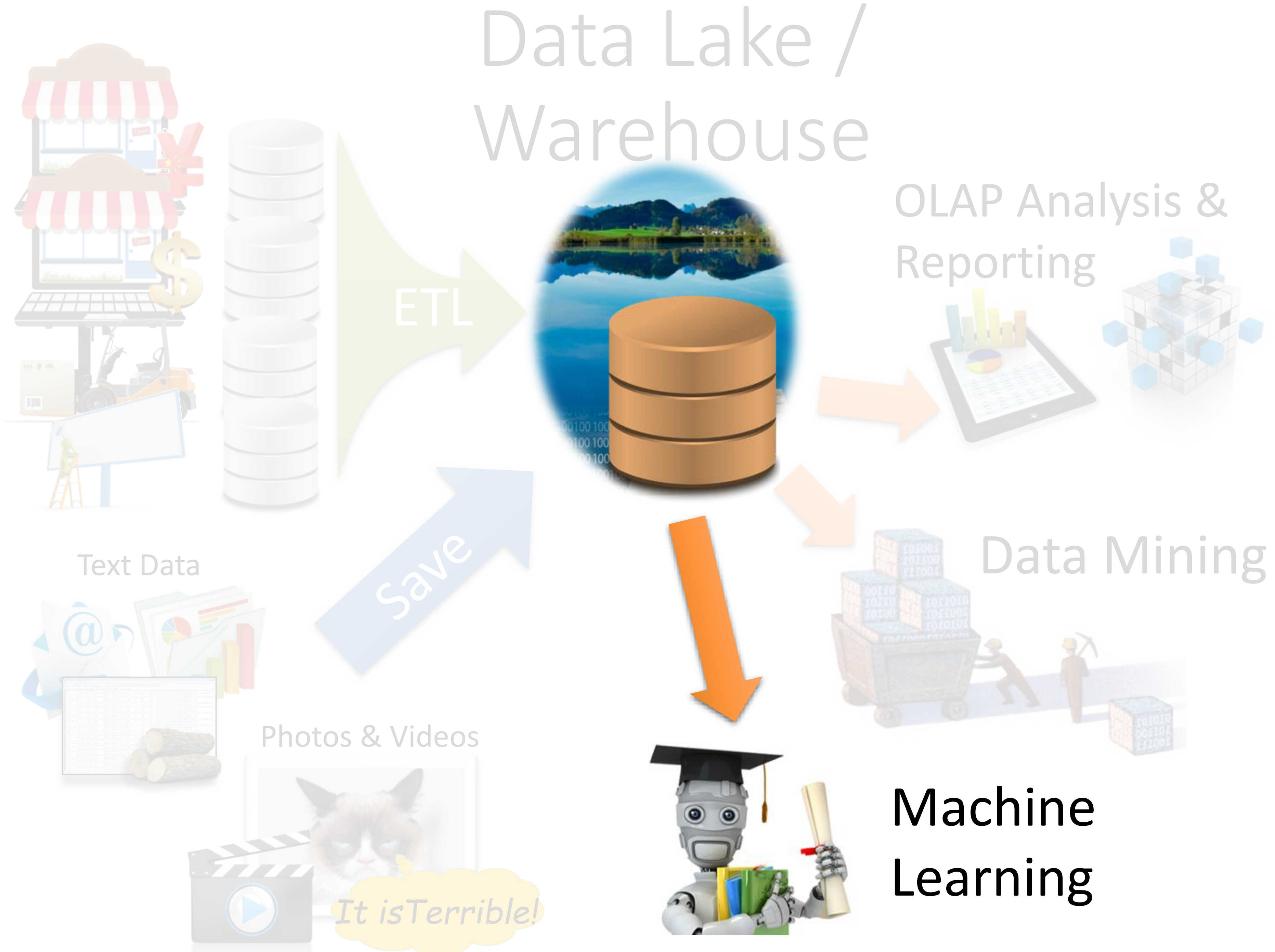
CS150: Database & Datamining

Lecture 21: Analytics & Machine Learning III

Xuming He

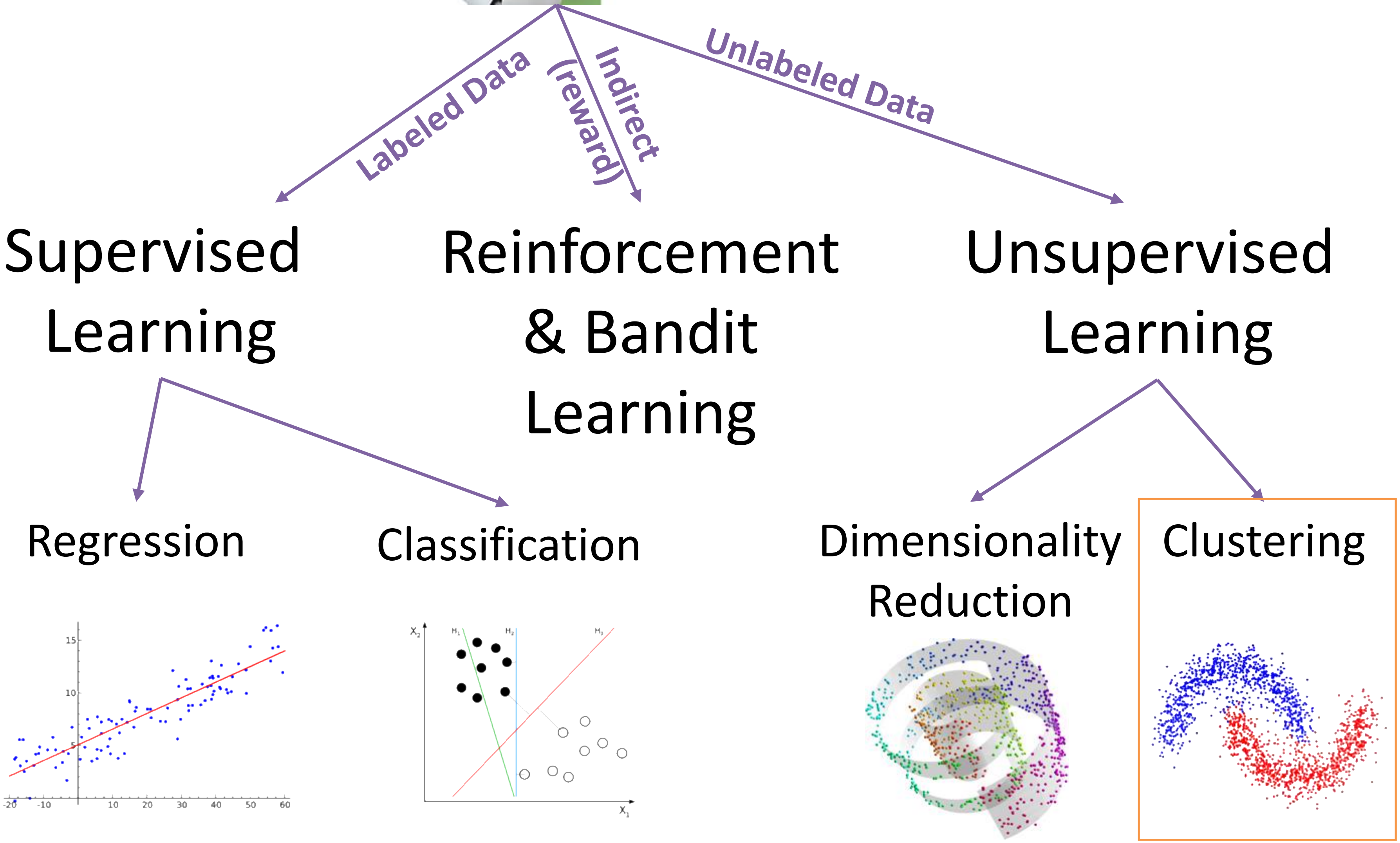
Spring 2019

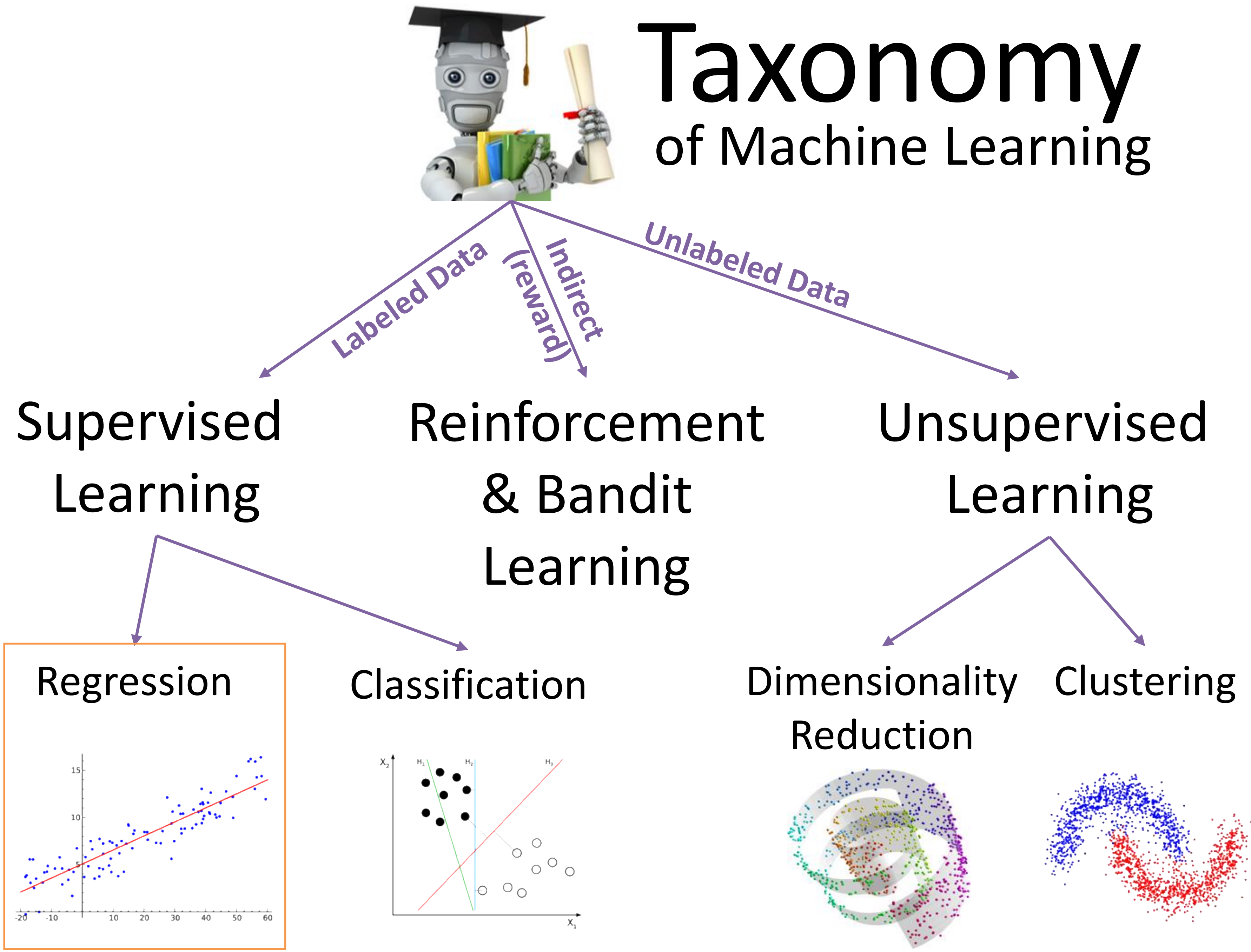
Acknowledgement: Slides are adopted from the Berkeley course CS186 by Joey Gonzalez and Joe Hellerstein, Stanford CS145 by Peter Bailis.



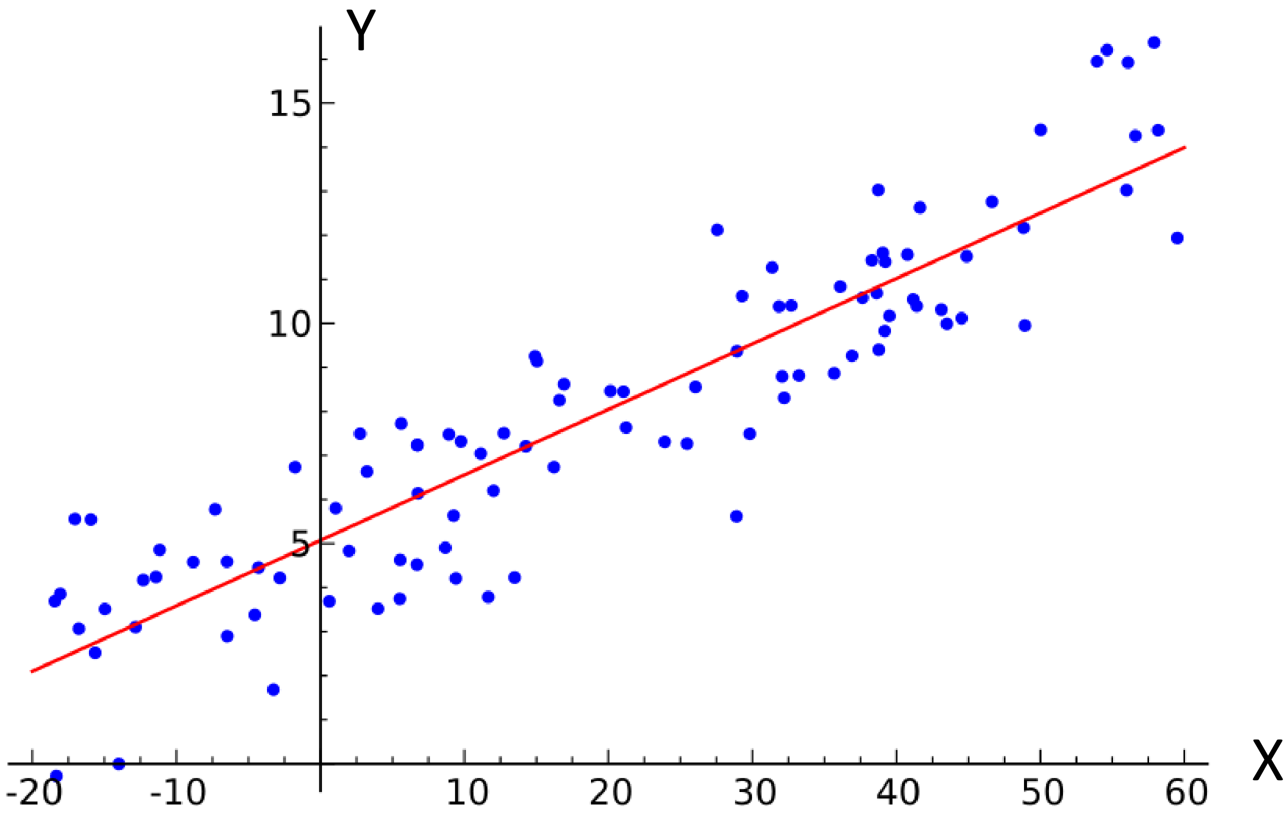
Data Lake /
Warehouse
ETL
OLAP Analysis &
Reporting
Data Mining
Save
Text Data
Photos & Videos
It isTerrible!
Machine
Learning

