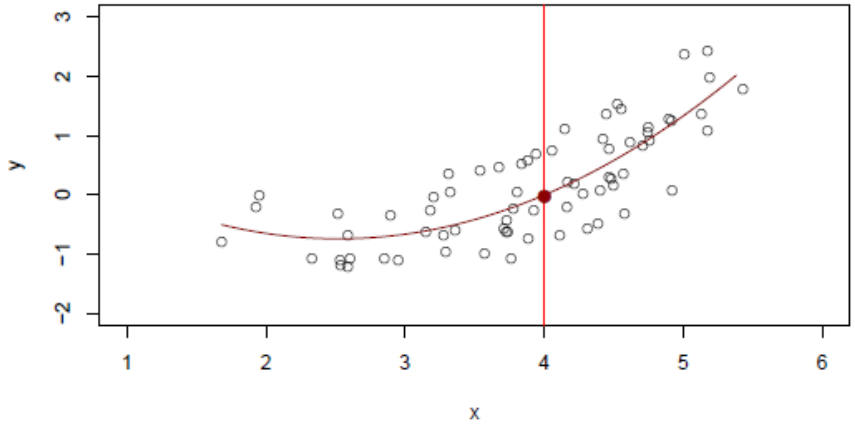


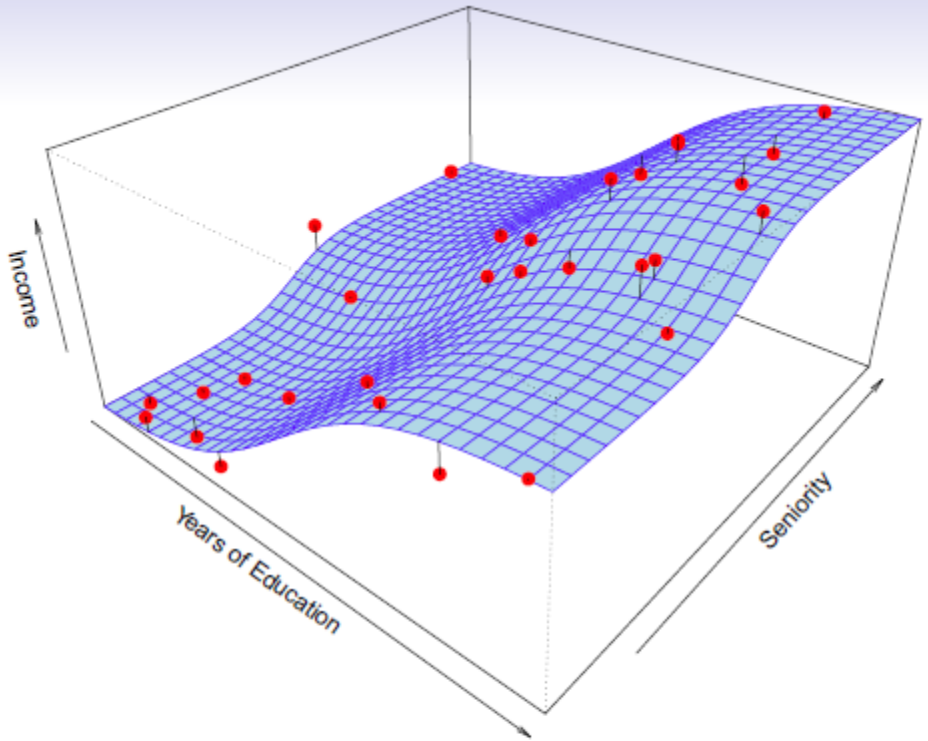
Parametric and structured models

A linear model $\hat{f}_L(X) = \hat{\beta}_0 + \hat{\beta}_1 X$ gives a reasonable fit here



A quadratic model $\hat{f}_Q(X) = \hat{\beta}_0 + \hat{\beta}_1 X + \hat{\beta}_2 X^2$ fits slightly better.

