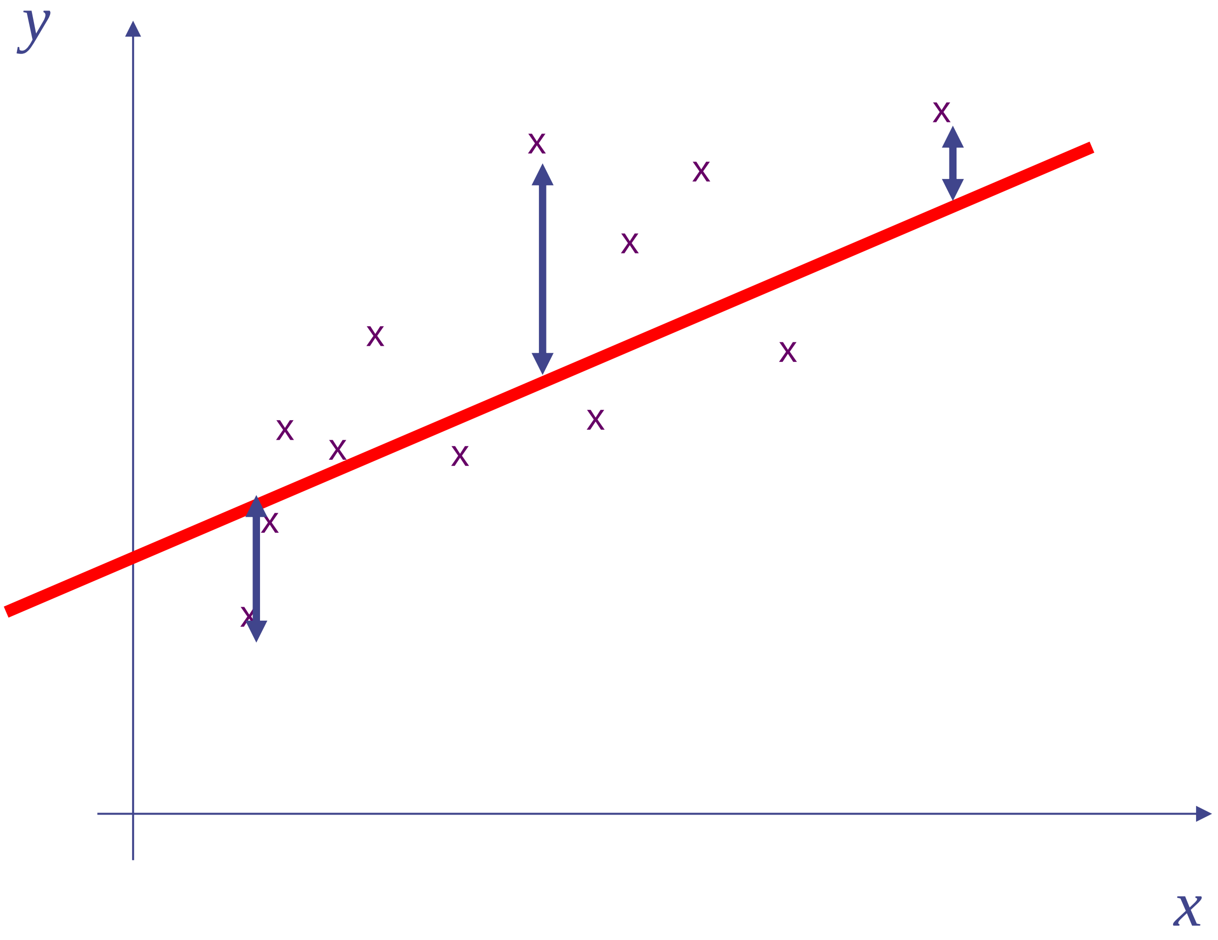
Machine Learning

4771

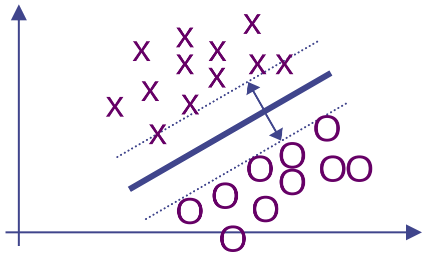
Instructor: Itsik Pe’er

Reminder: Gaussian Noise

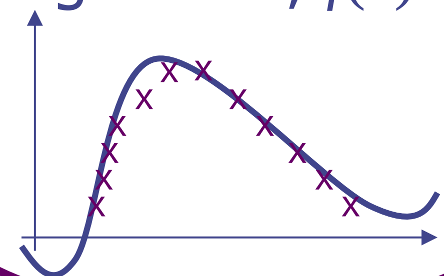


Regression

Classification

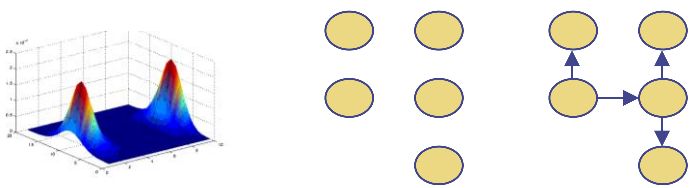


Regression, $f(x)=y$

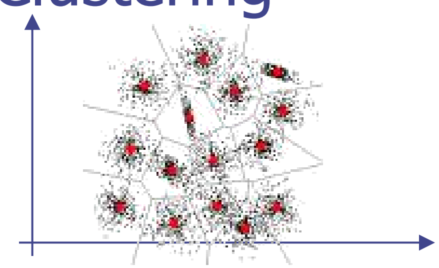


Supervised

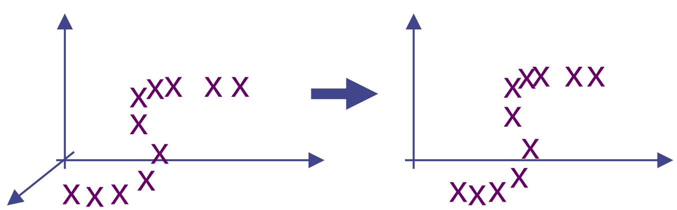
Density/Structure Estimation



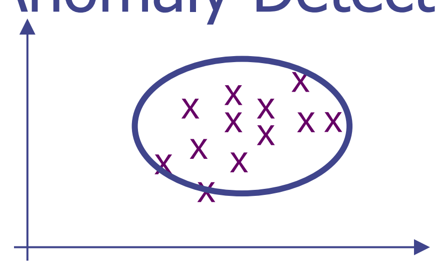
Clustering



Feature Selection



Anomaly Detection



Unsupervised

Outline

- Regression
- Empirical Risk Minimization
- Least Squares
- Higher Order Polynomials
- Under-fitting / Over-fitting
- Cross-Validation

Function Approximation

- Start with training dataset

$$X = \{(x_1, y_1), (x_1, y_1), \ldots, (x_N, y_N)\} \qquad x \in \mathbf{R}^D = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(D) \end{bmatrix}, y \in \mathbf{R}$$

- Have N (input, output) pairs
- Find a function $f(x)$ to predict y from x
  That fits the training data well

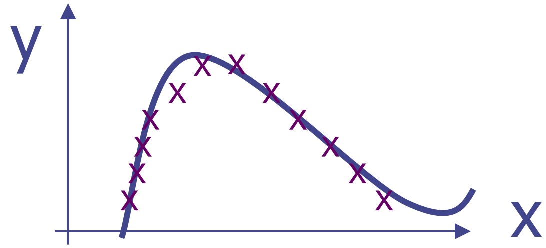
Function Approximation

- Start with training dataset

$$X = \{(x_1, y_1), (x_1, y_1), \dots, (x_N, y_N)\} \qquad x \in \mathbf{R}^D = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(D) \end{bmatrix}, y \in \mathbf{R}$$

- Have N (input, output) pairs
- Find a function $f(x)$ to predict y from x

  That fits the training data well