Taxonomy
of Machine Learning




Taxonomy
of Machine Learning
Labeled Data
Indirect (reward)
Unlabeled Data
Supervised Learning
Reinforcement & Bandit Learning
Unsupervised Learning
Regression
Classification
Dimensionality Reduction
Clustering

Simple Linear Regression



Linear Regression is Powerful

- One of the most widely used techniques
- Fundamental to many larger models
  - Logistic Regression
  - Collaborative filtering
- Easy to interpret
  - e.g., the weights tell us something about the features
    - Positive or negative relationships …
- Efficient to solve
  - Fast numerical methods
  - Closed form solutions

The Linear Model

Data:

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17
…	…	…

Real Valued Observations

$$y = \underbrace{\overbrace{\theta^T}^{\text{Vector of Parameters}} \overbrace{x}^{\text{Vector of Features}}}_{\sum_{i=1}^{p} \theta_i x_i} + \epsilon$$

Real Value Noise

Linear Combination of Covariates

$$\sum_{i=1}^{p} \theta_i x_i$$

$$\theta, x \in \mathbb{R}^p$$

The Linear Model

Data:

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17
…	…	…

Vector of Parameters

Vector of Features

Real Valued Observations

$$y = \underbrace{\overbrace{\theta}^{\text{Vector of Parameters}}{}^T \overbrace{x}^{\text{Vector of Features}}}_{\text{Linear Combination of Covariates}} + \epsilon$$

Real Value Noise

$$x \in \mathbb{R}^p$$

Linear Combination of Covariates

$$\sum_{i=1}^{p} \theta_i x_i$$

Noise Model:

$$\epsilon \sim N(0, \sigma^2)$$

“Real” data doesn’t typically consist of entirely **real** valued features.

$$\theta, x \in \mathbb{R}^p$$

Real Data and Vector Spaces

- What about data with more complex schemas?

x_1	x_2	Date	prod_id	comment	y
1.1	2.7	8/21/16	7	"the best glider …"	3.6
4.2	3.2	8/14/16	3	"vacation for two …"	7.5
9.8	9.2	9/20/16	4	"A special gift for …"	17
…	…	…	…	…	…

  - The math wants the features to be vectors …
- How do we encode dates, categorical fields, and text?

Feature Engineering

- A key part of most machine learning applications

- Common tasks:
  - Transforming raw features into **vector representations**
  - Encoding **prior knowledge** (e.g., translating currencies)
  - Transformations that **increase the expressivity** of the model … (more on this soon)

- Critical to model performance:
  - **engineers compete** to get the best features

- A few standard techniques (that we will cover):
  - one-hot encoding
  - bag-of-words

Encoding Categorical Data

- How do we represent fields like "Product Category"
- **Proposal 1:** *Enumerate categories*
  - Sports = 1, Furniture = 2, Clothing = 3, Shoes = 4, …
  - Store field number as a feature
  - **Implications**:
    - **similarity:** sports is closer to Furniture than shoes
    - **magnitude:** larger values ➔ ?
  - Not typically used (unless there are two categories ...)

One-hot encoding

- How do we represent fields like "Product Category"
- **Proposal 1:** *Enumerate categories*
  - Not typically used (unless there are two categories ...)
- **Proposal 2:** *Encode as binary vectors:*
  - Very commonly used and built-in to many packages
  - Enumerate all possible product categories (*m*)
  - Add *m* additional features to the record:
  - Put a one in the feature corresponding to the product category and a zero everywhere else.

Sports	Furniture	Clothing	Shoes	Electronics	Music	Books
0	0	0	0	1	0	0

Working with Text Data

➢ How do we convert text to vectors?

"Learning about machine learning is fun."

aardvark	aardwolf		**fun**		**learning**		**machine**		zyzzyva
0	0	…	1	…	2	…	1	…	0

Vector

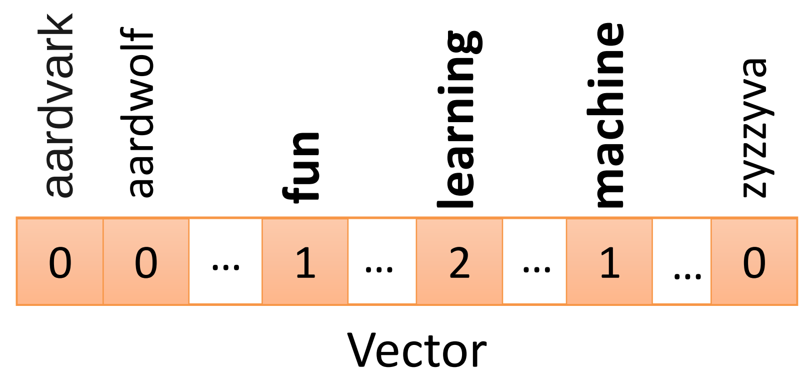
➢ Bag-of-words model

- Transform emails into *d*-dimensional vectors
  - *d* is the number of unique words in the language (big!)
- Each entry is number of occurrences of that word
- **Sparse**: Most words don't occur in most emails
- **Remove Stop-Words:** common words that provide little information (e.g., "is", "about")

Working with Text Data

➢ **Features:** Transforming the data

If all you had was this vector could you tell what the passage is about?



- Transform emails into *d*-dimensional vectors
  - *d* is the number of unique words in the language (big!)
- Each entry counts the number of words in that email
- **Sparse:** Most words don't occur in most emails
- **Remove Stop-Words:** common words that provide little information (e.g., "is", "about")


which
could
this

The Linear Model

Data:

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17
…	…	…

$$f_\theta(x) := \overbrace{\theta}^{\text{Vector of Parameters}}{}^T \overbrace{x}^{\text{Vector of Features}}$$

- ➢ Encode data is real valued vectors
- ➢ **Next:** find the optimal value for θ
  - How?

$$\theta, x \in \mathbb{R}^p$$

Finding the Best Parameters

Model: $$f_\theta(x) := \theta^T x$$

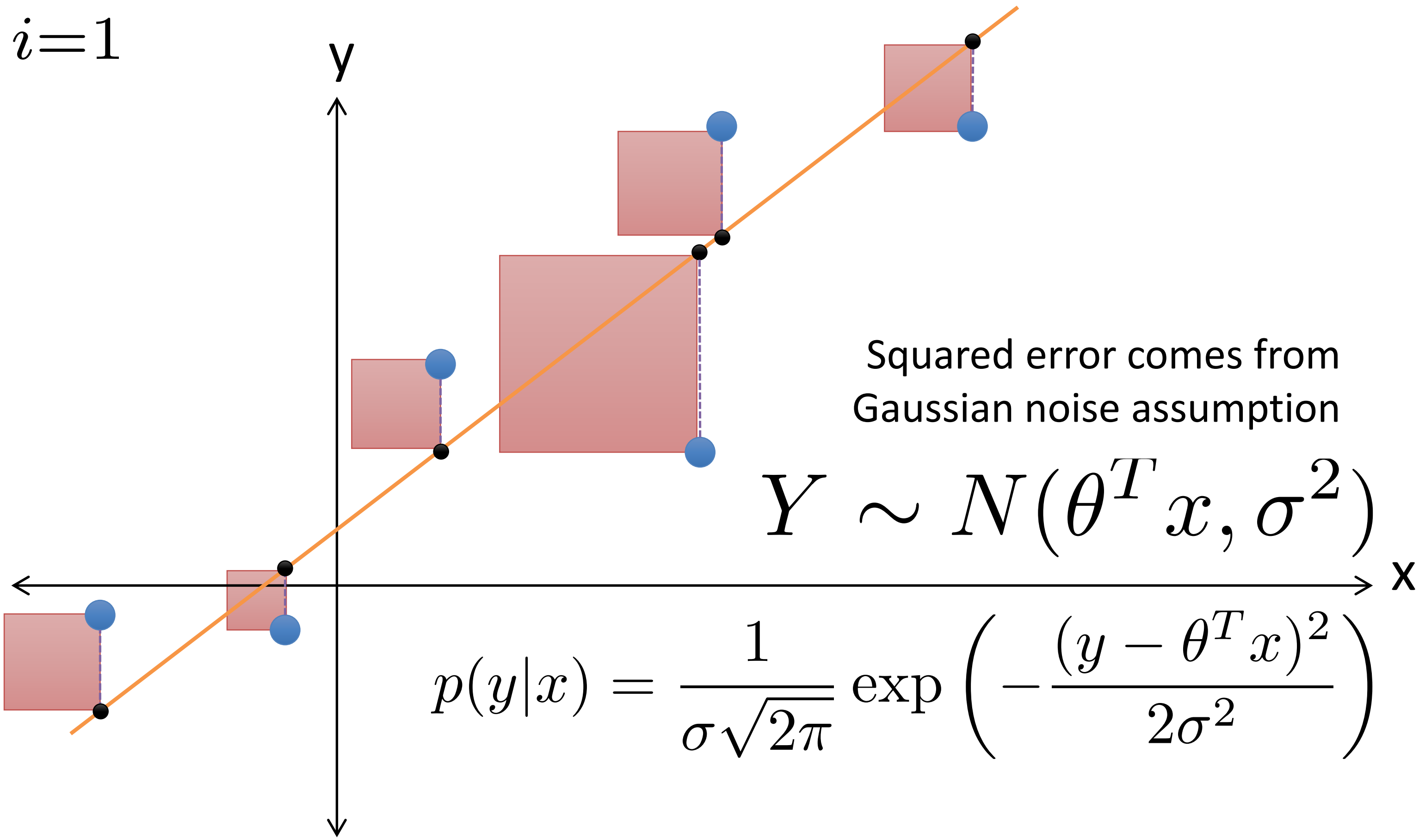
Step 1: define a **Loss Function:** *Average Prediction Error*

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \theta^T x_i\right)^2$$

➢ Difference between **true (y_i)** and **predicted $f_\theta(x_i)$** values

The meaning of Squared Loss (Error)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2$$



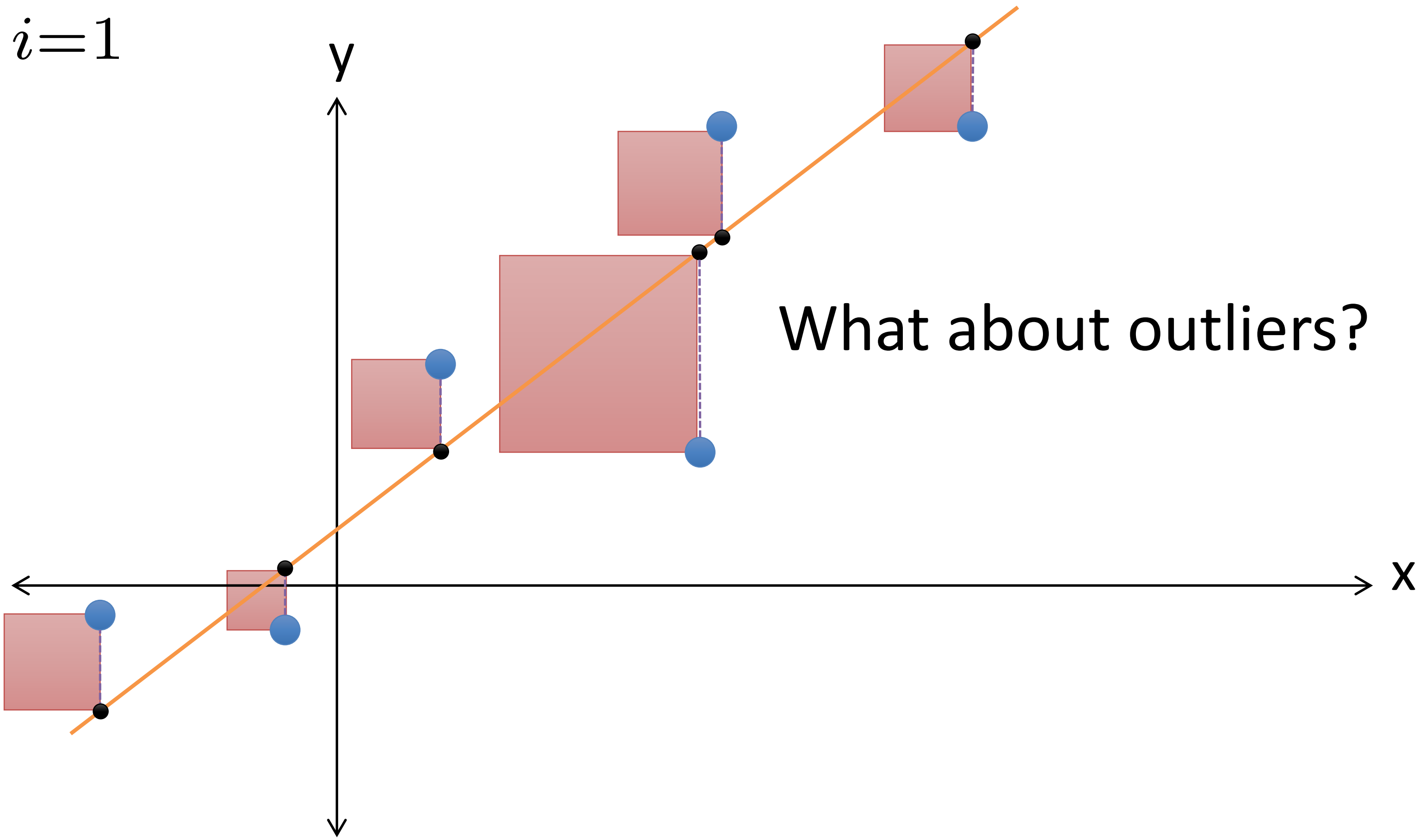
Squared error comes from
Gaussian noise assumption

$$Y \sim N(\theta^T x, \sigma^2)$$

$$p(y|x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\theta^T x)^2}{2\sigma^2}\right)$$

The meaning of Squared Loss (Error)