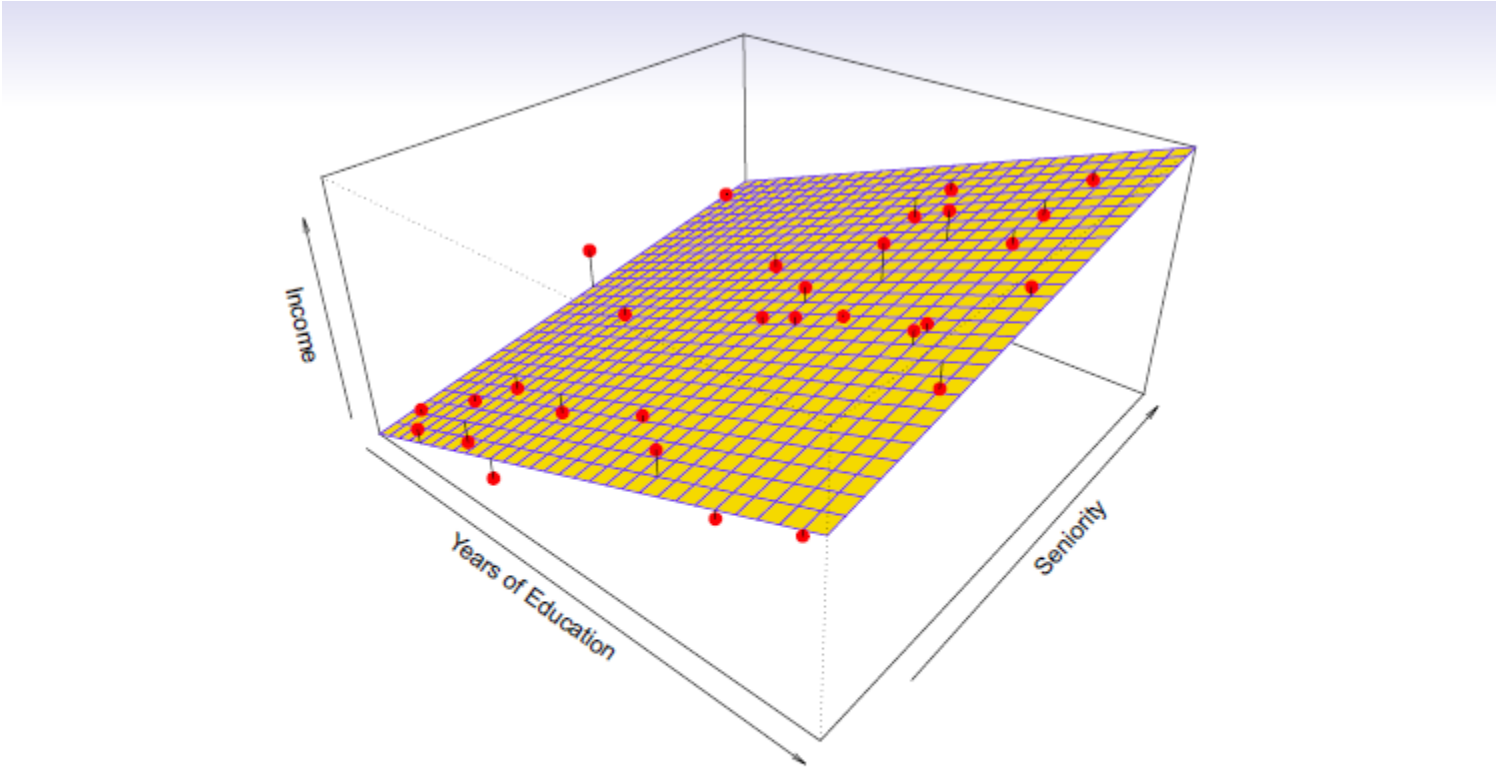




Simulated example. Red points are simulated values for income from the model

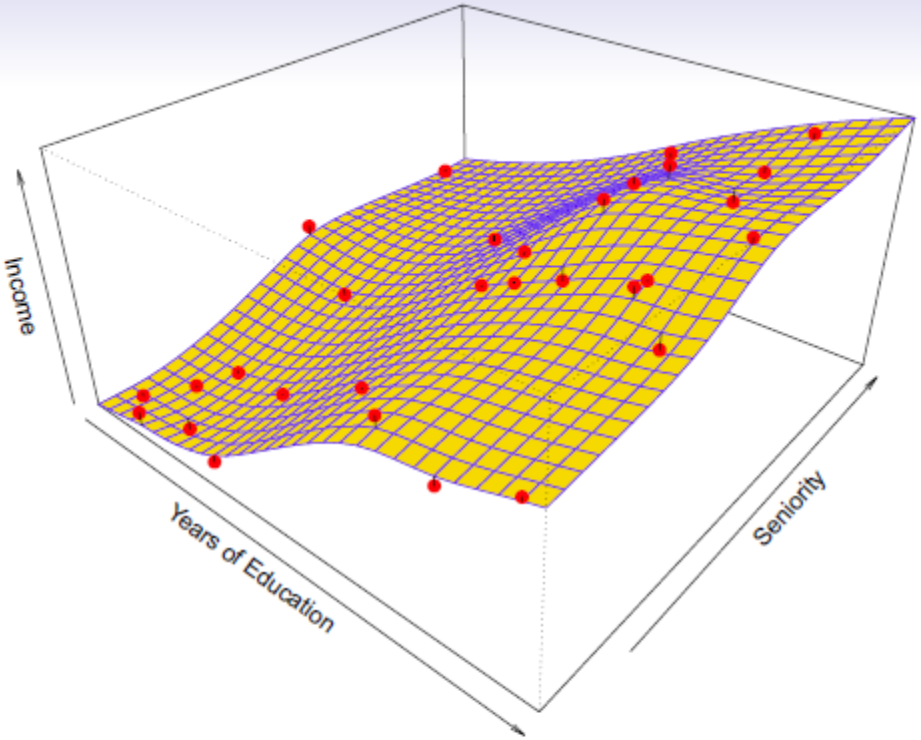
$$\texttt{income} = f(\texttt{education}, \texttt{seniority}) + \epsilon$$

f is the blue surface.

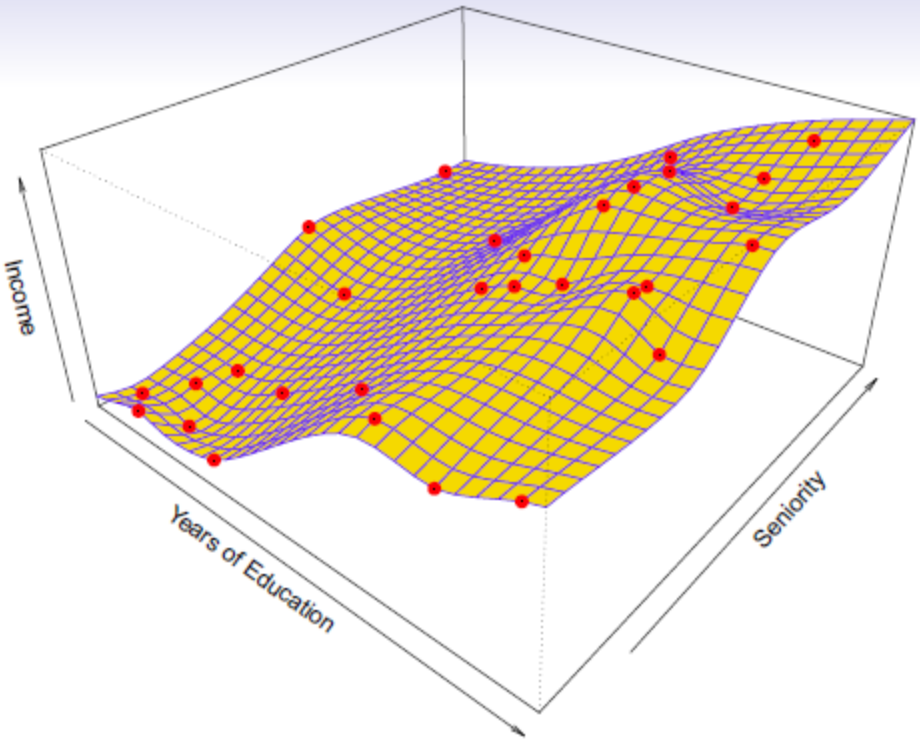


Linear regression model fit to the simulated data.

$$\hat{f}_L(\texttt{education}, \texttt{seniority}) = \hat{\beta}_0 + \hat{\beta}_1 \times \texttt{education} + \hat{\beta}_2 \times \texttt{seniority}$$



More flexible regression model $\hat{f}_S(\texttt{education}, \texttt{seniority})$ fit to the simulated data. Here we use a technique called a *thin-plate spline* to fit a flexible surface. We control the roughness of the fit



Even more flexible spline regression model $\hat{f}_S(\texttt{education}, \texttt{seniority})$ fit to the simulated data. Here the fitted model makes no errors on the training data! Also known as *overfitting*.