- Example: predict your grade y in ML

  x = [HW grades; quiz grades; midterm; final]

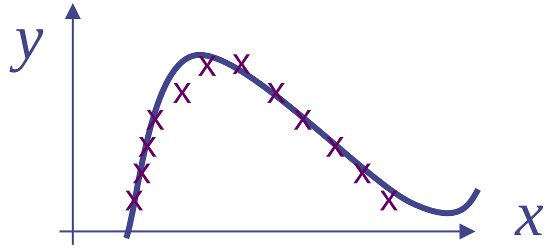
Function Approximation

- Start with training dataset

$$X = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\} \qquad x \in \mathbf{R}^D = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(D) \end{bmatrix}, y \in \mathbf{R}$$

- Have N (input, output) pairs
- Find a function $f(x)$ to predict y from x

  That fits the training data well



- Example: predict your grade y in ML

  x = [HW grades; quiz grades; midterm; final]

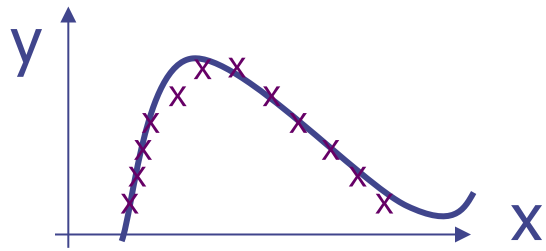
  x = [HW 1-4 grades; quiz 1 grade; midterm]

Function Approximation

- Start with training dataset

$$X = \{(x_1, y_1), (x_1, y_1), \ldots, (x_N, y_N)\} \qquad x \in \mathbf{R}^D = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(D) \end{bmatrix}, y \in \mathbf{R}$$

- Have N (input, output) pairs
- Find a function f(x) to predict y from x
  That fits the training data well



- Example: predict the price of house in
  dollars y using x = [#rooms; latitude; longitude; …]
- Need: a) Way to evaluate how good a fit we have
  b) Class of functions in which to search for f(x)

Empirical Risk Minimization

- Idea: minimize 'loss' on the training data set
- Empirical = use the training set to find the best fit
- Define a loss function of how good we fit a single point:

$$Loss\big(y, f(x)\big) = -\log Prob\big(y|f(x)\big)$$

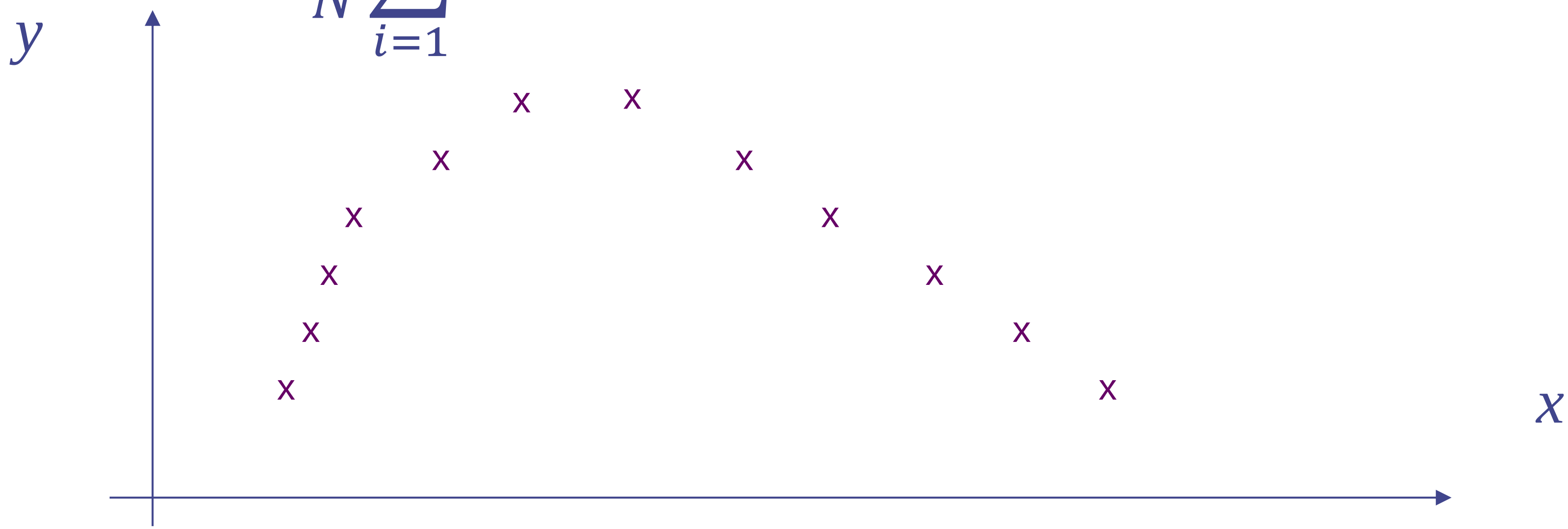
Empirical Risk Minimization

- Idea: minimize ‘loss’ on the training data set
- Empirical = use the training set to find the best fit
- Define a loss function of how good we fit a single point:

$$Loss\big(y, f(x)\big) = -\log Prob\big(y | f(x)\big)$$

- Empirical Risk = the average loss over the dataset

$$R = \frac{1}{N}\sum_{i=1}^{N} Loss(y_i, f(x_i))$$



Empirical Risk Minimization

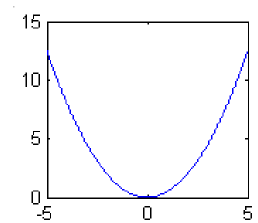
- Idea: minimize 'loss' on the training data set
- Empirical = use the training set to find the best fit
- Define a loss function of how good we fit a single point:

$$Loss\big(y, f(x)\big) = -\log Prob\big(y|f(x)\big)$$

- Empirical Risk = the average loss over the dataset

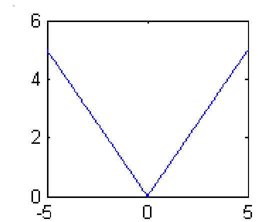
$$R = \frac{1}{N}\sum_{i=1}^{N} Loss(y_i, f(x_i))$$

- Gaussian noise: squared error from y value



$$Loss(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$$

- Other possible loss: absolute error

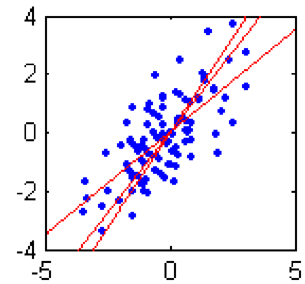


$$Loss(y_i, f(x_i)) = |y_i - f(x_i)|$$

Linear Function Classes

- Linear is simplest class of functions to search over:

$$f(x;\theta) = \theta^T x + \theta_0 = \theta_0 + \sum_{d=1}^{D} \theta_d x(d)$$



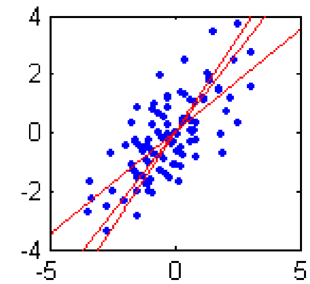
- Start with x being 1-dimensional (D=1):

$$f(x;\theta) = \theta_1 x + \theta_0$$

Linear Function Classes

- Linear is simplest class of functions to search over:

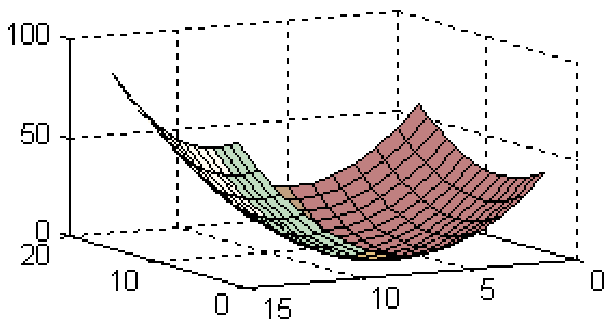
$$f(x;\theta) = \theta^T x + \theta_0 = \theta_0 + \sum_{d=1}^{D} \theta_d x(d)$$



- Start with x being 1-dimensional (D=1):

$$f(x;\theta) = \theta_1 x + \theta_0$$

- Plug in the above & minimize empirical risk over θ



$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0)^2$$

- Note: minimum occurs when $R(\theta)$ gets flat (not always!)
- Note: when $R(\theta)$ is flat, gradient $\nabla_\theta R = 0$

Min by Gradient=0

- Gradient=0 means the partial derivatives are all 0
- Take partials of empirical risk:

$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0)^2$$

$$\nabla_\theta R = \begin{bmatrix} \frac{\partial R}{\partial \theta_0} \\ \frac{\partial R}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Min by Gradient=0

- Gradient=0 means the partial derivatives are all 0

$$\nabla_\theta R = \begin{bmatrix} \frac{\partial R}{\partial \theta_0} \\ \frac{\partial R}{\partial \theta_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Take partials of empirical risk:

$$R(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \theta_1 x_i - \theta_0)^2$$

