CS150A Database

Lu Sun
School of Information Science and Technology
ShanghaiTech University
Dec. 20, 2022

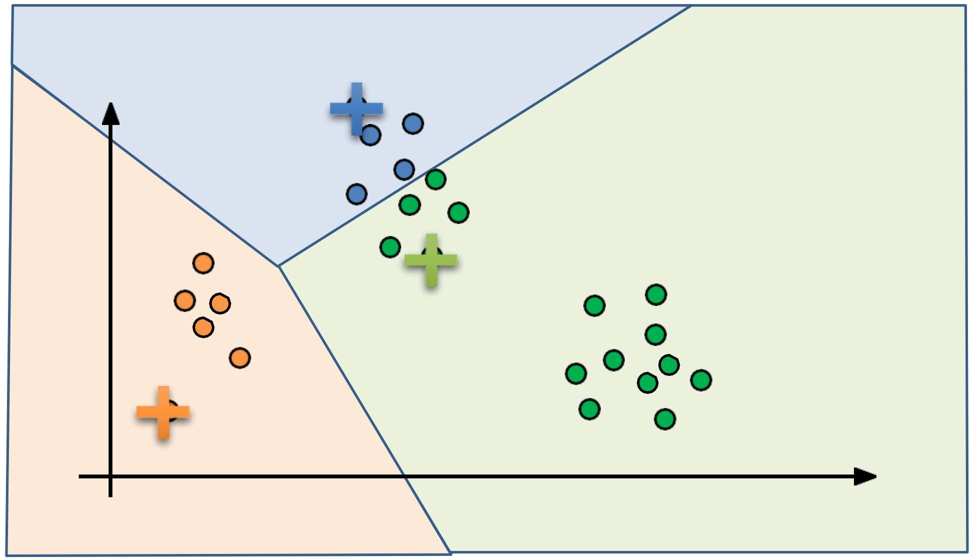
Today:
- Analytics and ML in Data Systems:
  - Part 3
  - Linear Regression

Readings:
- Database Management Systems (DBMS), Chapters 23&24

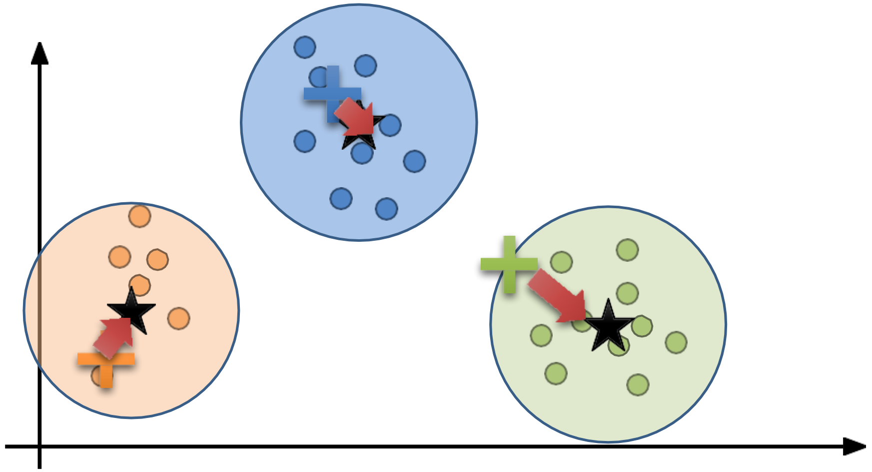
Acknowledgement: Joseph E. Gonzalez@UCBerkeley's scourse notes