Some trade-offs

- Prediction accuracy versus interpretability.
  — Linear models are easy to interpret; thin-plate splines are not.
- Good fit versus over-fit or under-fit.
  — How do we know when the fit is just right?
- Parsimony versus black-box.
  — We often prefer a simpler model involving fewer variables over a black-box predictor involving them all.

Assessing Model Accuracy

Suppose we fit a model $\hat{f}(x)$ to some training data $\mathsf{Tr} = \{x_i, y_i\}_1^N$, and we wish to see how well it performs.

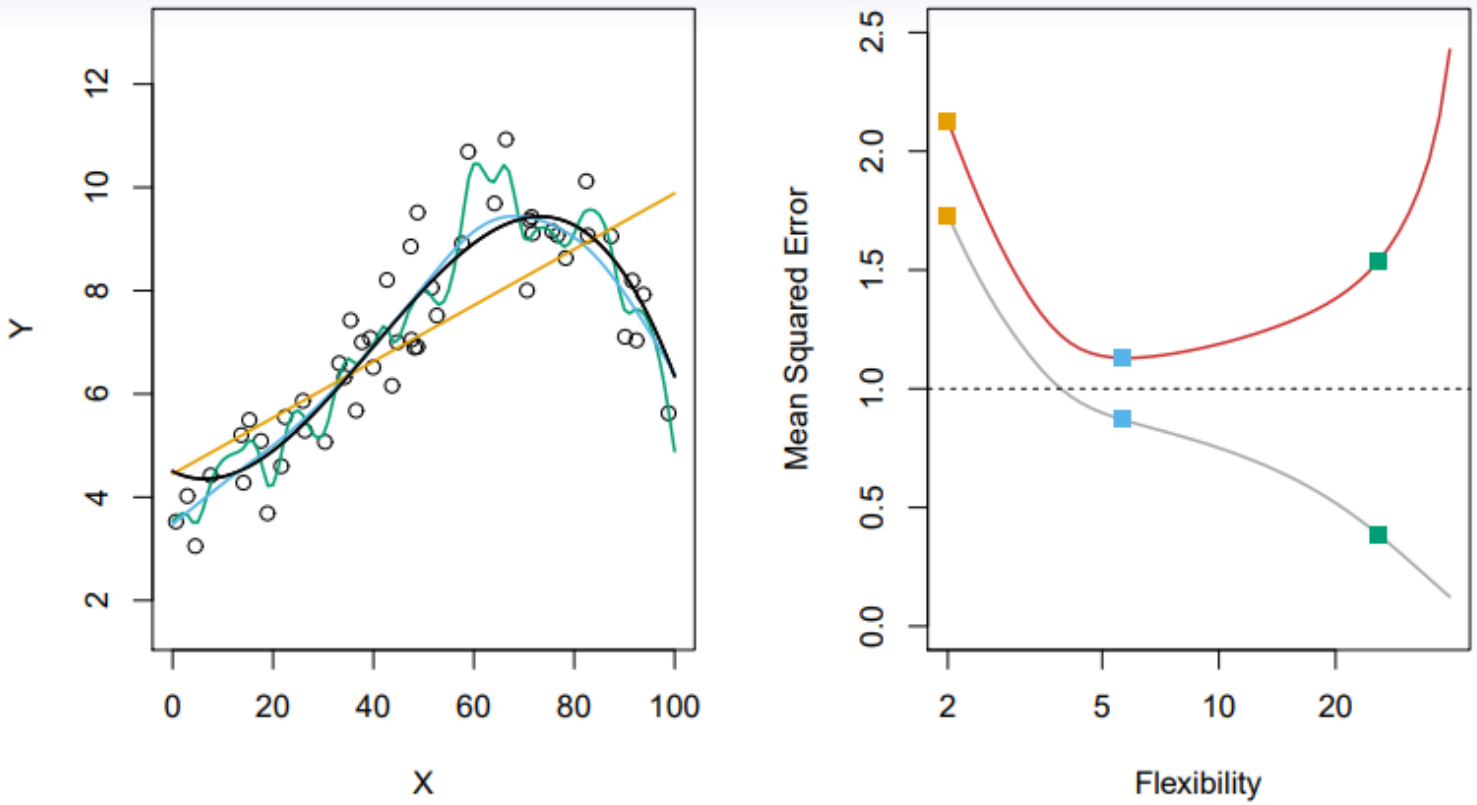
- We could compute the average squared prediction error over Tr:

$$\text{MSE}_{\mathsf{Tr}} = \text{Ave}_{i \in \mathsf{Tr}}[y_i - \hat{f}(x_i)]^2$$