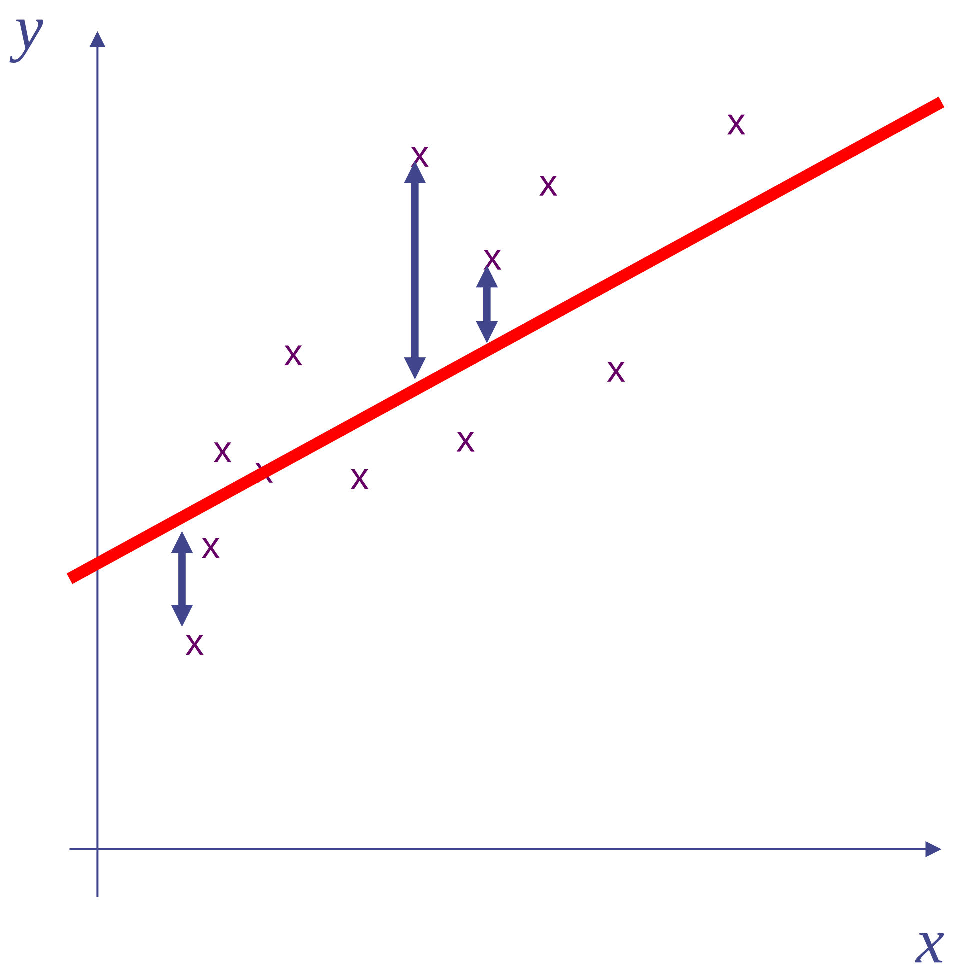
$$\frac{\partial R}{\partial \theta_0} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta_1 x_i - \theta_0)(-1) = 0$$

$$\frac{\partial R}{\partial \theta_1} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \theta_1 x_i - \theta_0)(-x_i) = 0$$