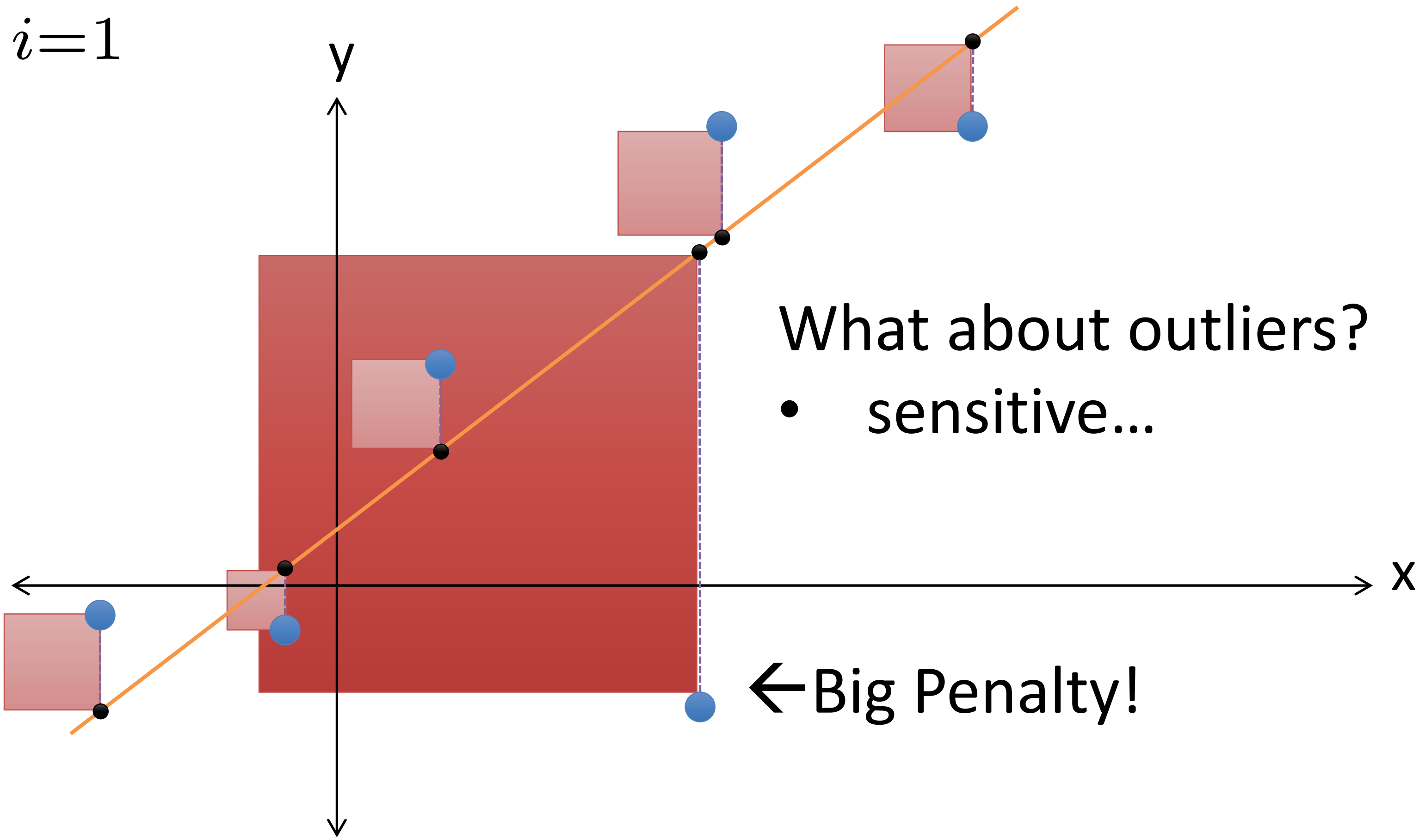
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2$$



What about outliers?

The meaning of Squared Loss (Error)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2$$



What about outliers?

- sensitive…

←Big Penalty!

Finding the Best Parameters

Model: $f_\theta(x) := \theta^T x$

Step 1: define a **Loss Function:**

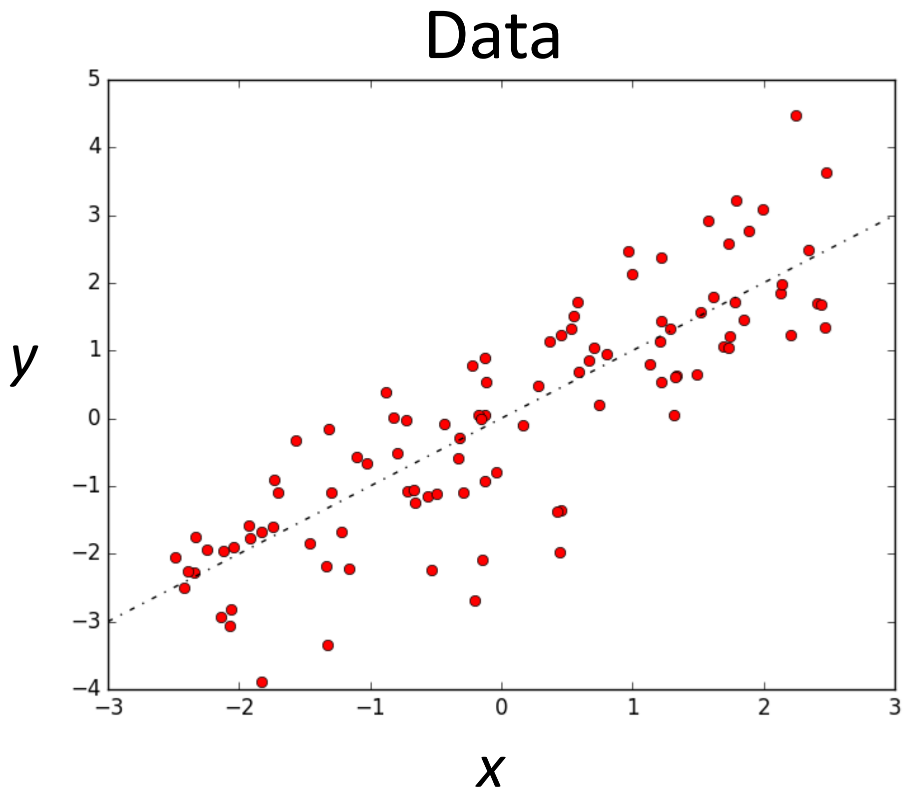
$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2$$

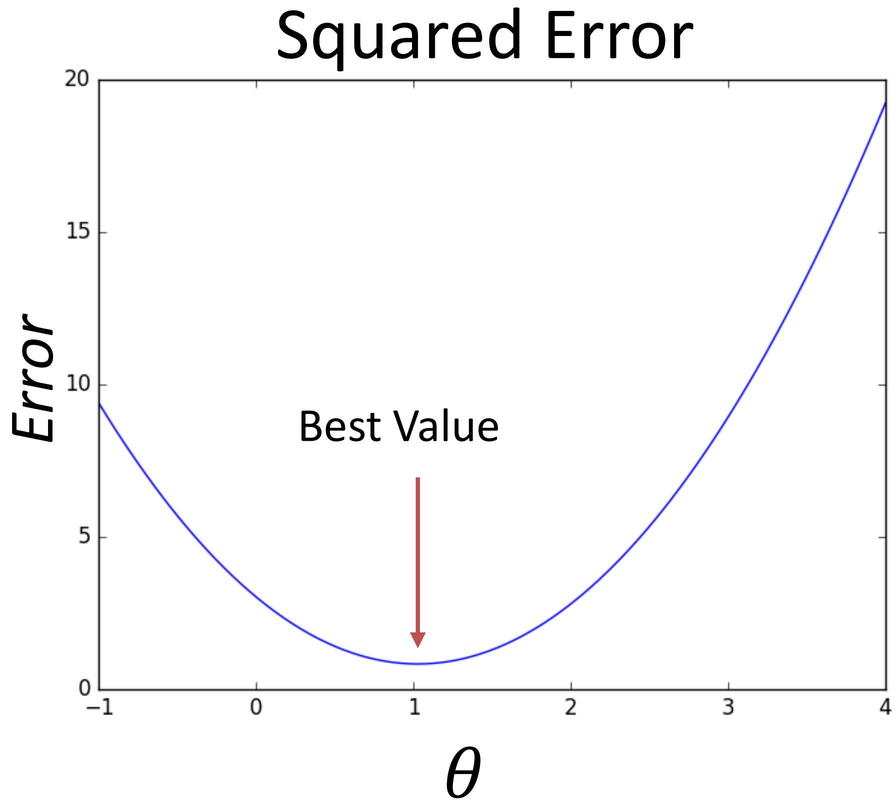
Step 2: Search for best model parameters θ

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2$$

Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

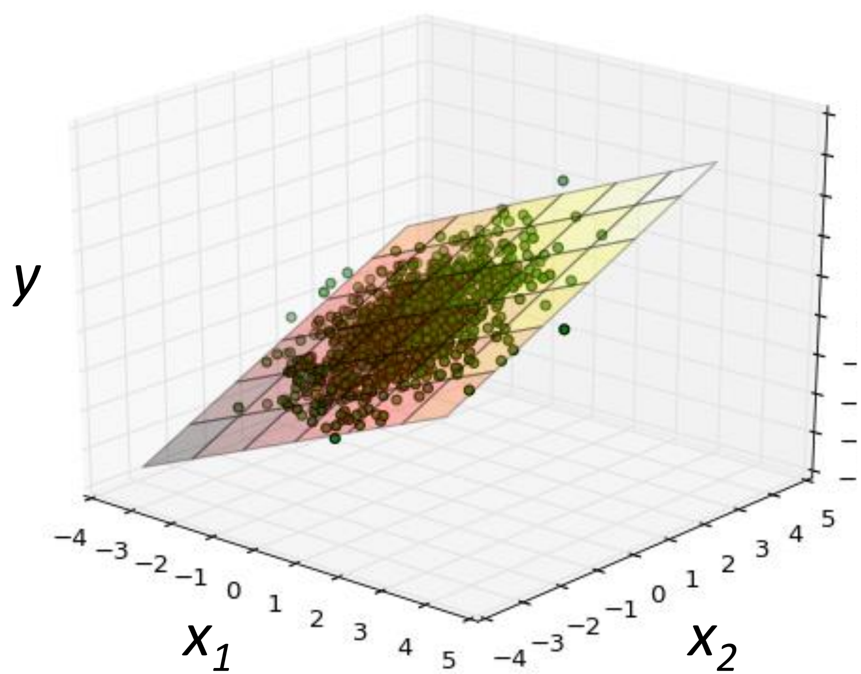




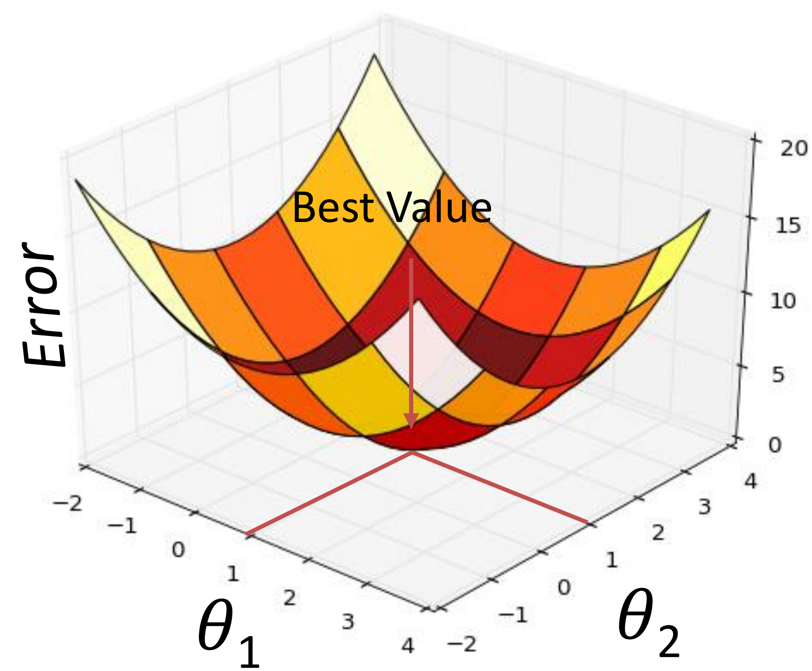
Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

Data



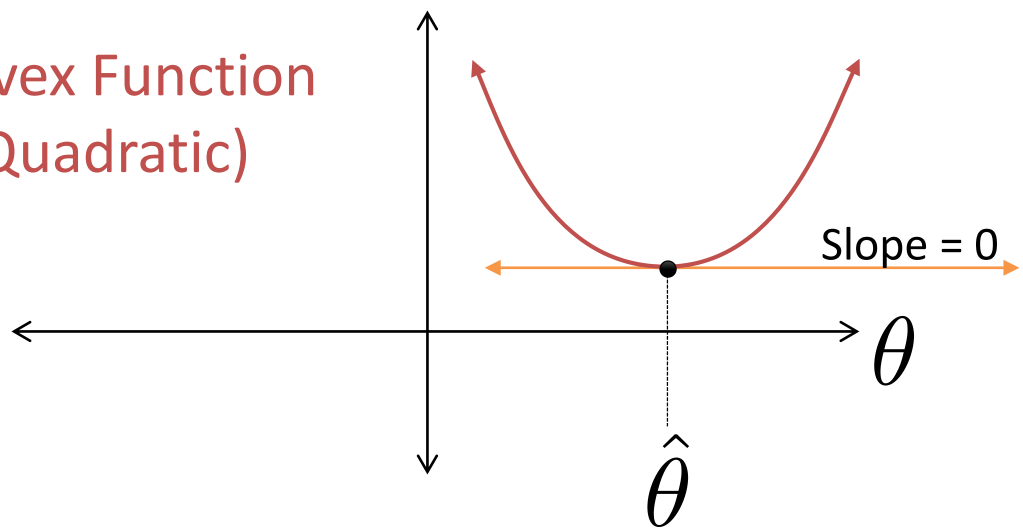
Squared Error



Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

Convex Function (Quadratic)



- Take the gradient and set it equal to zero

Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

- Taking the gradient

Chain Rule

$$\nabla_\theta \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 = -2\frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i) x_i$$

$$= -2\frac{1}{n} \sum_{i=1}^{n} y_i x_i + 2\frac{1}{n} \sum_{i=1}^{n} (\theta^T x_i) x_i$$

- Setting equal to zero and solving for θ (sys. Linear eq.)

$$\sum_{i=1}^{n} (\theta^T x_i) x_{ij} = \sum_{i=1}^{n} y_i x_{ij} \qquad \forall j \in \{1, \ldots, d\}$$

Easier in matrix form …

Writing the data in Matrix form

- Represent data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ as:

Covariate (Design) Matrix

$$X = n\left\{ \underbrace{\begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \cdots & \\ — & x_n & — \end{pmatrix}}_{p} \right. \in \mathbb{R}^{np}$$

Response Vector

$$Y = n\left\{ \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{1} \right. \in \mathbb{R}^{n}$$