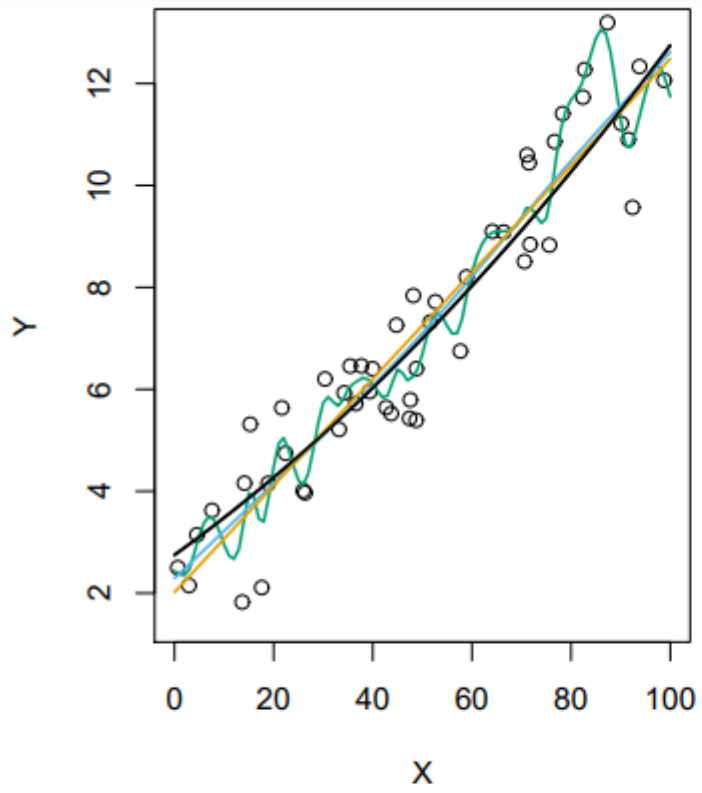
This may be biased toward more overfit models.

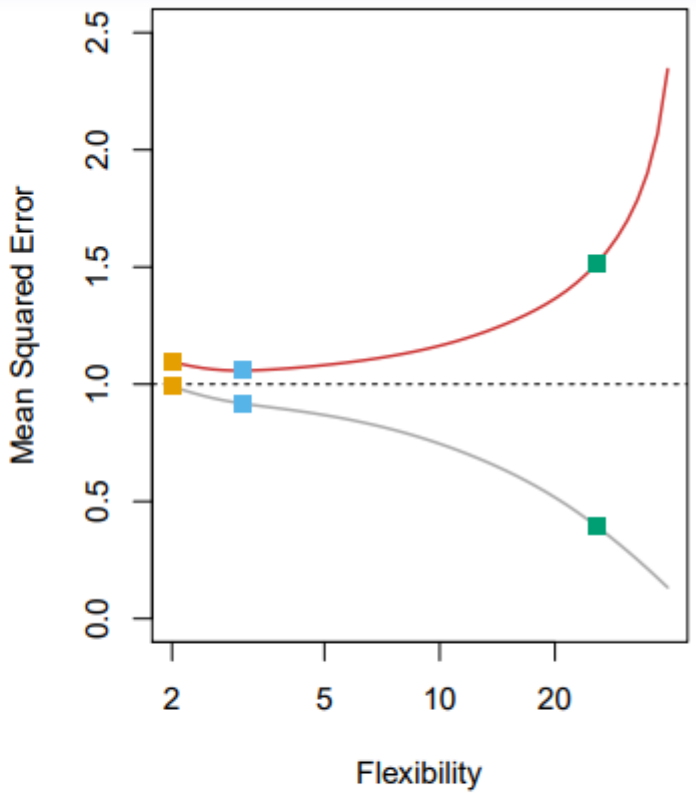
- Instead we should, if possible, compute it using fresh *test* data $\mathsf{Te} = \{x_i, y_i\}_1^M$:

$$\text{MSE}_{\mathsf{Te}} = \text{Ave}_{i \in \mathsf{Te}}[y_i - \hat{f}(x_i)]^2$$