Interpreting the equations

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0) = 0$$

$$\frac{1}{N}\sum_{i=1}^{N}x_i(y_i - \theta_1 x_i - \theta_0) = 0$$

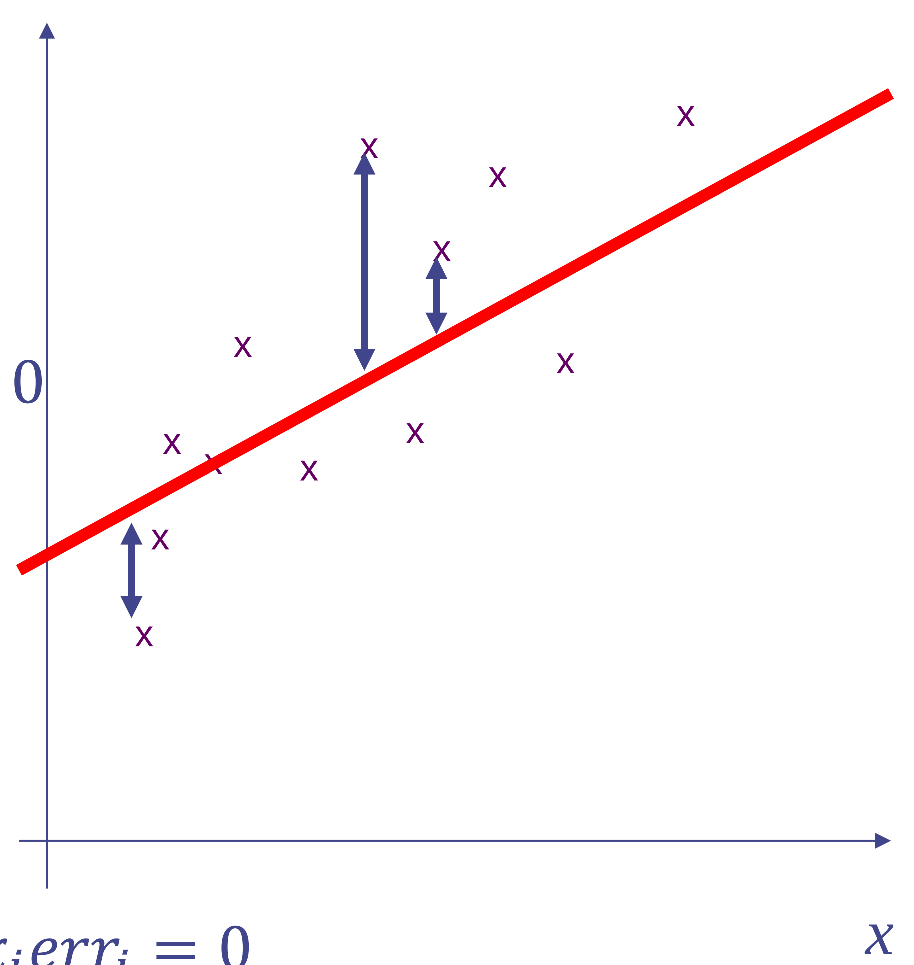


Interpreting the equations

$$\frac{1}{N}\sum_{i=1}^{N}\overbrace{(y_i-\theta_1 x_i-\theta_0)}^{err_i}=0$$

$$\hat{E}(Err)=Avg(err_i)=\frac{1}{N}\sum_{i=1}^{N}err_i=0$$

$$\frac{1}{N}\sum_{i=1}^{N}x_i(y_i-\theta_1 x_i-\theta_0)=0$$



$$\widehat{Cov}(X,Err)=\hat{E}(x_i err_i)=\frac{1}{N}\sum_{i=1}^{N}x_i err_i=0$$

Solving Least Squares

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0) = 0$$

$$\frac{1}{N}\sum_{i=1}^{N}x_i(y_i - \theta_1 x_i - \theta_0) = 0$$

Solving Least Squares

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0) = 0$$

$$\frac{1}{N}\sum_{i=1}^{N}x_i(y_i - \theta_1 x_i - \theta_0) = 0$$

$$\theta_0 = \frac{1}{N}\sum_{i=1}^{N}y_i - \theta_1\frac{1}{N}\sum_{i=1}^{N}x_i$$

$$\theta_1\sum_{i=1}^{N}x_i^2 = \sum_{i=1}^{N}x_i y_i - \theta_0\sum_{i=1}^{N}x_i$$

$$\theta_1 = \frac{\sum x_i y_i - \frac{1}{N}\sum y_i \sum x_i}{\sum x_i^2 - \frac{1}{N}\sum x_i \sum x_i}$$

Multi-Dimensional Regression

- More elegant/general to solve $\nabla_\theta R = 0$ with linear algebra
- Rewrite empirical risk in vector-matrix notation:

$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0)^2$$

$$=$$

Can add more dimensions by adding columns to $\boldsymbol{X}$ matrix and rows to θ vector

Multi-Dimensional Regression

- More elegant/general to solve $\nabla_\theta R = 0$ with linear algebra
- Rewrite empirical risk in vector-matrix notation:

$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0)^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \begin{bmatrix}1 & x_i\end{bmatrix}\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix}\right)^2$$

$$= \frac{1}{2N}\left\|\begin{bmatrix}y_1 \\ \vdots \\ y_N\end{bmatrix} - \begin{bmatrix}1 & x_1 \\ \vdots & \vdots \\ 1 & x_N\end{bmatrix}\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix}\right\|^2$$

$$= \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\theta\|^2$$

Multi-Dimensional Regression

- More elegant/general to solve $\nabla_\theta R = 0$ with linear algebra
- Rewrite empirical risk in vector-matrix notation:

$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i - \theta_0)^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \begin{bmatrix}1 & x_i\end{bmatrix}\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix}\right)^2$$

$$= \frac{1}{2N}\left\|\begin{bmatrix}y_1 \\ \vdots \\ y_N\end{bmatrix} - \begin{bmatrix}1 & x_1 \\ \vdots & \vdots \\ 1 & x_N\end{bmatrix}\begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix}\right\|^2$$

$$= \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\theta\|^2$$

Can add more dimensions by adding columns to $\boldsymbol{X}$ matrix and rows to θ vector

Multi-Dimensional Regression

- More elegant/general to solve $\nabla_\theta R = 0$ with linear algebra
- Rewrite empirical risk in vector-matrix notation:

$$R(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \theta_1 x_i(1) - \cdots - \theta_D x_i(D) - \theta_0)^2$$

$$= \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - [1 \quad x_i(1)\cdots \quad x_i(D)]\begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_D\end{bmatrix}\right)^2$$

$$= \frac{1}{2N}\left\|\begin{bmatrix}y_1\\ \vdots\\ y_N\end{bmatrix} - \begin{bmatrix}1 & x_1(1)\cdots & x_1(D)\\ \vdots & \vdots \quad \ddots & \vdots\\ 1 & x_N(1)\cdots & x_N(D)\end{bmatrix}\begin{bmatrix}\theta_0\\ \theta_1\\ \vdots\\ \theta_D\end{bmatrix}\right\|^2$$

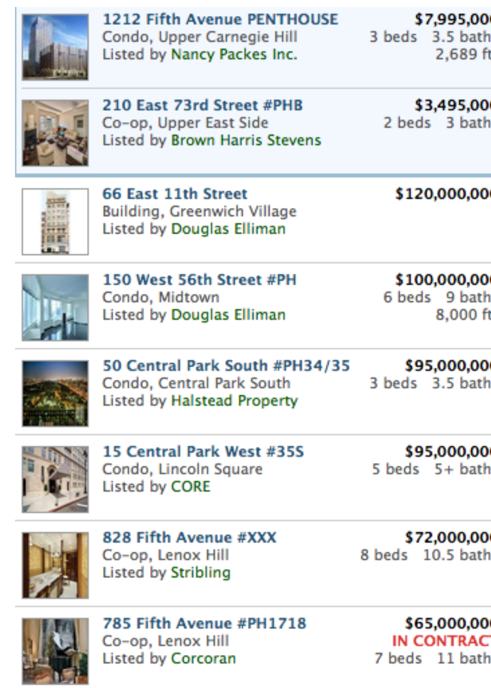
$$= \frac{1}{2N}\|\boldsymbol{y} - \boldsymbol{X}\theta\|^2$$

Can add more dimensions by adding columns to $\boldsymbol{X}$ matrix and rows to θ vector

Multi-Dimensional Regression

- More realistic dataset: many measurements
- Have N apartments each with D measurements
- Each row of X is [#rooms; latitude; longitude,…]