Minimizing the Squared Error

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

- Setting equal to zero and solving for θ:

$$\sum_{i=1}^{n} (\theta^T x_i) x_i = \sum_{i=1}^{n} y_i x_i \Rightarrow X^T X \theta = X^T y$$

- Normal Equation:

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

- Solved using any standard linear algebra library

Examples

- ➢Goal: Predict movie rating automatically

复仇者联盟3：无限战争 Avengers: Infinity War (2018)

导演: 安东尼·罗素 / 乔·罗素
编剧: 杰克·科比 / 克里斯托弗·马库斯 / 斯蒂芬·麦克菲利 / 吉姆·斯特林
主演: 小罗伯特·唐尼 / 克里斯·海姆斯沃斯 / 克里斯·埃文斯 / 马克·鲁弗洛 / 乔什·布洛林 / 更多...
类型: 动作 / 科幻 / 奇幻 / 冒险
官方网站: marvel.com/avengers
制片国家/地区: 美国
语言: 英语
上映日期: 2018-05-11(中国大陆) / 2018-04-23(加州首映) / 2018-04-27(美国)
片长: 150分钟
又名: 复联3 / 复仇者联盟：无限之战(台) / 复仇者联盟3：无尽之战 / Avengers: Infinity War - Part I / The Avengers 3: Part 1
IMDb链接: tt4154756

豆瓣评分

8.5

182012人评价

5星 44.8%
4星 38.6%
3星 14.1%
2星 1.8%
1星 0.8%

好于 96% 科幻片
好于 97% 动作片

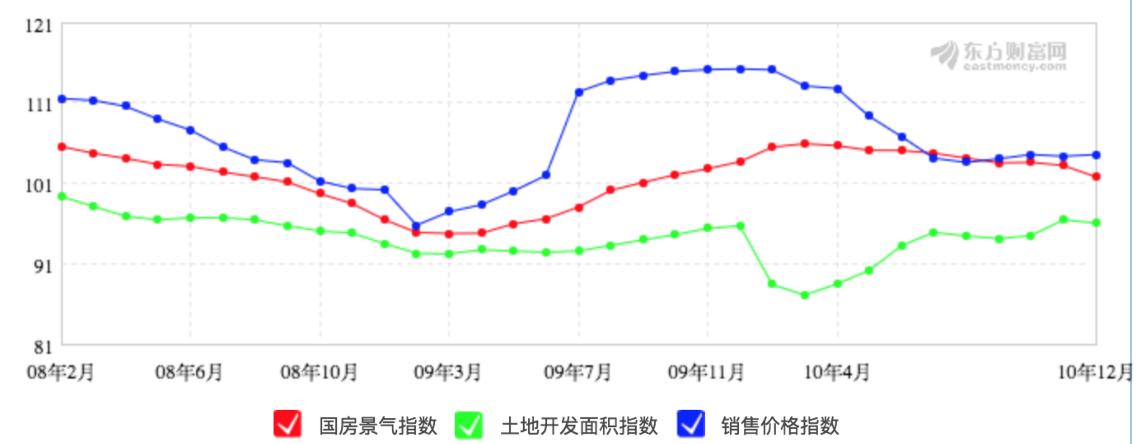
Examples

- Goal: Predict the price of the house

中国

- ·居民消费价格指数(CPI)
- ·工业品出厂价格指数(PPI)
- ·国内生产总值(GDP)
- ·采购经理人指数(PMI)
- ·城镇固定资产投资
- ·房价指数（08—10年）
- ·新房价指数
- ·企业景气及企业家信心指数
- ·工业增加值增长
- ·企业商品价格指数
- ·消费者信心指数
- ·社会消费品零售总额
- ·货币供应量
- ·海关进出口增减情况一览表
- ·全国股票交易统计表
- ·外汇和黄金储备
- ·交易结算资金(银证转账)
- ·股票账户统计表(新)
- ·股票账户统计表
- ·外商直接投资数据(FDI)
- ·财政收入

中国 房价指数



月 份	国房景气指数		土地开发面积指数		销售价格指数	
	指数值	同比增长	指数值	同比增长	指数值	同比增长
2010年12月份	101.79	-1.80%	96.03	0.36%	104.54	-9.25%
2010年11月份	103.20	0.41%	96.42	1.11%	104.34	-9.36%
2010年10月份	103.57	1.51%	94.46	-0.14%	104.55	-8.99%
2010年09月份	103.52	2.41%	94.10	0.18%	104.03	-9.05%
2010年08月份	104.11	4.03%	94.42	1.25%	103.57	-8.96%
2010年07月份	104.72	6.85%	94.87	2.51%	104.10	-7.34%

Parallelize the Linear Regression

- Question: How do we estimate the linear model with a large, distributed database?

Recall: Linear Regression

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2$$

- Setting equal to zero and solving for θ:

$$\sum_{i=1}^{n} (\theta^T x_i) x_i = \sum_{i=1}^{n} y_i x_i \Rightarrow X^T X \theta = X^T y$$

- Normal Equation:

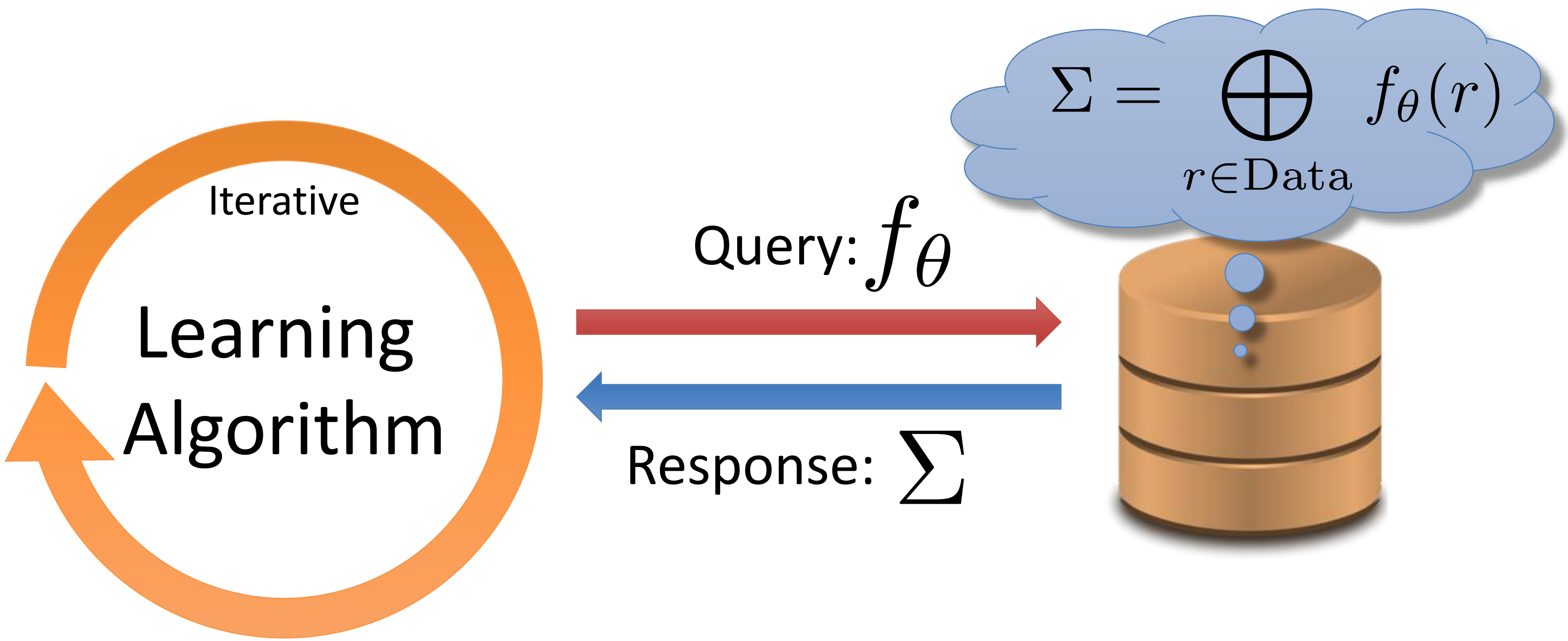
$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

- Solved using any standard linear algebra library

Can we compute

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

using the statistical query pattern?



Can we compute

$$\hat{\theta} = (\underbrace{X^T X}_{\text{Query 1}})^{-1} \underbrace{X^T Y}_{\text{Query 2}}$$

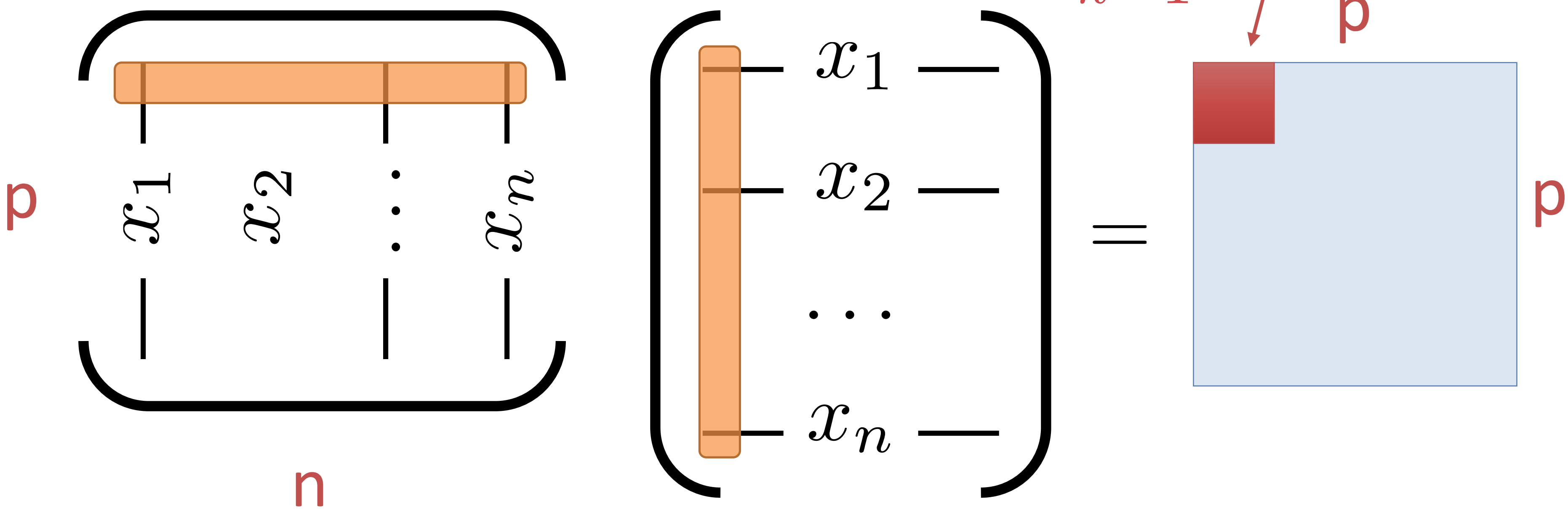
using the statistical query pattern?

Break computation into two queries

Computing Query 1

$$X^T X =$$

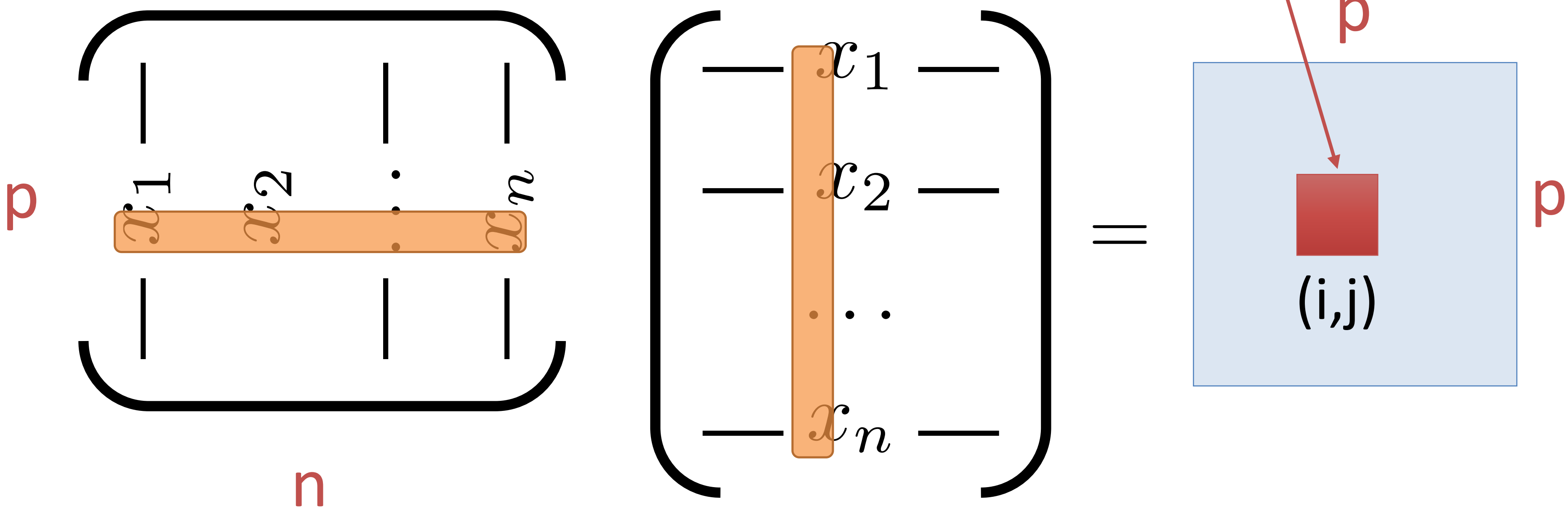
$$\sum_{k=1}^{n} X_{k,1} X_{k,1}$$



Computing Query 1

$$X^T X =$$

$$\sum_{k=1}^{n} X_{k,i} X_{k,j}$$



Computing Query 1

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17
…	…	…

- Compute the row-wise some:

$$X^T X = \sum_{i=1}^{n} x_i x_i^T$$

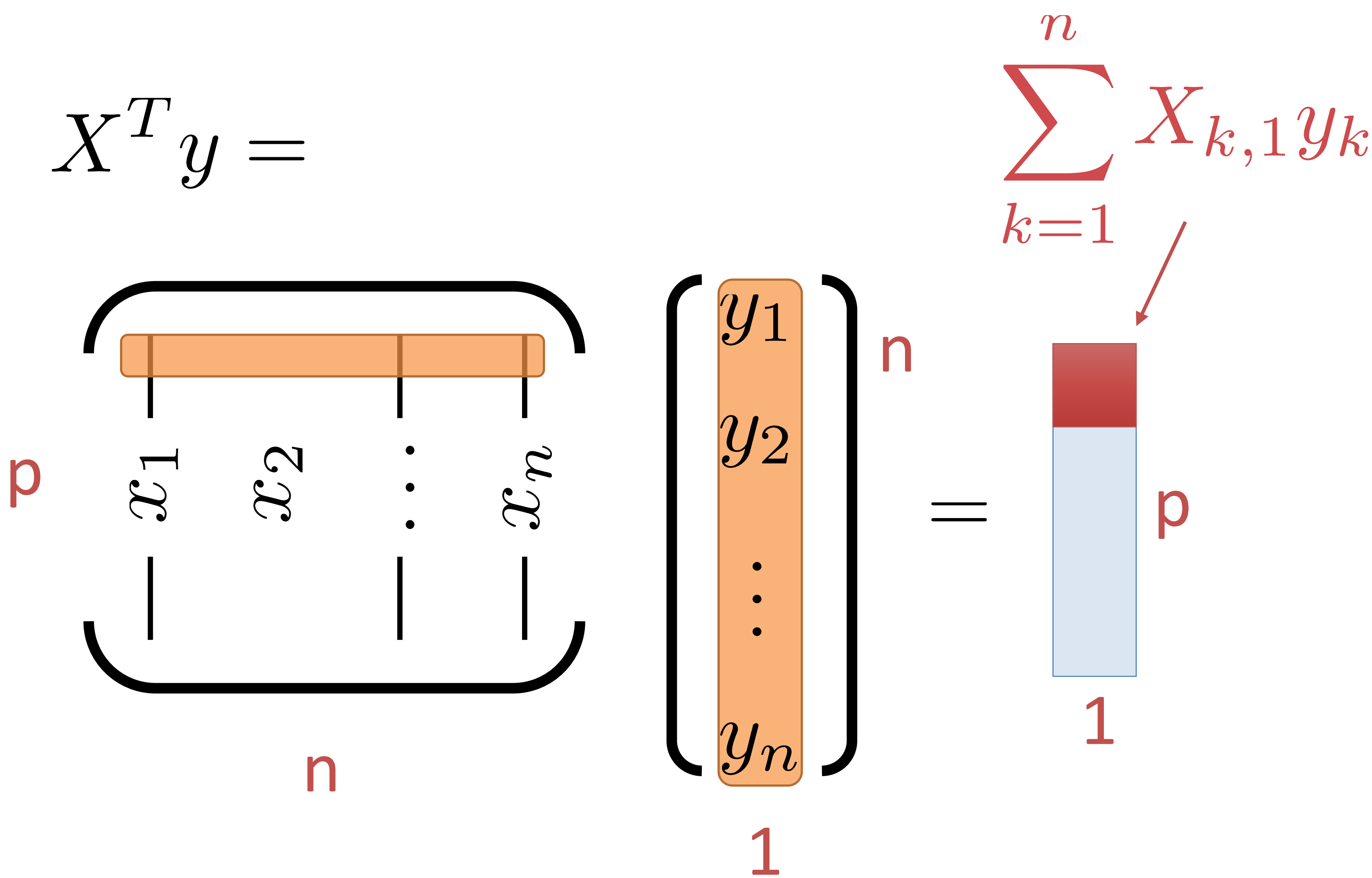
  - **MapFunction(x)**: computes *p* by *p* outer product: xx^T
  - **ReduceFunction**: matrix sum:

- Pure SQL Expression:

```
SELECT
      sum(x1*x1) AS c11, sum(x1*x2) AS c12,
      ~~sum(x2*x1) AS c21~~, sum(x2*x2) AS c22
FROM data
```

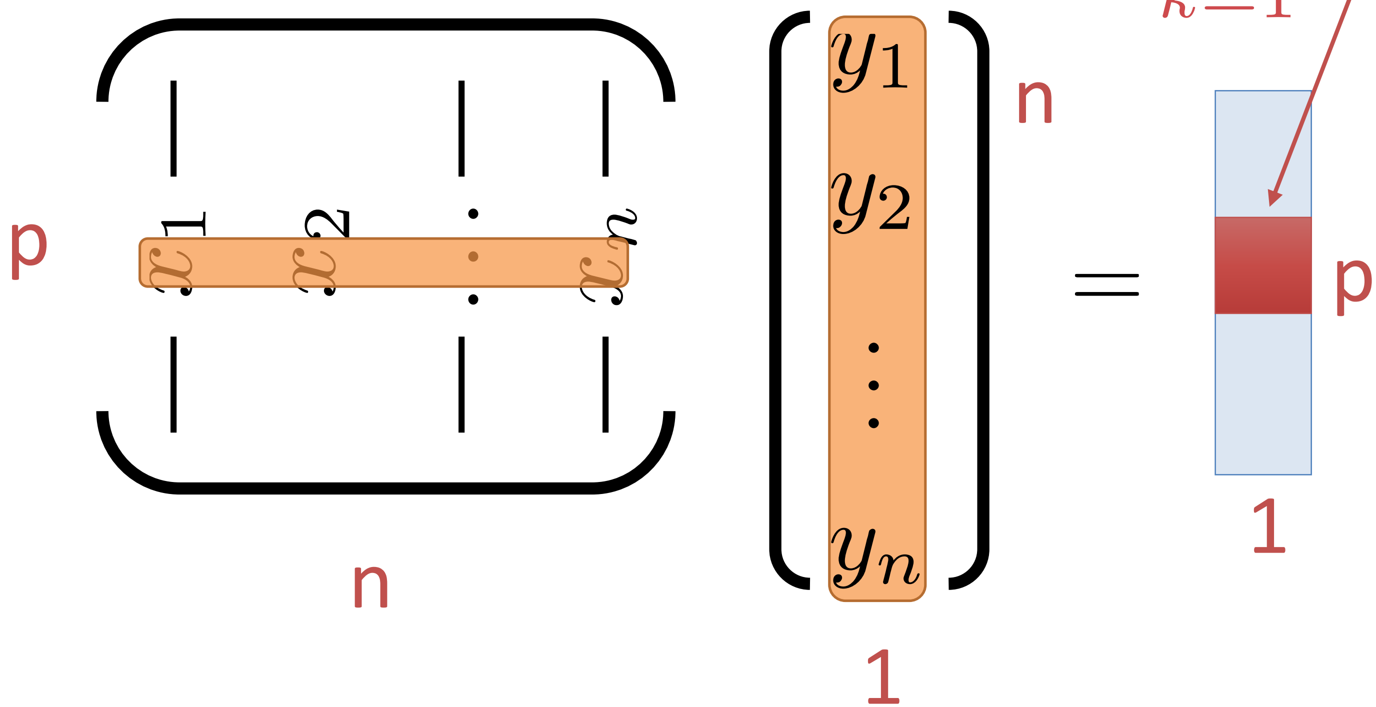
Computing Query 2

$$X^T y =$$



Computing Query 2

$$X^T y = \sum_{k=1}^{n} X_{k,i} y_k$$



Computing Query 2

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17
…	…	…

- Compute the row-wise some:

$$X^T y = \sum_{i=1}^{n} x_i y_i$$

  - **MapFunction(x)**: computes *p* by *1* vector: xy
  - **ReduceFunction**: vector sum

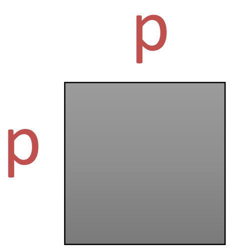
- Pure SQL Expression:

```
SELECT
     sum(x1*y) AS d1, sum(x2*y) AS d2
FROM data
```

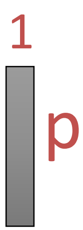
Least Squares Regression using the **Statistical Query Pattern**

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

- In database compute sums:



$$C = X^T X = \sum_{i=1}^{n} x_i x_i^T \qquad O(np^2)$$



$$d = X^T y = \sum_{i=1}^{n} x_i y_i \qquad O(np)$$

- On client compute:

$$\hat{\theta} = C^{-1} d \qquad O(p^3)$$

Least Squares Regression using the Statistical Query Pattern

What if p is large?

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

- In database compute sums:

p p

$$C = X^T X = \sum_{i=1}^{n} x_i x_i^T \qquad O(np^2)$$

1 p

… could be expensive …

$$d = X^T y = \sum_{i=1}^{n} x_i y_i \qquad O(np)$$

- On client compute:

$$\hat{\theta} = C^{-1} d \qquad O(p^3)$$

- Rather than directly solving:

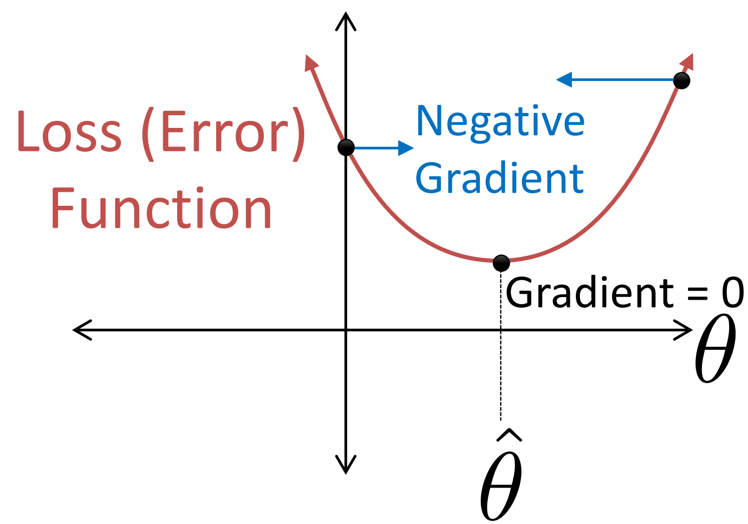
$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \theta^T x_i)$$

- Instead we compute the gradient of the loss:

$$G(\theta; X, y) = \nabla_\theta \frac{1}{n} \sum_{i=1}^{n} L(y_i, \theta^T x_i) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L(y_i, \theta^T x_i)$$

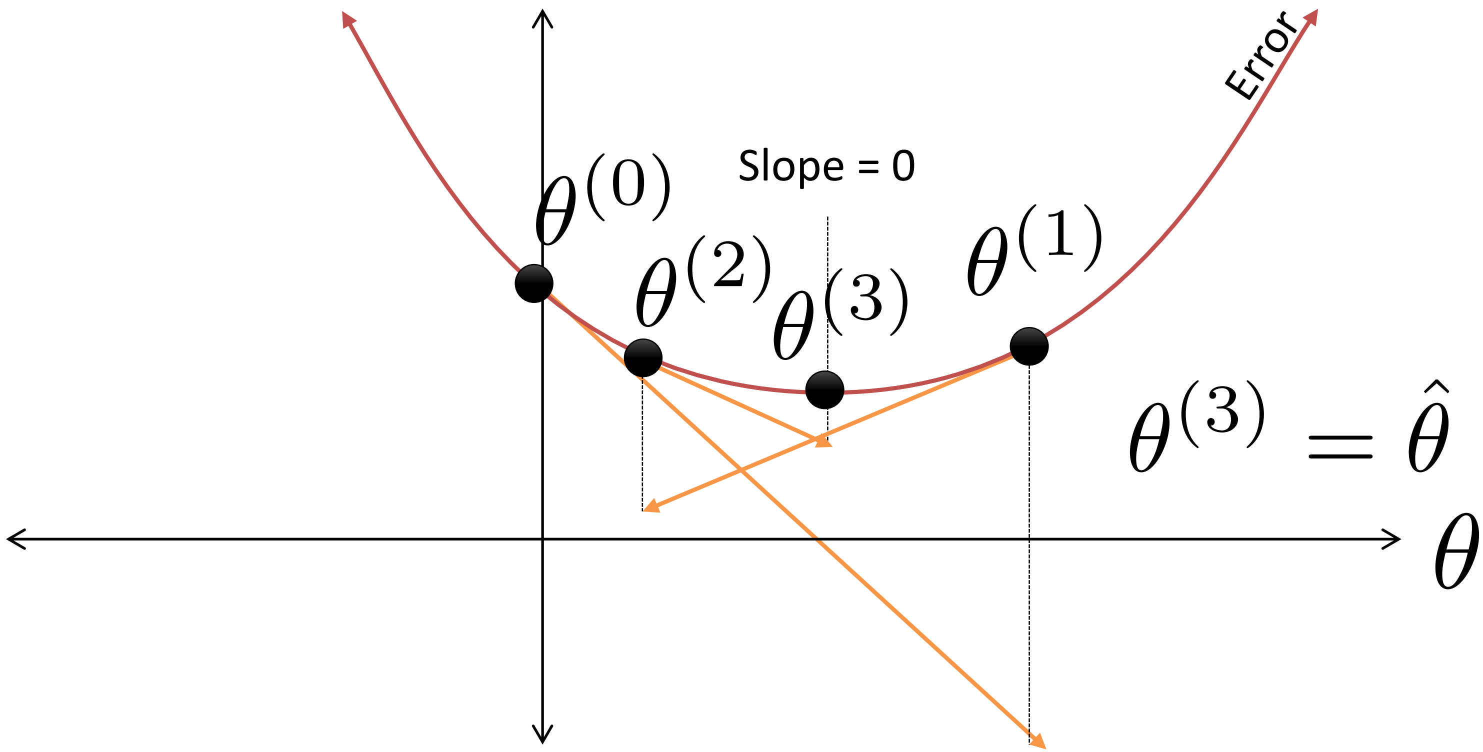
$$= \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i) x_i$$

- **Big Idea:** Negative gradient points in the direction of steepest descent



Gradient Descent Algorithm

```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
   θ^(t+1) ← θ^(t) - Stepsize^(t) * G(θ; X,Y)
   t ← t + 1
```



Gradient Descent Algorithm

```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
   θ^(t+1) ← θ^(t) – Stepsize^(t) * G(θ; X,y)
   t ← t + 1
```

➢ Does this fit the statistical query pattern

- Yes! Only dependence on data is:

Complexity?

Size? 1 p

$$G(\theta; X, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)x_i \qquad O(np)$$

- Can we go even faster?
  - **Stochastic Gradient Descent (SGD)**: Approximate the gradient by sampling data (typically several hundred records per query).