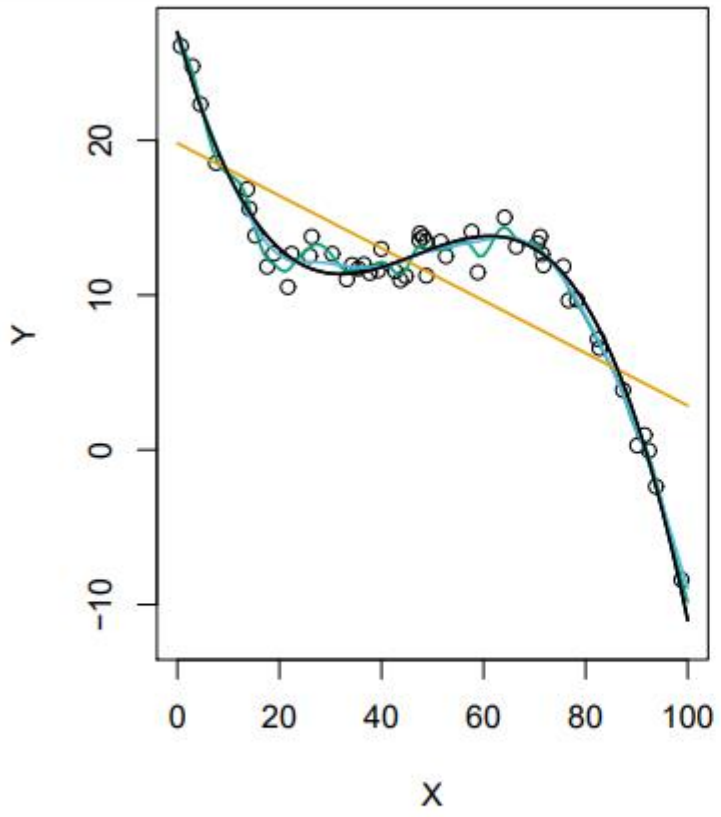


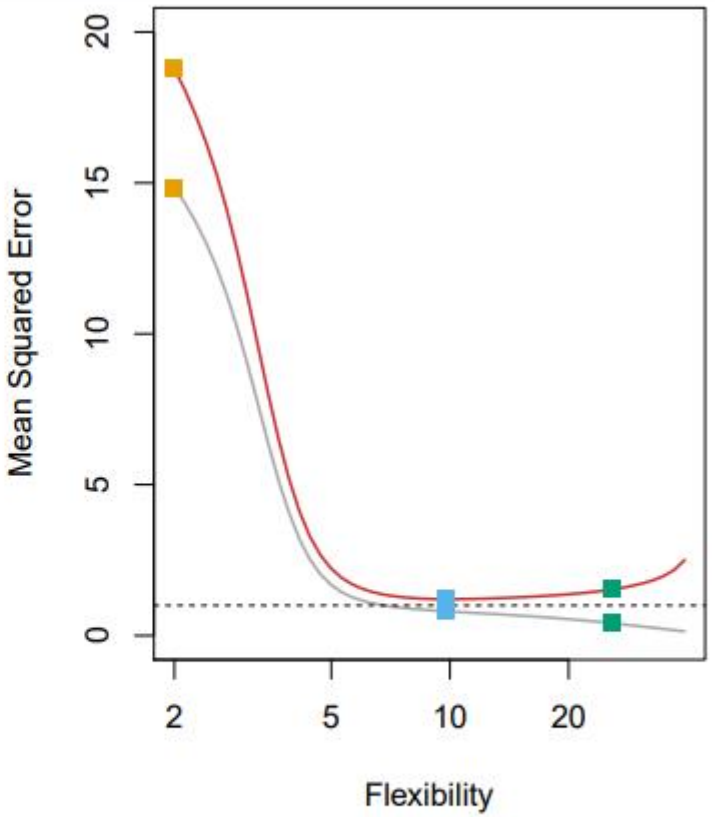
Black curve is truth. Red curve on right is MSE_{Te}, grey curve is MSE_{Tr}. Orange, blue and green curves/squares correspond to fits of different flexibility.





Here the truth is smoother, so the smoother fit and linear model do really well.





Here the truth is wiggly and the noise is low, so the more flexible fits do the best.

Bias-Variance Trade-off

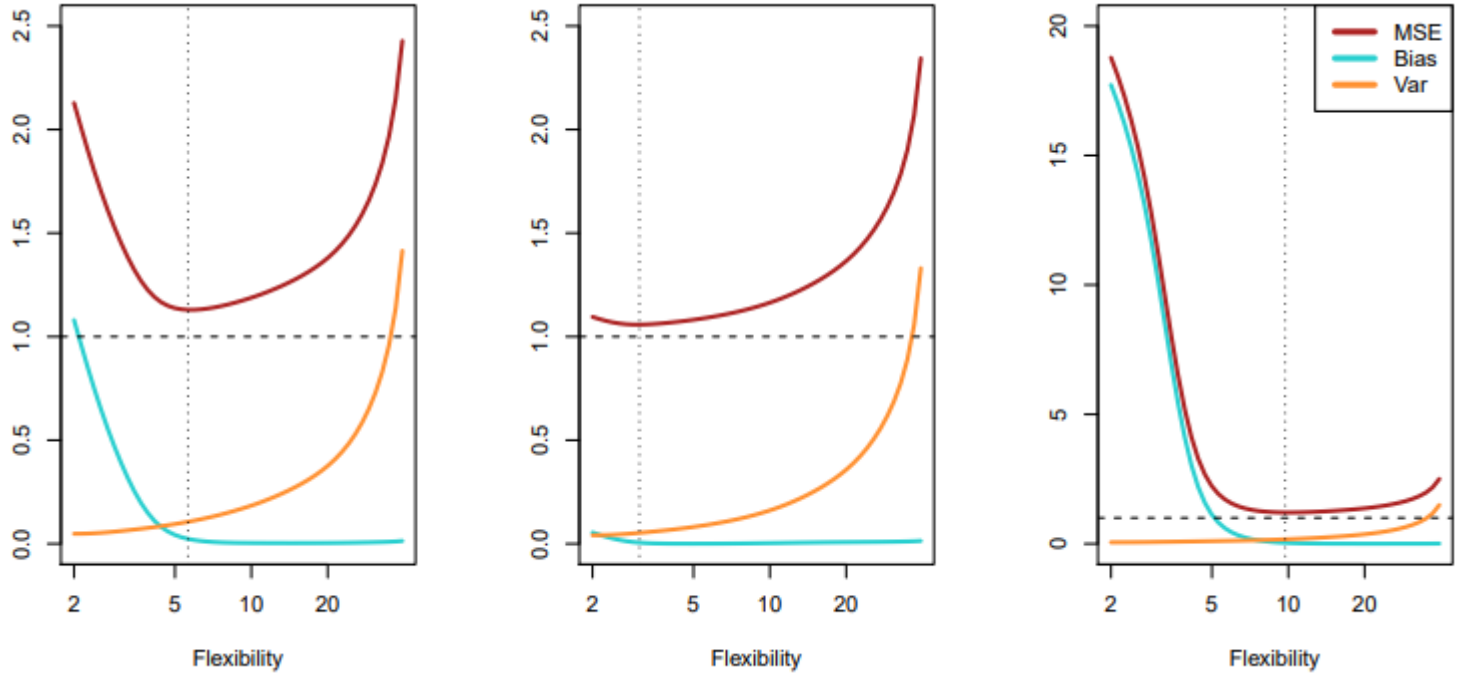
Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