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1(1) \cdots & x_1(D) \\ \vdots & \vdots \quad \ddots & \vdots \\ 1 & x_N(1) \cdots & x_N(D) \end{bmatrix}$$



Multi-Dimensional Regression

- Solving gradient=0 $\nabla_\theta R = 0$

$$\nabla_\theta \left(\frac{1}{2N} \|\boldsymbol{y} - \boldsymbol{X}\theta\|^2 \right) = 0$$

Multi-Dimensional Regression

- Solving gradient=0 $\nabla_\theta R = 0$

$$\nabla_\theta \left(\frac{1}{2N} \|\boldsymbol{y} - \boldsymbol{X}\theta\|^2 \right) = 0$$

$$\frac{1}{2N} \nabla_\theta \big((\boldsymbol{y} - \boldsymbol{X}\theta)^T (\boldsymbol{y} - \boldsymbol{X}\theta) \big) = 0$$

$$\frac{1}{2N} \nabla_\theta (y^T y - 2y^T \boldsymbol{X}\theta + \theta^T \boldsymbol{X}^T \boldsymbol{X}\theta) = 0$$

$$\frac{1}{2N} (-2\boldsymbol{X}^T y + 2\boldsymbol{X}^T \boldsymbol{X}\theta) = 0$$

$$\boldsymbol{X}^T \boldsymbol{X}\theta = \boldsymbol{X}^T y$$

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

Multi-Dimensional Regression

- Solving gradient=0

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

- If the matrix $\boldsymbol{X}$ is skinny, the solution is probably unique
- If $\boldsymbol{X}$ is fat (more dimensions than points) we get multiple solutions for theta which give zero error.

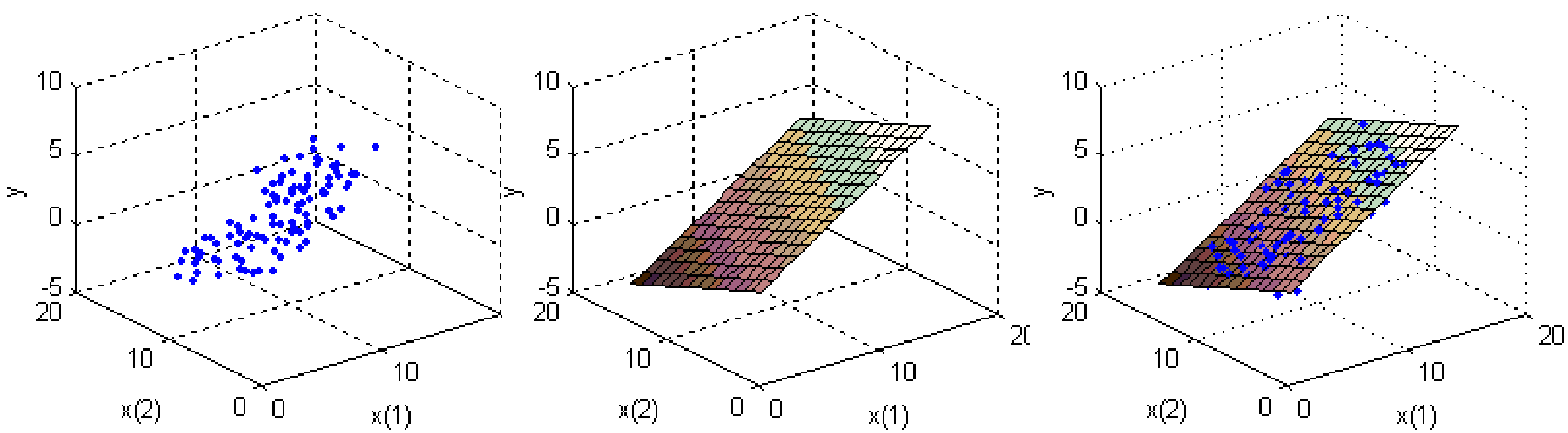
- The pseudeoinverse (numpy.linalg.pinv(X)) returns the theta with zero error and which has the smallest norm.

$$min_\theta ||\theta||^2 \; such \; that \; \boldsymbol{X}\theta = y$$

2D Linear Regression

- Once best θ^* is found, we can plug it into the function:

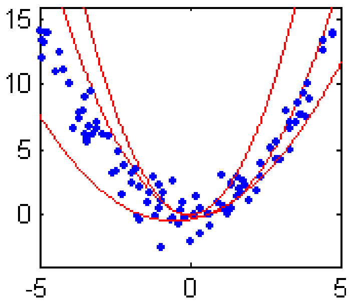
$$f(x;\theta) = \theta_2^* x(2) + \theta_1^* x(1) + \theta_0^*$$



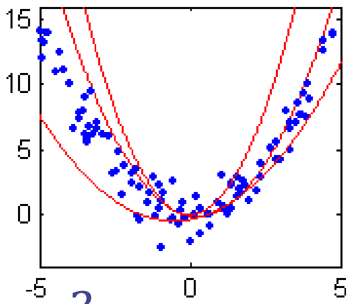
- What would a fat X look like?