Stochastic Gradient Descent

- Update the parameters for each training case in turn, according to its own gradients

```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
    θ^(t+1) ← θ^(t) – Stepsize^(t) * G(θ; X,y)
    t ← t + 1
```

$$G(\theta; X, y) = (y_i - \theta^T x_i) x_i$$

$$\text{Stepsize}^{(t)} = \frac{1}{t+1}$$

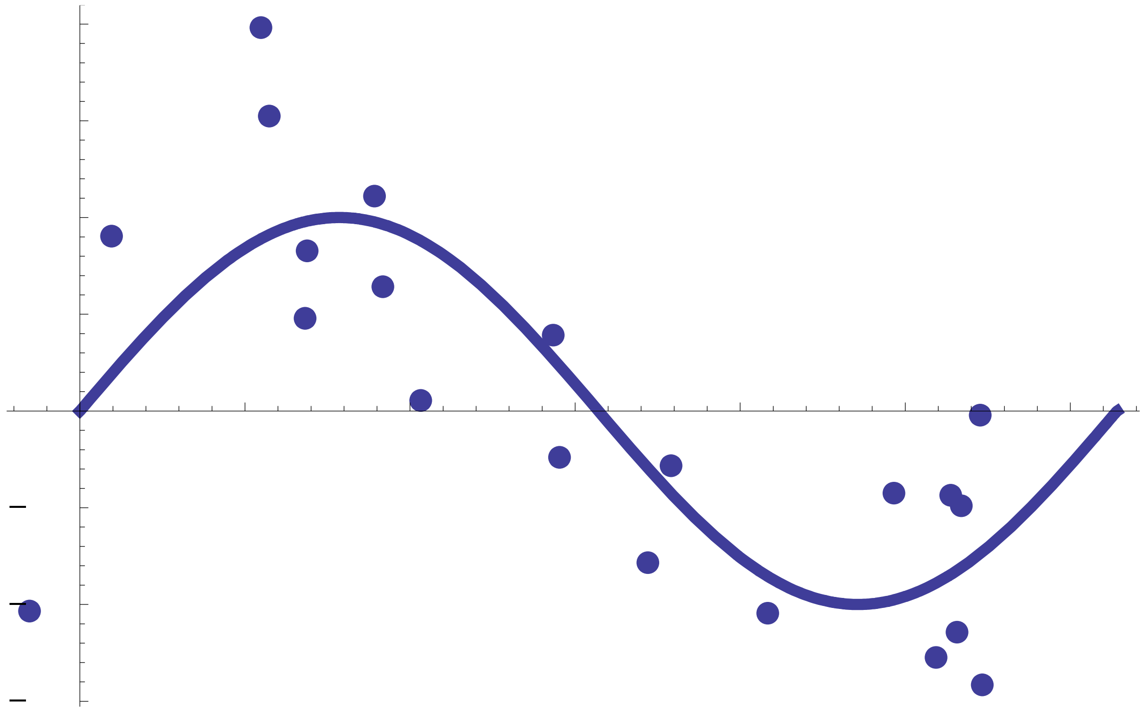
Complexity?

$O(p)$

Advanced Topics in Linear Regression

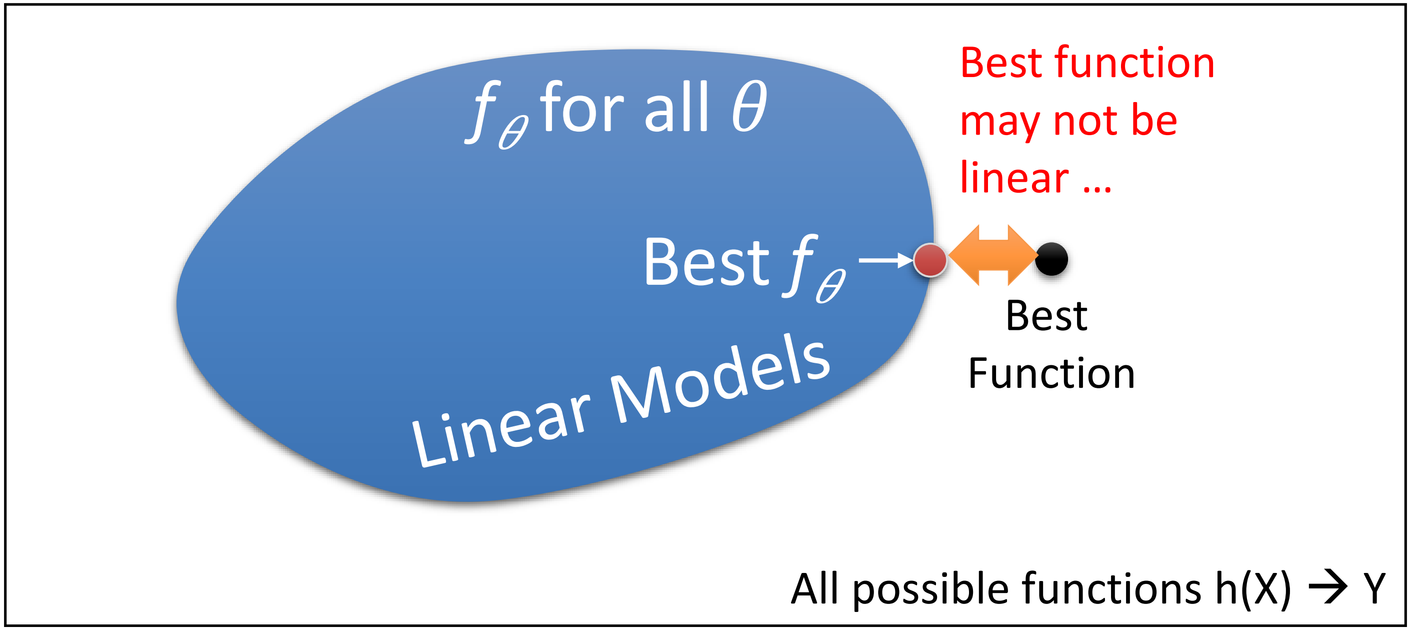
Fitting Non-linear Data

- What if Y has a non-linear response?

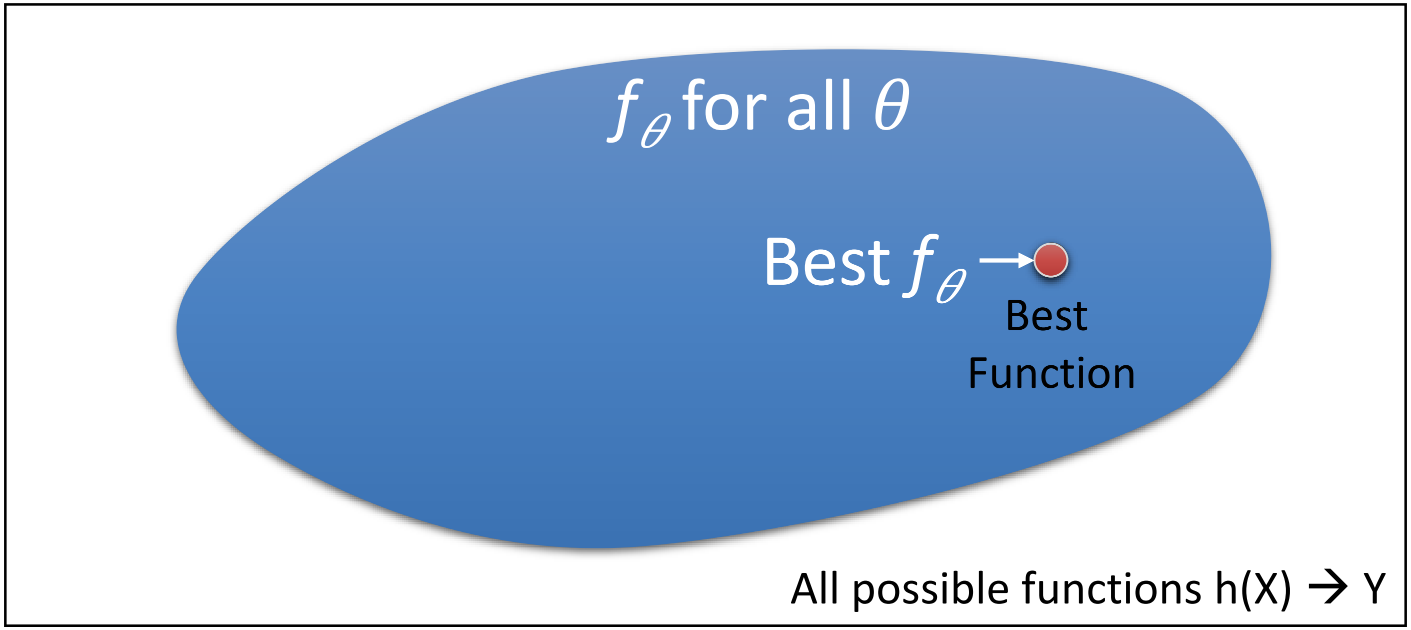


- Can we still use a linear model?

Finding the Best Parameters



Finding the Best Parameters



Feature Engineering

➢ By applying non-linear transformation ϕ:

$$\phi : \mathbb{R}^p \rightarrow \mathbb{R}^k$$

- Example:

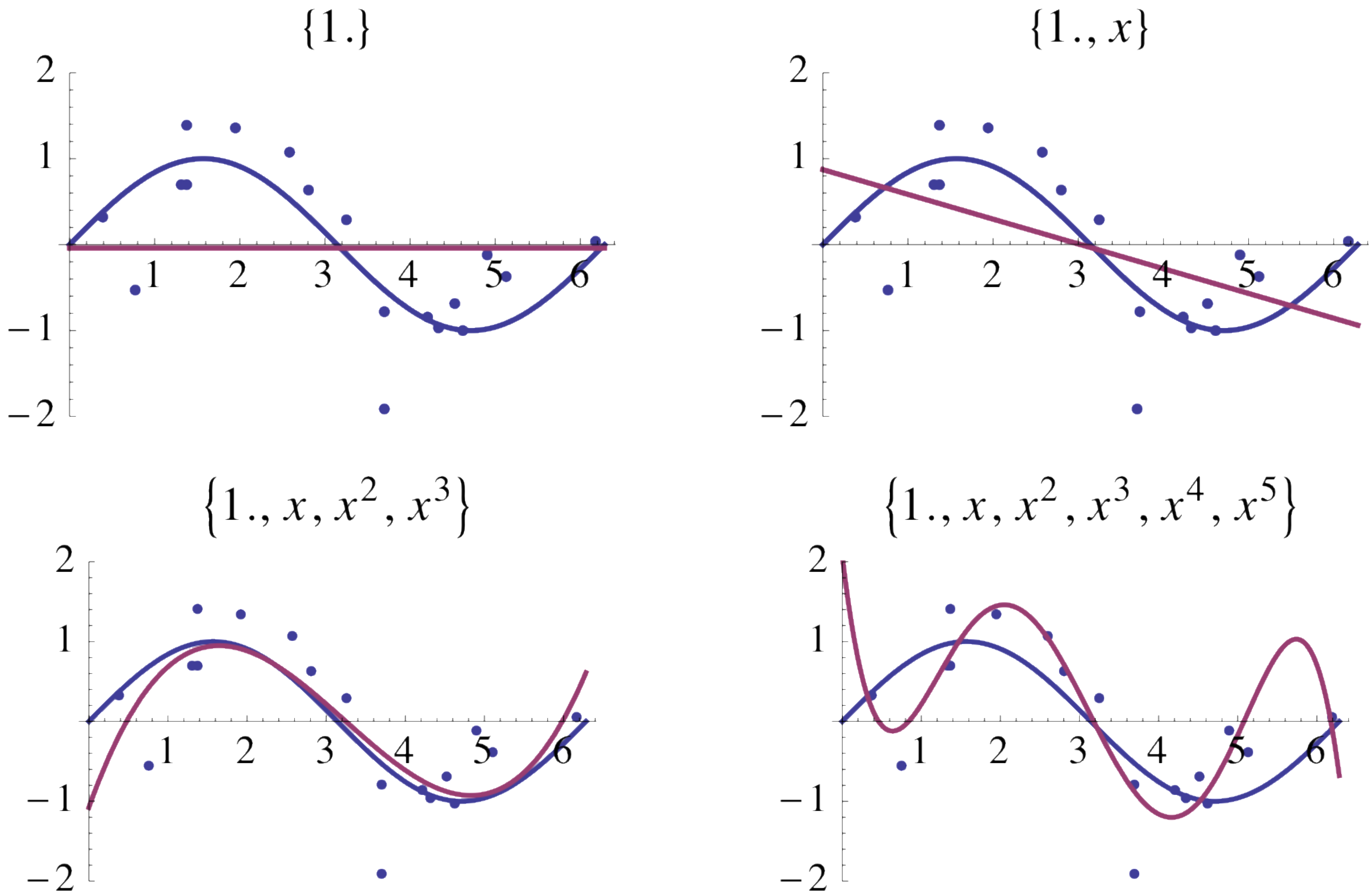
$$\phi(x) = \{1, x, x^2, \ldots, x^k\}$$

x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17

$$\xrightarrow{\phi}$$

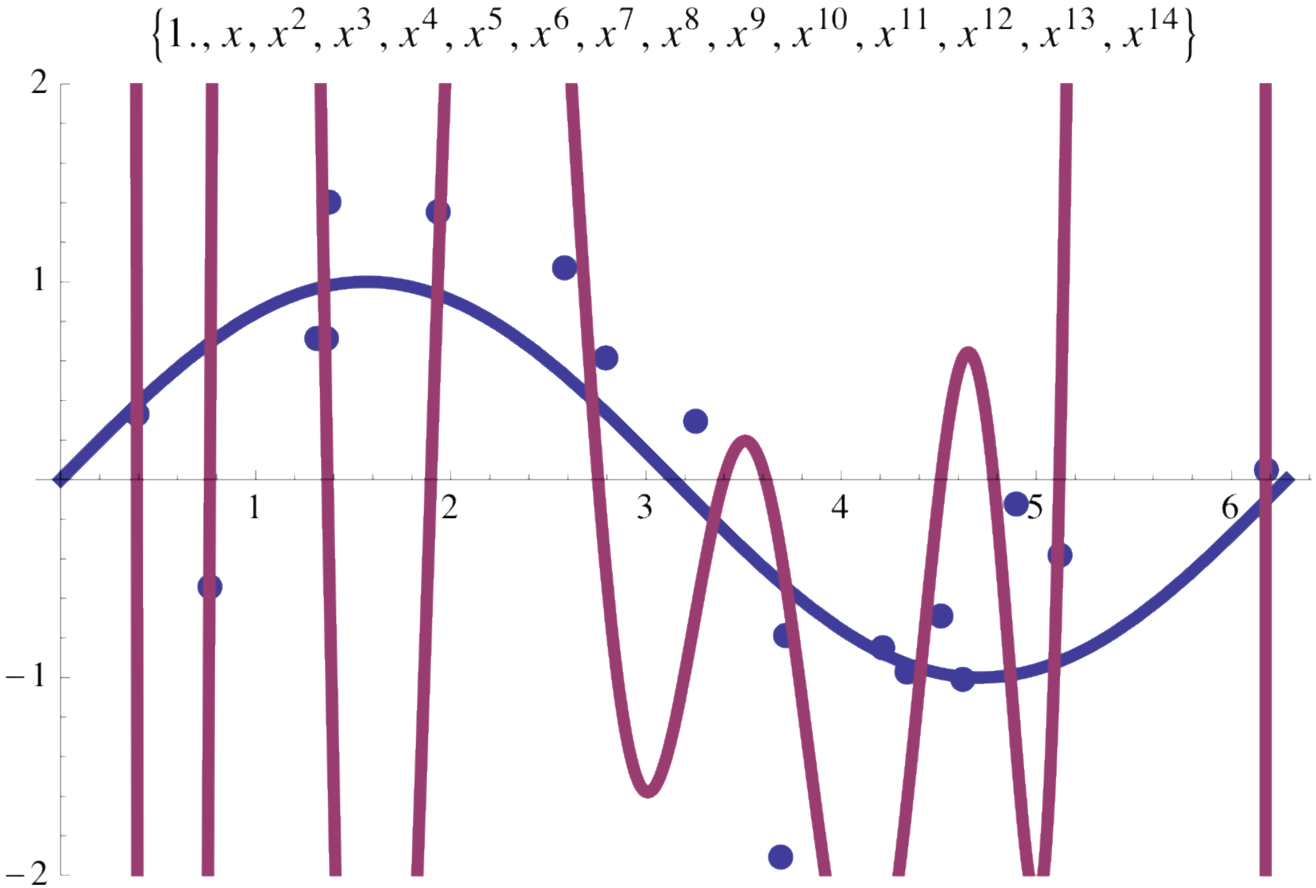
x_1	x_2	$x_1 * x_1$	$x_2 * x_2$	$x_1 * x_2$	y
1.1	2.7	1.21	7.29	2.97	3.6
4.2	3.2	17.64	10.24	13.44	7.5
9.8	9.2	90.04	84.64	90.16	17