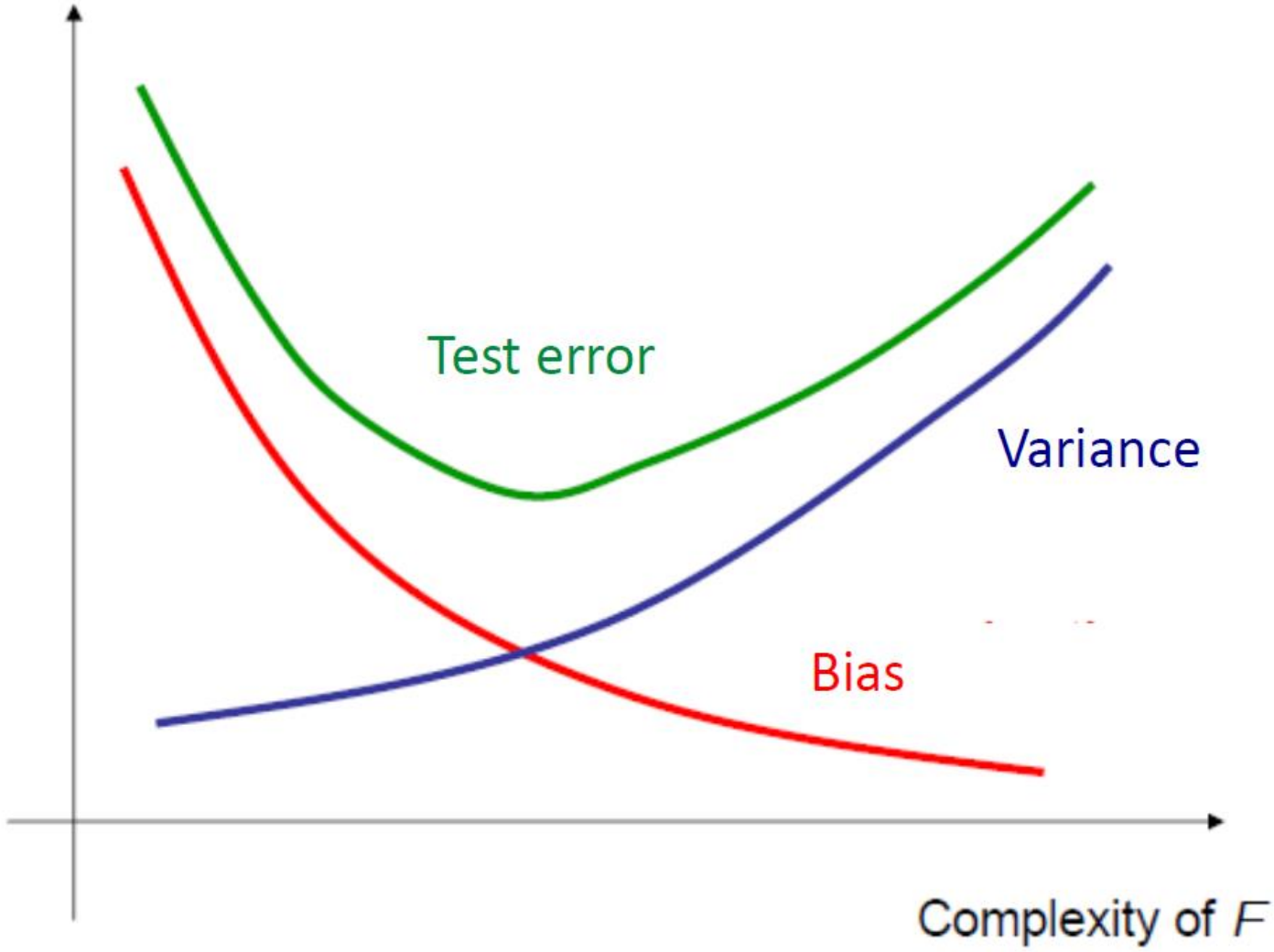
The expectation averages over the variability of y_0 as well as the variability in Tr. Note that $\text{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$.

Typically as the *flexibility* of $\hat{f}$ increases, its variance increases, and its bias decreases. So choosing the flexibility based on average test error amounts to a *bias-variance trade-off.*

Bias-variance trade-off for the three examples



Effect of Model Complexity

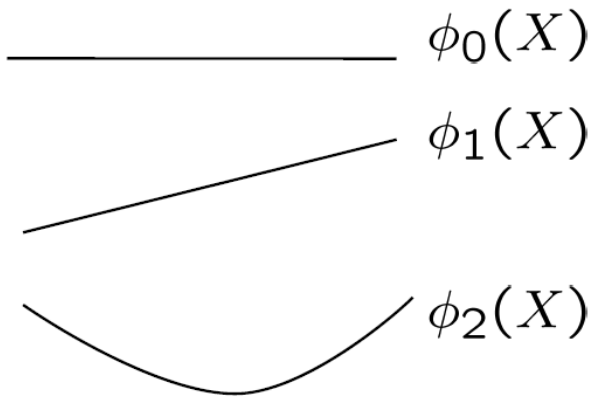


Regression with nonlinear features

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

Weight of each feature

Nonlinear features



In general, use any nonlinear features

e.g. e^X, log X, 1/X, sin(X), …

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

or

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

$$\mathbf{A} = \begin{bmatrix} \phi_0(X_1) & \phi_1(X_1) & \dots & \phi_m(X_1) \\ \vdots & & \ddots & \vdots \\ \phi_0(X_n) & \phi_1(X_n) & \dots & \phi_m(X_n) \end{bmatrix}$$

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

$$\mathbf{X} = [\phi_0(X) \ \phi_1(X) \ \dots \ \phi_m(X)]$$

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Primal problem:

$$\min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|_2^2$$
$$\text{s.t. } z_i = Y_i - X_i\beta$$

Lagrangian:

$$\sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i(z_i - Y_i + X_i\beta)$$

α_i – Lagrange parameter, one per training point

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \min_{\beta, z_i} \sum_{i=1}^{n} z_i^2 + \lambda\|\beta\|^2 + \sum_{i=1}^{n} \alpha_i(z_i - Y_i + X_i\beta)$$

$\alpha = \{\alpha_i\}$ for i = 1,…, n

Taking derivatives of Lagrangian wrt β and z_i we get:

$$\beta = -\frac{1}{2\lambda}\mathbf{A}^\top\alpha \qquad z_i = -\frac{\alpha_i}{2}$$

Dual problem: $$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y}$$

n-dimensional optimization problem

Ridge regression (dual)

$$\min_{\beta} \sum_{i=1}^{n} (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2 \qquad \widehat{\beta} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y}$$

Dual problem:

$$\max_{\alpha} \; -\frac{\alpha^\top\alpha}{4} - \frac{1}{4\lambda}\alpha^\top\mathbf{A}\mathbf{A}^\top\alpha - \alpha^\top\mathbf{Y} \qquad \Rightarrow \widehat{\alpha} = -\left(\frac{\mathbf{A}\mathbf{A}^\top}{\lambda} + \mathbf{I}\right)^{-1} 2\,\mathbf{Y}$$

can get back $\hat{\beta} = -\frac{1}{2\lambda}\mathbf{A}^\top\hat{\alpha} = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top + \lambda\mathbf{I})^{-1}\mathbf{Y}$

Weighted average of training points

Weight of each training point (but typically not sparse)

Kernelized ridge regression

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

Using dual, can re-write solution as:

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

How does this help?