Polynomial Function Classes

- Back to 1-dim x (D=1) BUT Nonlinear
- Polynomial: $$f(x;\theta) = \sum_{p=0}^{P} \theta_p x^p$$



Polynomial Function Classes



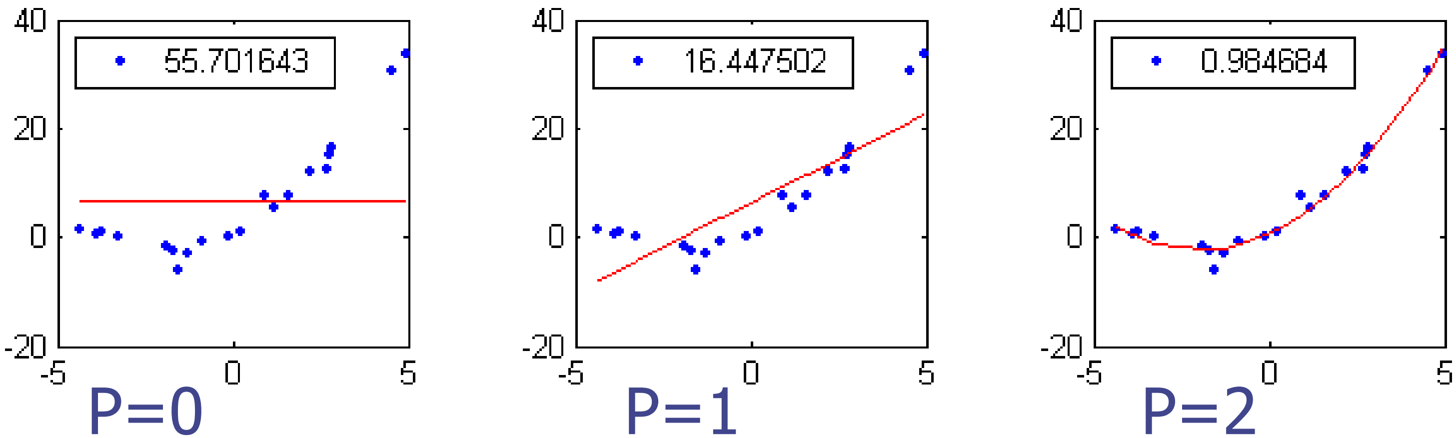
- Back to 1-dim x (D=1) BUT Nonlinear
- Polynomial: $$f(x;\theta) = \sum_{p=0}^{P} \theta_p x^p$$
- Writing Risk:

$$R(\theta) = \frac{1}{2N} \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} - \begin{bmatrix} 1 & x_1^1 \cdots & x_1^P \\ \vdots & \vdots \ \ddots & \vdots \\ 1 & x_N^1 \cdots & x_N^P \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_P \end{bmatrix} \right\|^2$$

- Order-P polynomial regression fitting for 1D variable is same as P-dimensional linear regression!
- Construct a multidim $\boldsymbol{X}$-vector from x scalar $$\boldsymbol{X}_i = [x_i^0 \quad x_i^1 \quad x_i^2 \quad x_i^3]$$
- More generally any $$\boldsymbol{X}_i = [\phi_0(x_i) \quad \phi_1(x_i) \quad \phi_2(x_i) \quad \phi_3(x_i)]$$

Underfitting/Overfitting

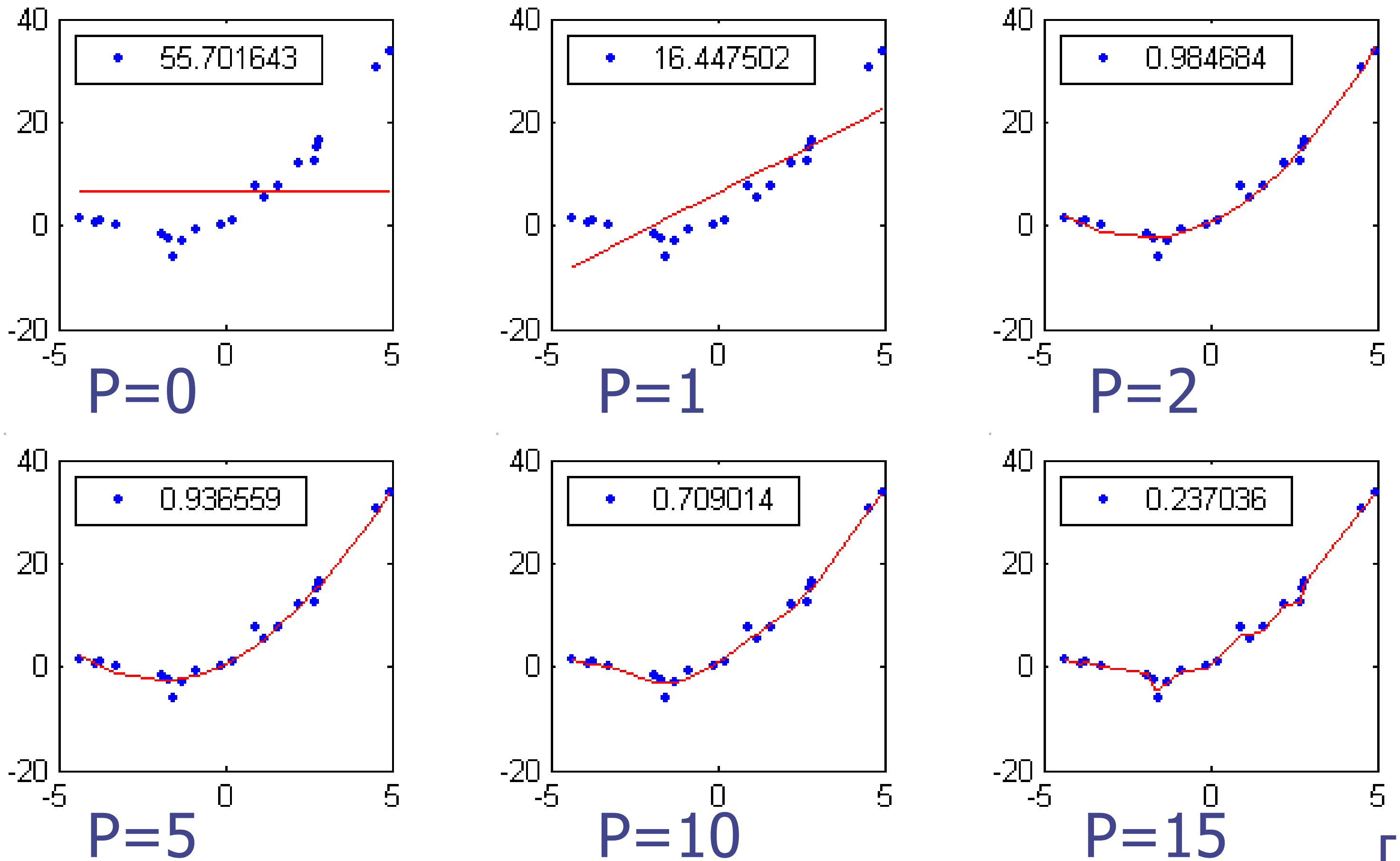
- Try varying P. Higher P fits a more complex function class
- Observe $R(\theta^*)$ drops with bigger P