Under-fitting



Over-fitting

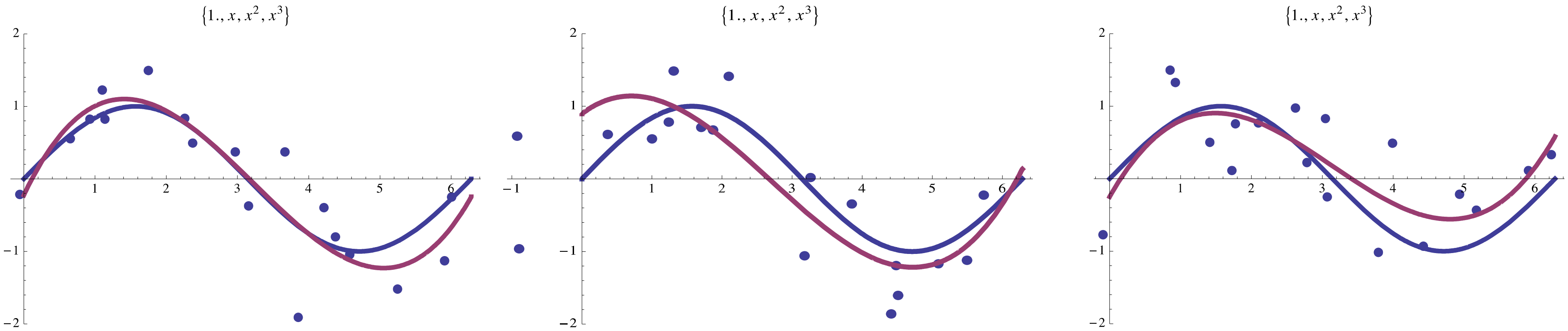
Really Over-fitting!



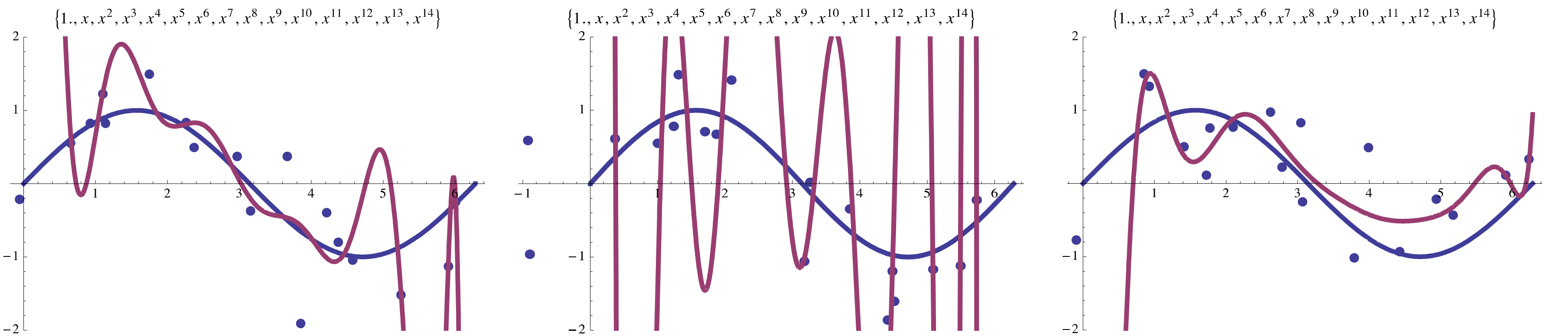
- ➢Errors on training data are small
- ➢But errors on new points are likely to be large

What if I train on different data?

Simple Model → Low Variability



Complex Model → High Variability



Bias-Variance Tradeoff

- ➢ So far we have minimized the **training error**: the error on the training data.
  - low training error does not guarantee good expected performance (due to over-fitting)
- ➢ We would like to reason about the **test error**

<u>Theorem:</u>

Test Error = Noise + Bias^2 + Variance

Noisy data has inherent error (measurement error)

Error due to models **inability to fit** the data. **(Under Fitting)**

Error due to inability to estimate model parameters. **(Over-fitting)**

Bias Variance Plot

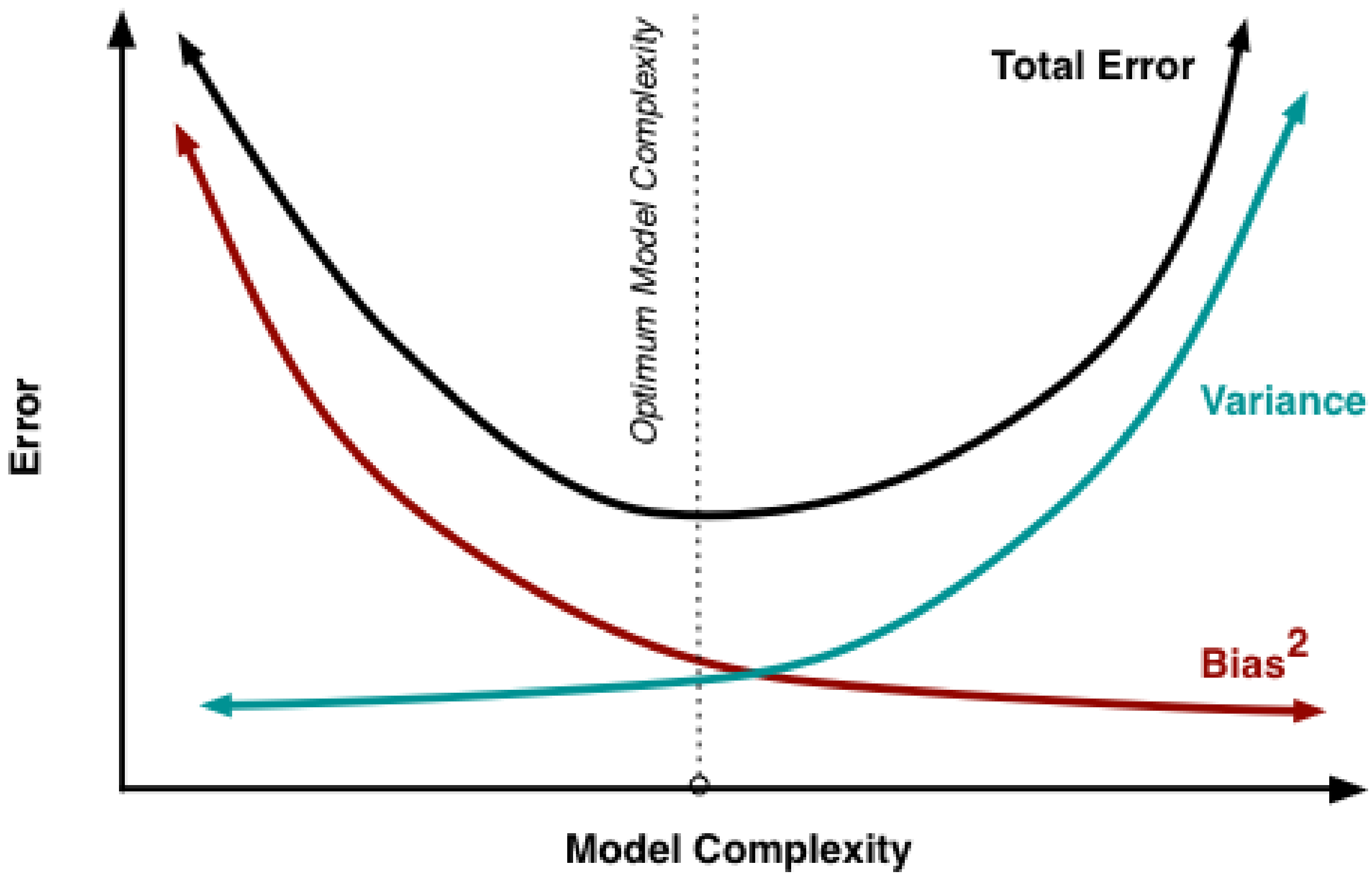


Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

Regularization to Reduce Over-fitting

- High dimensional models tend to over-fit
  - How could we “favor” lower dimensional models?
- **Solution Intuition:**
  - Too many features → over-fitting

$$f(x) = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_d x_d$$

  - Force many of the $\theta_i \approx 0$ (e.g., i > 2) (“effectively fewer features”)

$$\begin{aligned} f(x) &= \theta_1 x_1 + \theta_2 x_2 + 0 x_3 + \ldots + 0 x_d \\ &= \theta_1 x_1 + \theta_2 x_2 \end{aligned}$$

  - Keeping weights close to zero **reduces variance**

Regularization to Reduce Over-fitting

- We can add a regularization term:

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \overbrace{(y_i - \theta^T x_i)^2}^{\text{Squared Loss}} + \underbrace{\lambda}_{\text{Regularization Parameter}} \overbrace{R(\theta)}^{\text{Regularization Function}}$$

- Common of Regularization Functions:

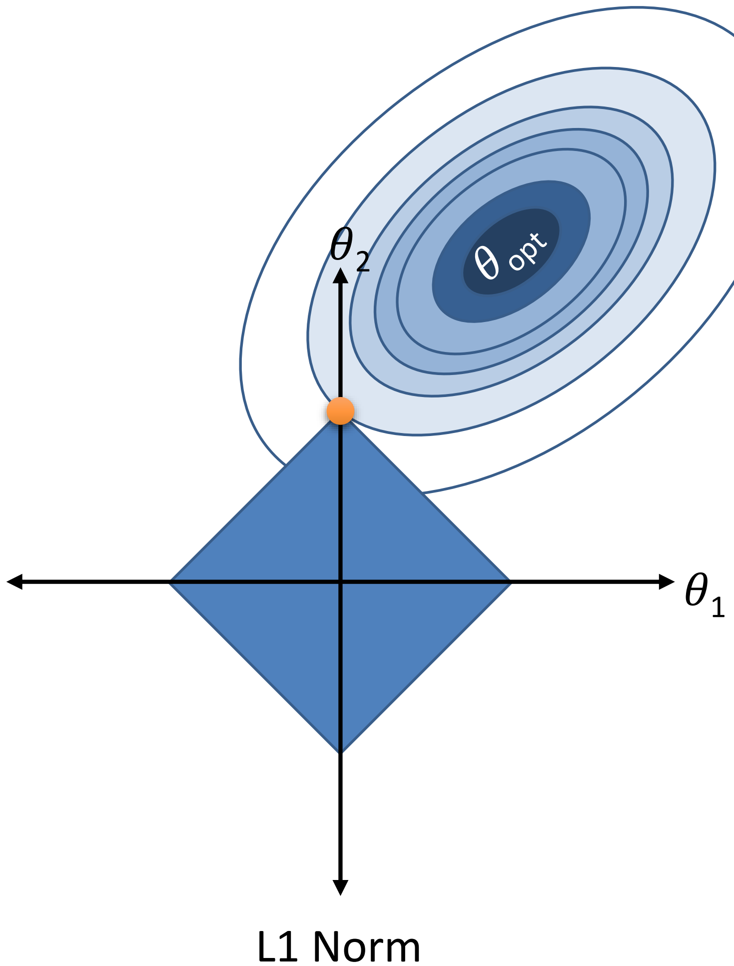
Ridge (L2-Reg) Regression $$R_{\text{Ridge}}(\theta) = \sum_{i=1}^{d} \theta_i^2$$

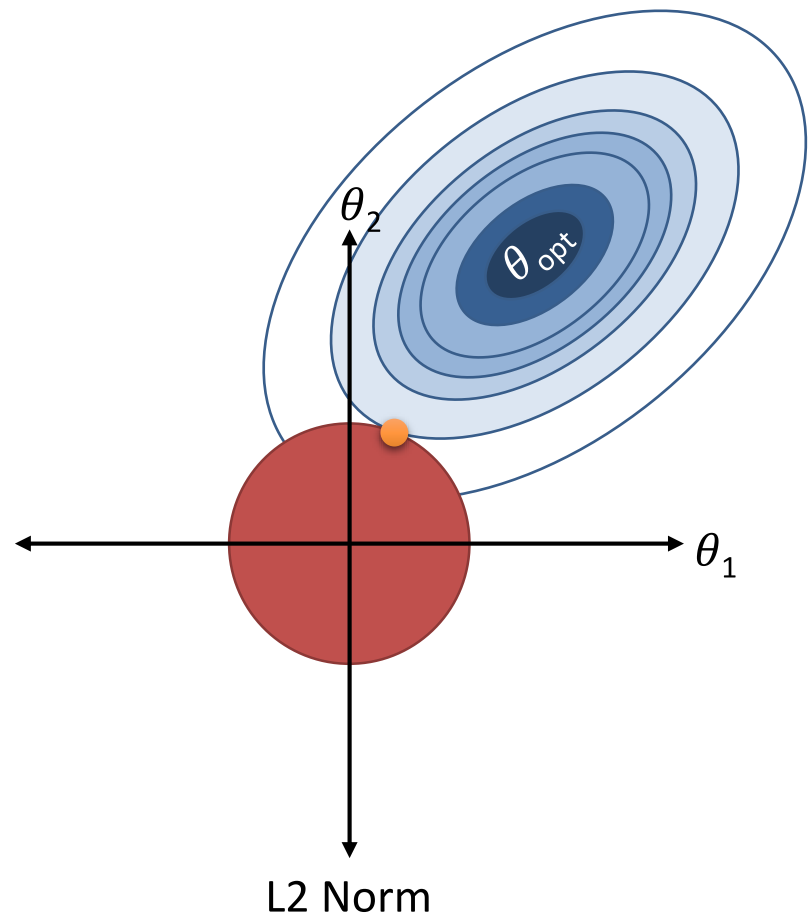
Lasso (L1-Reg) $$R_{\text{Lasso}}(\theta) = \sum_{i=1}^{d} |\theta_i|$$

  - Encourage small parameter values

- The parameter λ determines amount of reg.
  - Larger → more reg. → lower variance → higher bias

Regularization and Norm Balls





Regularization to Reduce Over-fitting

- We can add a regularization term:

$$\hat{\theta} = \arg\min_{\theta\in\mathbb{R}^p} \frac{1}{n}\sum_{i=1}^{n} \overbrace{(y_i - \theta^T x_i)^2}^{\text{Squared Loss}} + \underbrace{\lambda}_{\text{Regularization Parameter}} \overbrace{R(\theta)}^{\text{Regularization Function}}$$

- Solving the regularized problem:
  - Closed form solution for Ridge regression (L2):

$$\hat{\theta}_{\text{Ridge}} = (X^T X + \lambda I)^{-1} X^T Y$$

  - Iterative methods for Lasso (L1):
    - Stochastic gradient …
- How do we choose λ?