- Only need to invert n x n matrix (instead of p x p or m x m)
- More importantly, kernel trick!

$$\widehat{f}_n(X) = \mathbf{K}_X(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{Y} \quad \text{where} \quad \mathbf{K}_X(j) = \boldsymbol{\phi}(X) \cdot \boldsymbol{\phi}(X_i)$$
$$\mathbf{K}(i,j) = \boldsymbol{\phi}(X_i) \cdot \boldsymbol{\phi}(X_j)$$

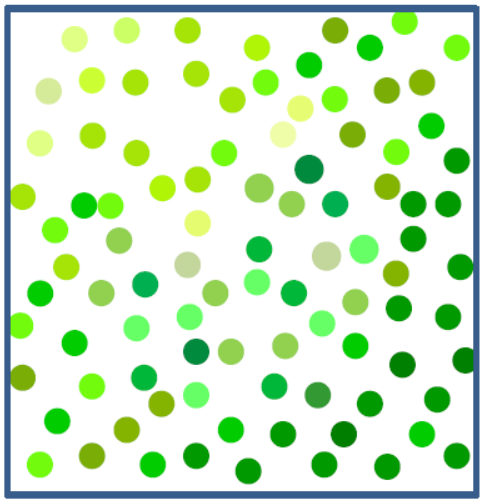
Work with kernels, never need to write out the high-dim vectors

We will come back to it later !!

Local Kernel Regression

- What is the temperature

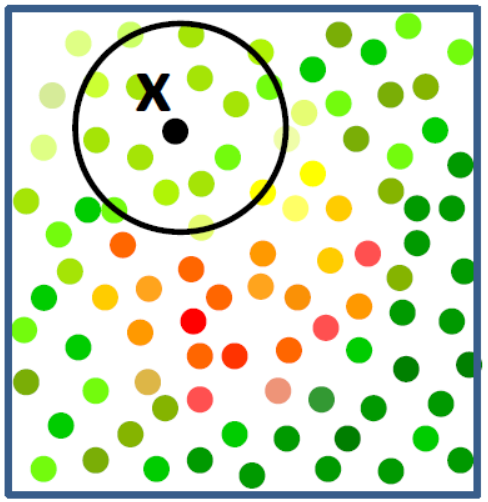
  in the room?



$$\widehat{T} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Average

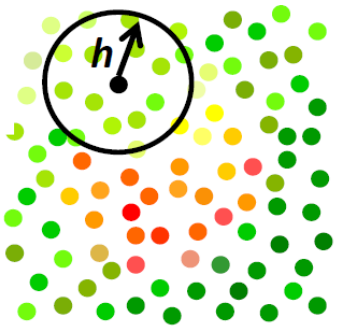
at location x?



$$\widehat{T}(x) = \frac{\sum_{i=1}^{n} Y_i \mathbf{1}_{||X_i - x|| \le h}}{\sum_{i=1}^{n} \mathbf{1}_{||X_i - x|| \le h}}$$

"Local" Average

Local Kernel Regression

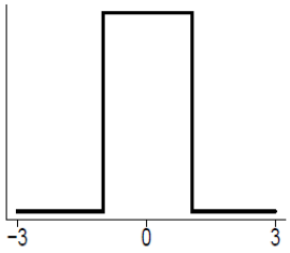


- Nonparametric estimator akin to kNN
- Nadaraya-Watson Kernel Estimator

$$\hat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \quad \text{Where} \quad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

- Weight each training point based on distance to test point
- Boxcar kernel yields local average

$$\text{boxcar kernel}: \quad K(x) = \frac{1}{2} I(x),$$



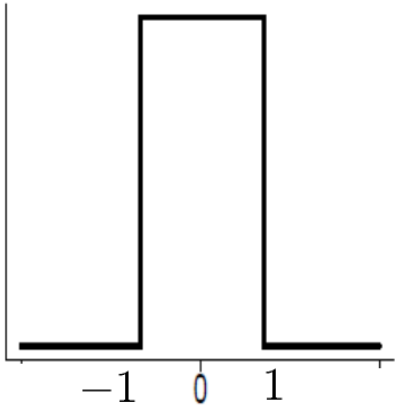
Kernels

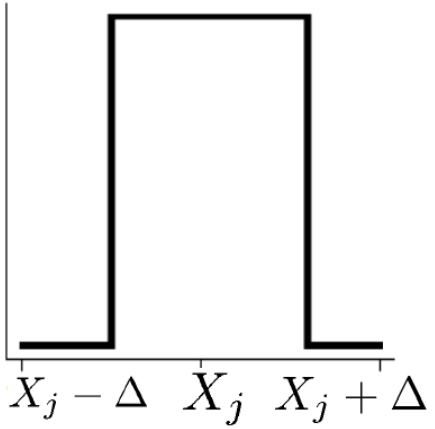
$$K(x) \geq 0,$$
$$\int K(x)dx = 1$$

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

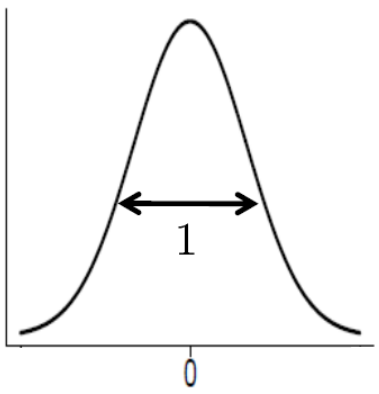
$$K(x) = \frac{1}{2}I(x),$$

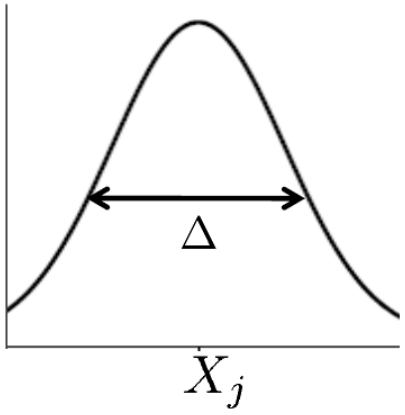




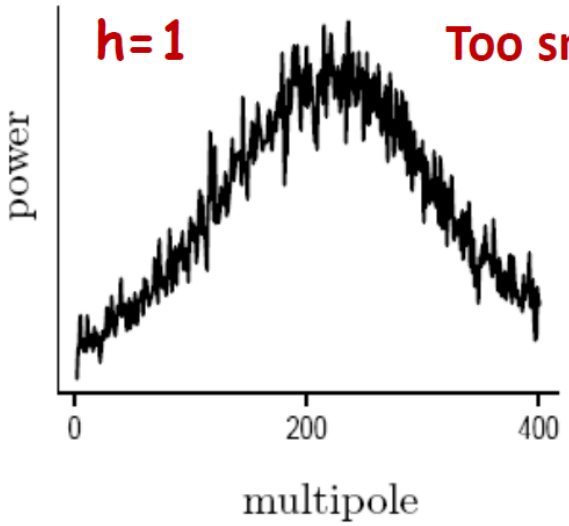
Gaussian kernel :