[Bishop]

Underfitting/Overfitting

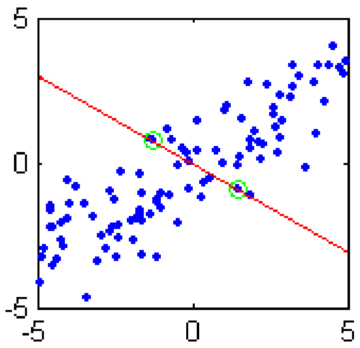
- Try varying P. Higher P fits a more complex function class
- Observe $R(\theta^*)$ drops with bigger P



[Bishop]

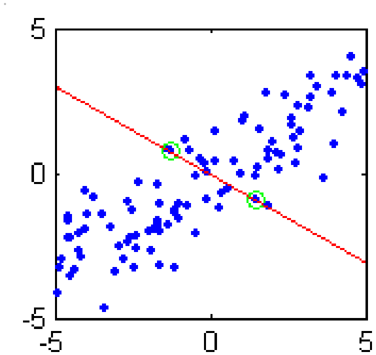
Evaluating The Regression

- Unfair to use empirical to find best order P
- High P (vs. N) can overfit, even linear case!
- min $R(\theta^*)$ not on training but on future data
- Want model to *Generalize* to future data



Evaluating The Regression

- Unfair to use empirical to find best order P
- High P (vs. N) can overfit, even linear case!
- min $R(\theta^*)$ not on training but on future data
- Want model to *Generalize* to future data



True loss: $R_{true}(\theta) = \int p(x,y) \frac{1}{2} (y - \theta^T x)^2 dx\, dy$

- One approach: split data into training / testing portion

$\{(x_1, y_1), \ldots, (x_N, y_N)\}$ $\{(x_{N+1}, y_{N+1}), \ldots, (x_{N+M}, y_{N+M})\}$

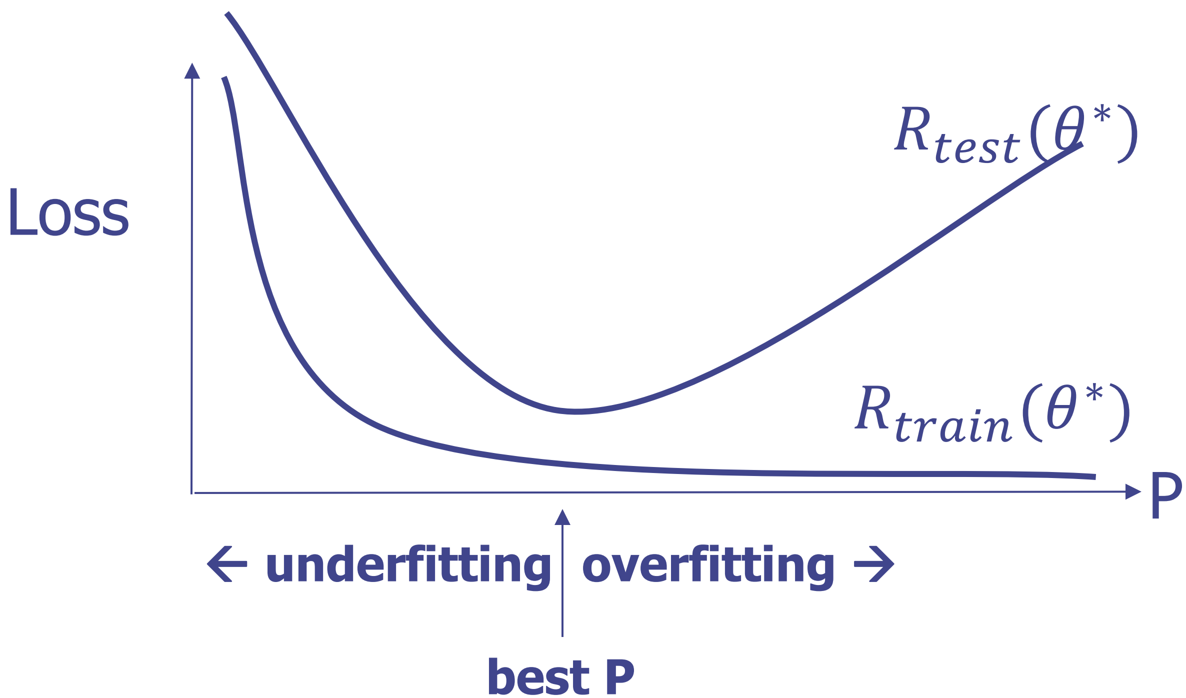
- Estimate θ* with training loss: $R_{train}(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \theta^T x_i)^2$

- Evaluate P with testing loss:

$$R_{test}(\theta^*) = \frac{1}{2M} \sum_{i=N+1}^{N+M} (y_i - \theta^{*T} x_i)^2$$

Crossvalidation

- Try fitting with different polynomial order P
- Select P which gives lowest $R_{test}(\theta^*)$



- Think of P as a measure of the complexity of the model
- Higher order polynomials are more flexible and complex

Summary

- Regression: approximating a function
- Minimizing loss function
- Least squares solution to linear regression
- Vector/matrix representation
- Hi-dimension, polynomial regression
- Overfit/underfit