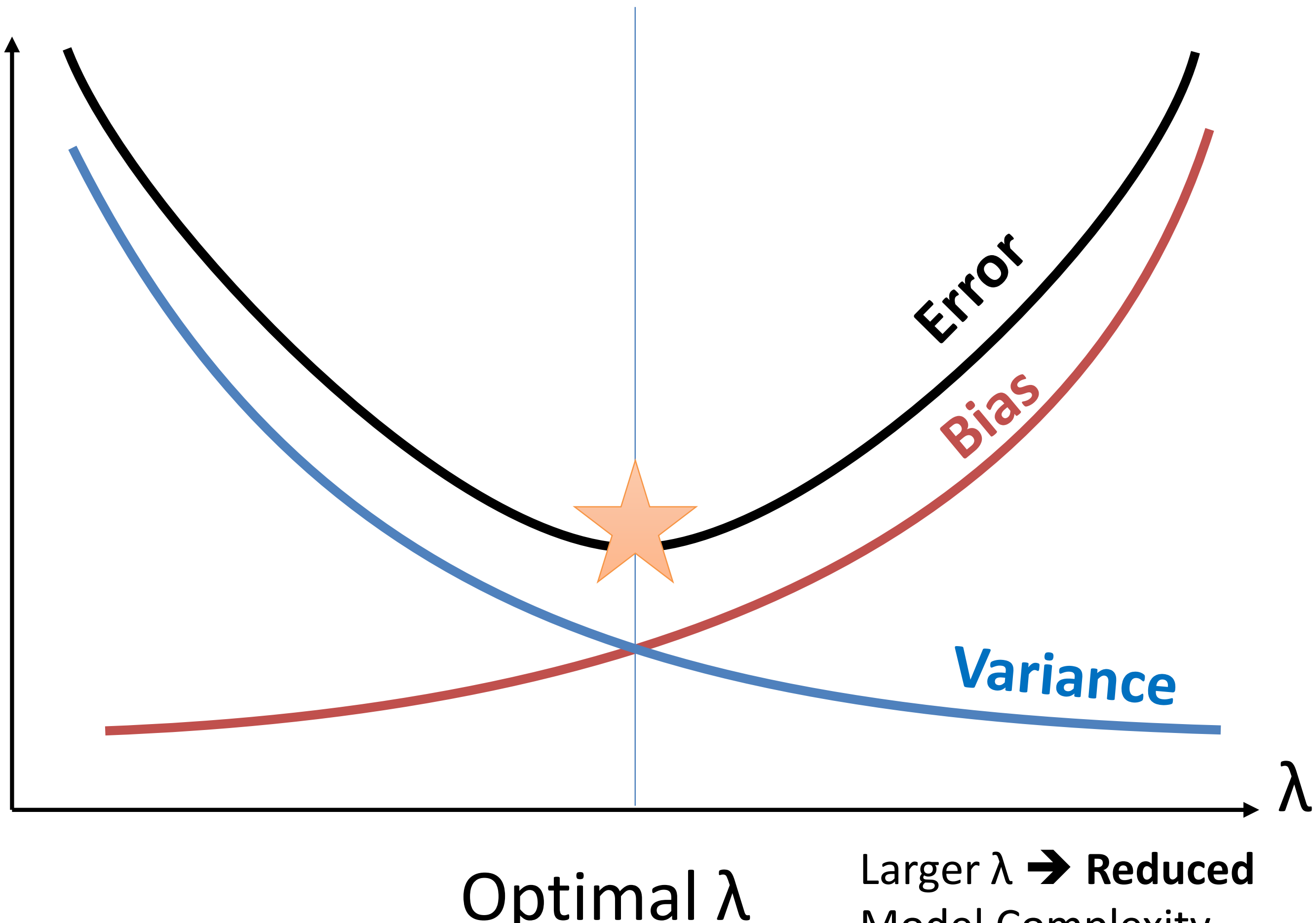
Picking The Regularization Parameter λ

- **Proposal:** Minimize **training** error

$$\arg\min_{\theta \in \mathbb{R}^p, \boxed{\lambda \geq 0}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta^T x_i)^2 + \lambda R(\theta)$$

  - Trivial solution ➔ λ = 0
- Intuition we want to minimize **test** error
  - **Test error**: error on unseen data
- **2nd Proposal:** Split training data into training and evaluation sets
  - For a range of λ values compute optimal θ_λ using only the reduced training set
  - Evaluate θ_λ on the separate evaluation set and select the λ with the lowest error

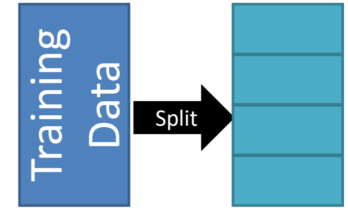
Bias Variance Plot



Larger λ ➔ **Reduced** Model Complexity

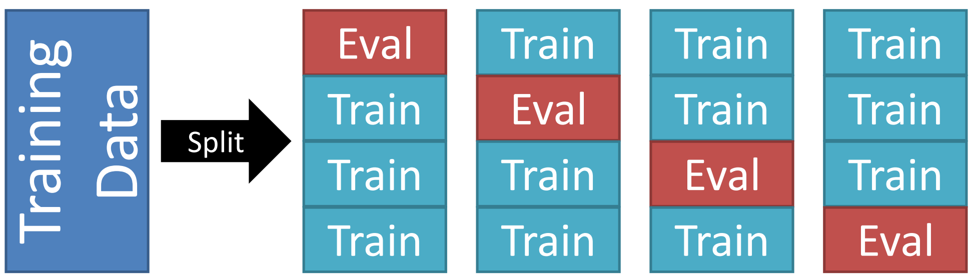
Image from http://scott.fortmann-roe.com/docs/BiasVariance.html

K-Fold Cross Validation



- Split training data into K-equally sized parts
  - In practice K is relatively small (e.g., 5)
- For each part train on the other k-1 parts and compute the error on that part:

K = 4

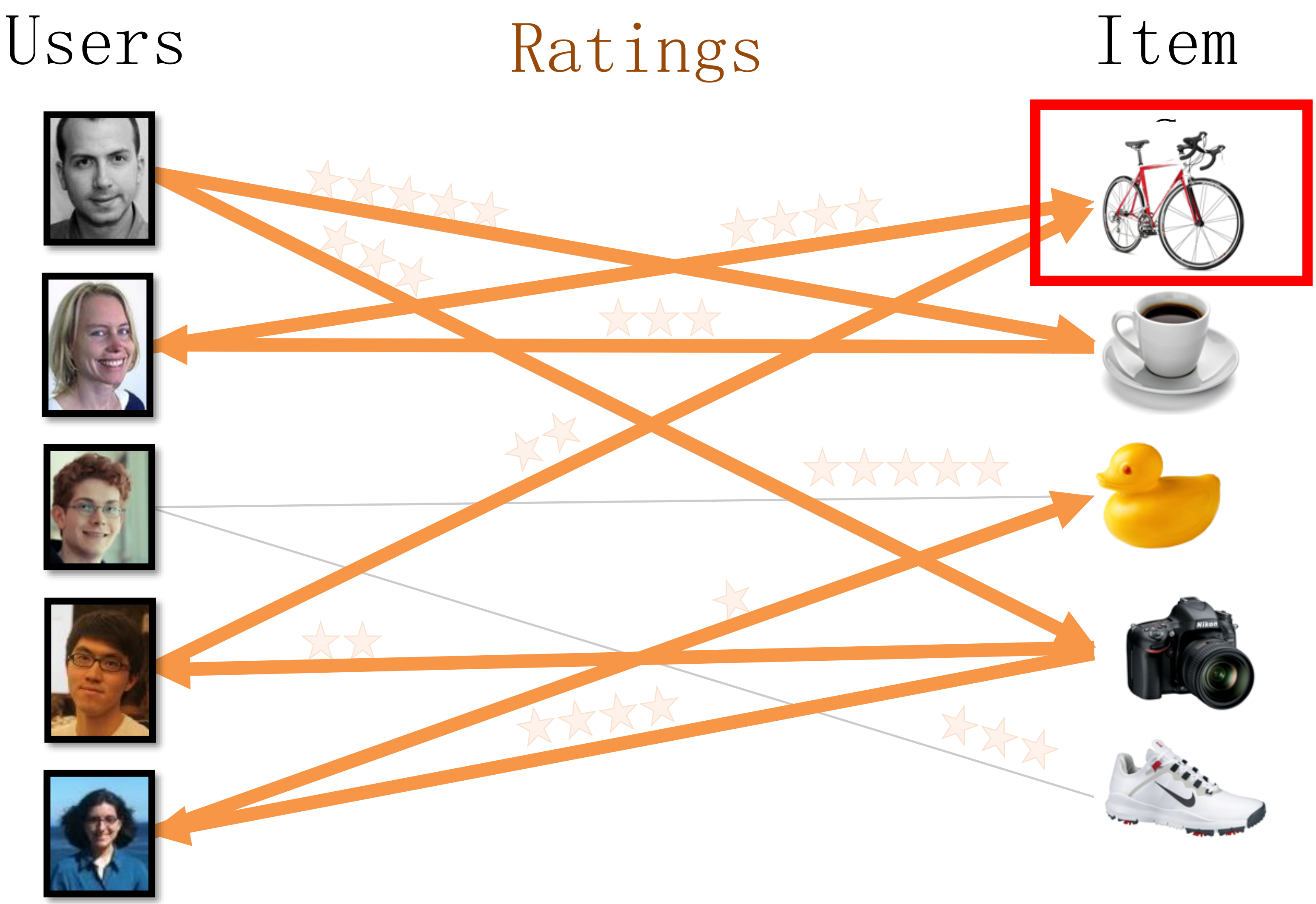


Error is averaged over all 4 steps

- Compute the average test error over held out parts
- Select reg. param. that minimizes average test error

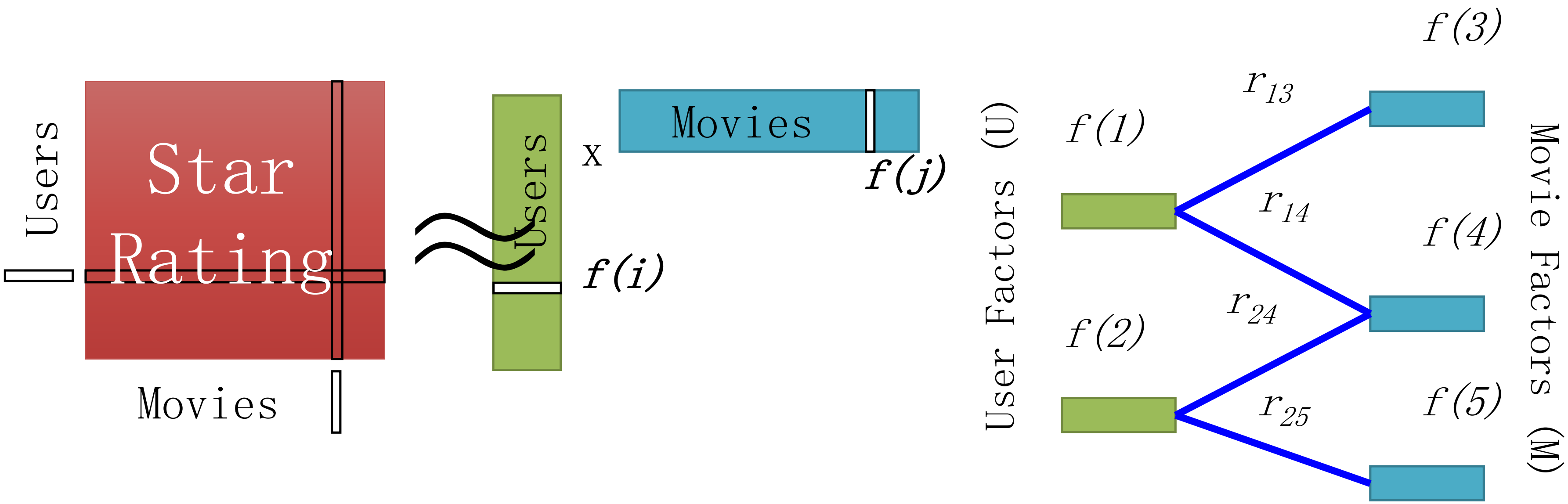
Using Regression for Content Recommendation

Recommending Products



Recommending Products

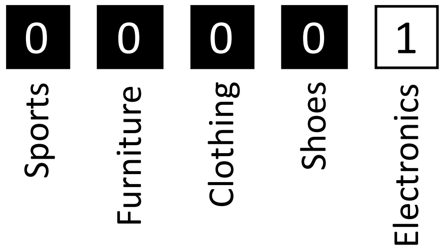
Low-Rank Matrix Factorization:

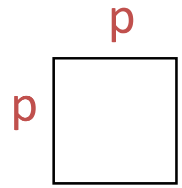


Iterate:

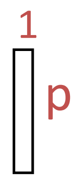
$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

Summary of Regression





$$C = X^T X = \sum_{i=1}^{n} x_i x_i^T$$



$$d = X^T y = \sum_{i=1}^{n} x_i y_i$$