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





Choice of kernel bandwidth h



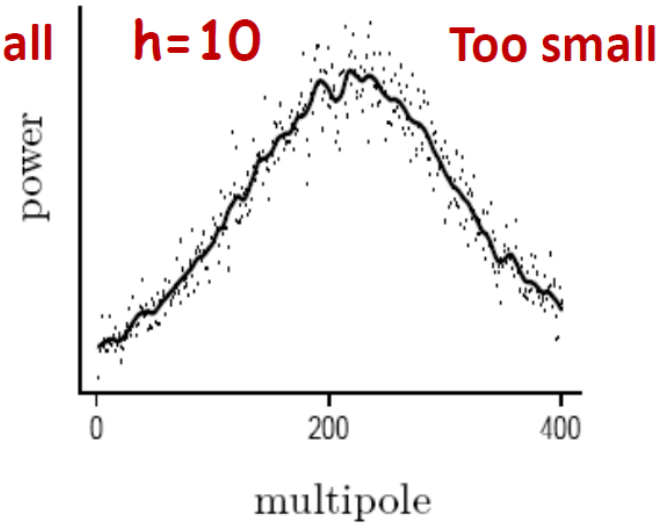
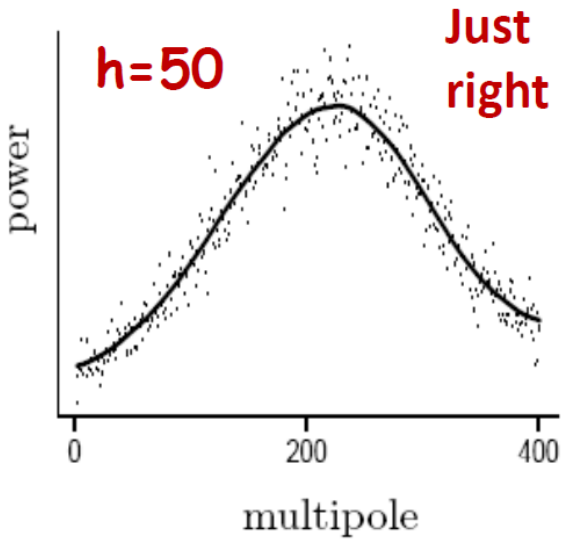
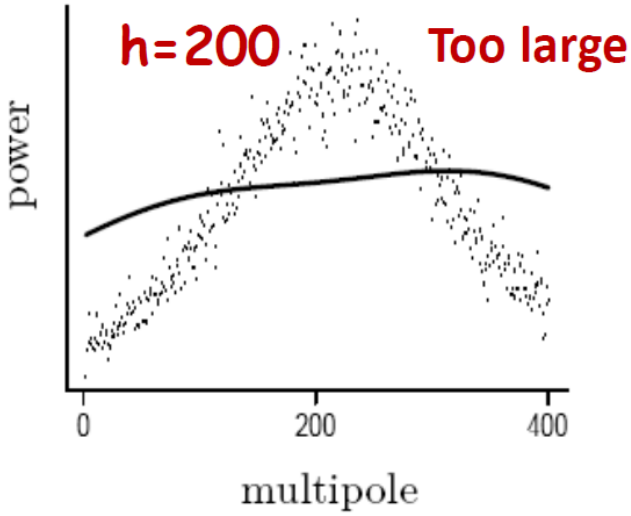


Image Source:
Larry's book – All of Nonparametric Statistics





Choice of kernel is not that important

Kernel Regression as Weighted Least Squares

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

Weighted Least Squares

Kernel regression corresponds to locally constant estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta$ (a constant)

Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta$ (a constant)

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

β: constant

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

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

Notice that $\sum_{i=1}^{n} w_i = 1$

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

To minimize J, let's find its derivatives with respect to θ.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta \left((X\theta)^T X\theta - (X\theta)^T\vec{y} - \vec{y}^T(X\theta) + \vec{y}^T\vec{y}\right) \\
a^Tb = b^Ta \quad &= \frac{1}{2}\nabla_\theta \left(\theta^T(X^TX)\theta - \vec{y}^T(X\theta) - \vec{y}^T(X\theta)\right) \\
&= \frac{1}{2}\nabla_\theta \left(\theta^T(X^TX)\theta - 2(X^T\vec{y})^T\theta\right) \\
\nabla_x b^Tx = b \quad & \\
\nabla_x x^TAx = 2Ax \quad &= \frac{1}{2}\left(2X^TX\theta - 2X^T\vec{y}\right) \\
&= X^TX\theta - X^T\vec{y}
\end{aligned}$$

$$\theta = (X^T X)^{-1} X^T y$$

$(X^T X)^{-1}$ is inverse of matrix $X^T X$.

Octave: `pinv(X'*X)*X'*y`

Normal equation

$$\theta = (X^T X)^{-1} X^T y$$

- What if $X^T X$ is non-invertible? (singular/ degenerate)
- Octave: `pinv(X'*X)*X'*y`

Local Linear/Polynomial Regression

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

Weighted Least Squares

Local Polynomial regression corresponds to locally polynomial estimator obtained from (locally) weighted least squares

i.e. set $f(X_i) = \beta_0 + \beta_1(X_i - X) + \frac{\beta_2}{2!}(X_i - X)^2 + \cdots + \frac{\beta_p}{p!}(X_i - X)^p$

(local polynomial of degree p around X)