K-Means Clustering

Compute Assignments

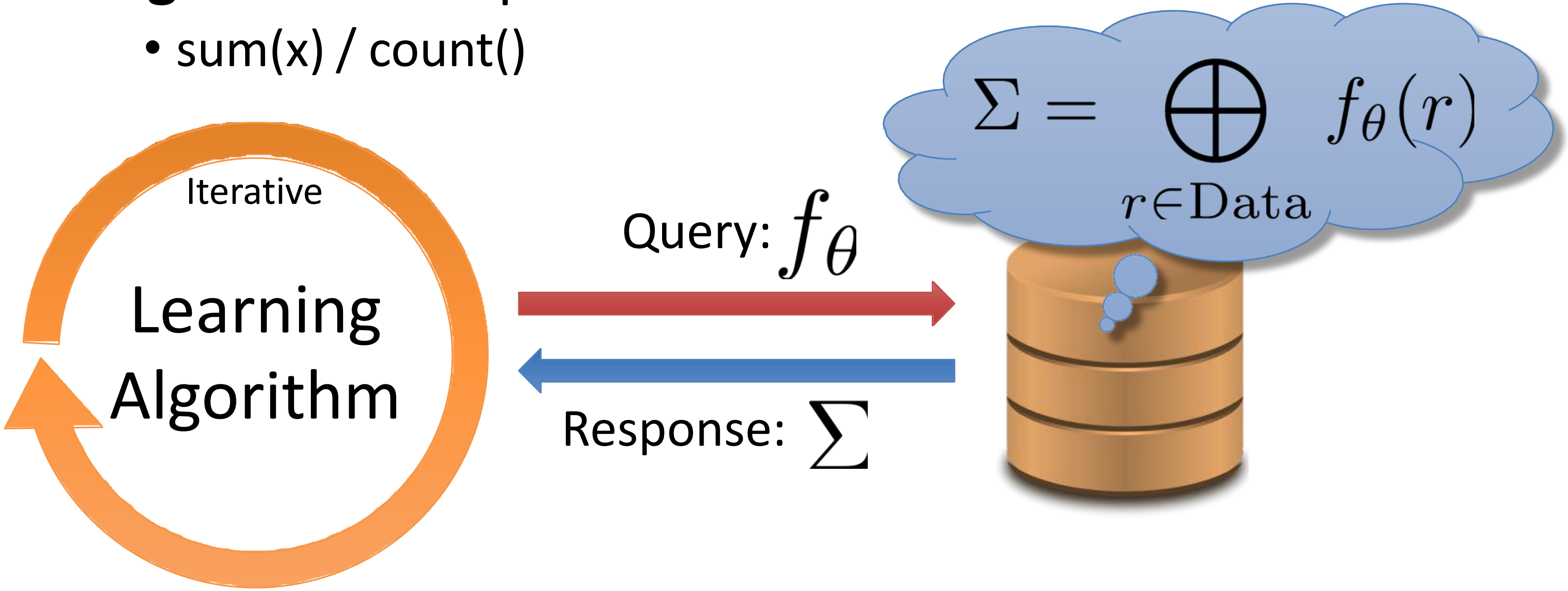


Update Centers



Statistical Query Pattern
Common Machine Learning Pattern

- **K-Means Query**: for each old centers compute the sum of the nearest points
- **Response:** sums and counts of points
- **Algorithm:** compute new centers
  - sum(x) / count()



D. Caragea et al., *A Framework for Learning from Distributed Data Using Sufficient Statistics and Its Application to Learning Decision Trees*. Int. J. Hybrid

Res-A: weighted reservoir sampling

- **Goal:** *Sample **k** records from a stream where record **i** is included in the sample with probability proportional to* w_i
- **Algorithm:**
  - For each record ***i*** draw a uniform random number:

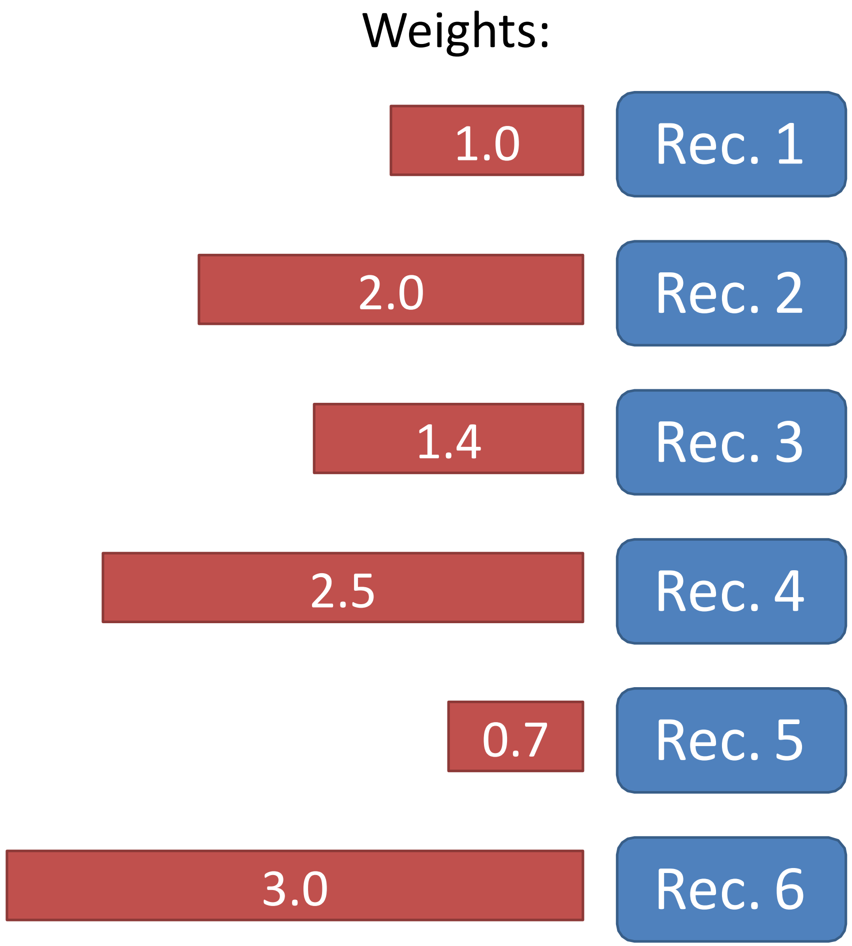
$$u_i \sim \textbf{Unif}(0, 1)$$

  - Select the top-k records ordered by: u_i^{1/w_i}

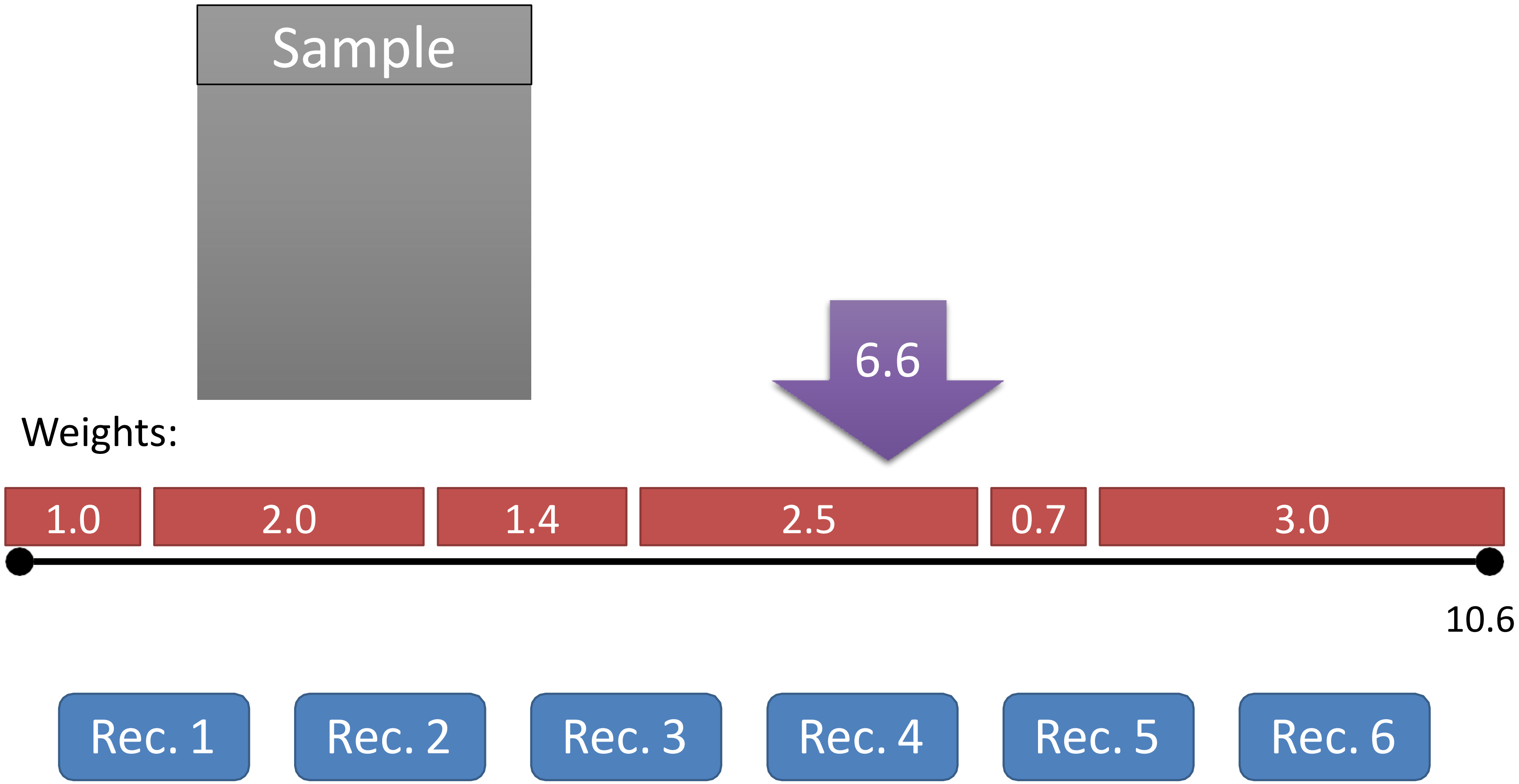
- **Common ML Pattern?**
  - **Query Function:** *[pow(rand(), 1 / record.w), record]*
  - **Agg. Function:** *top-k heap*

Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Inf. Process. Lett.* 97, 5 (March 2006), 181-185

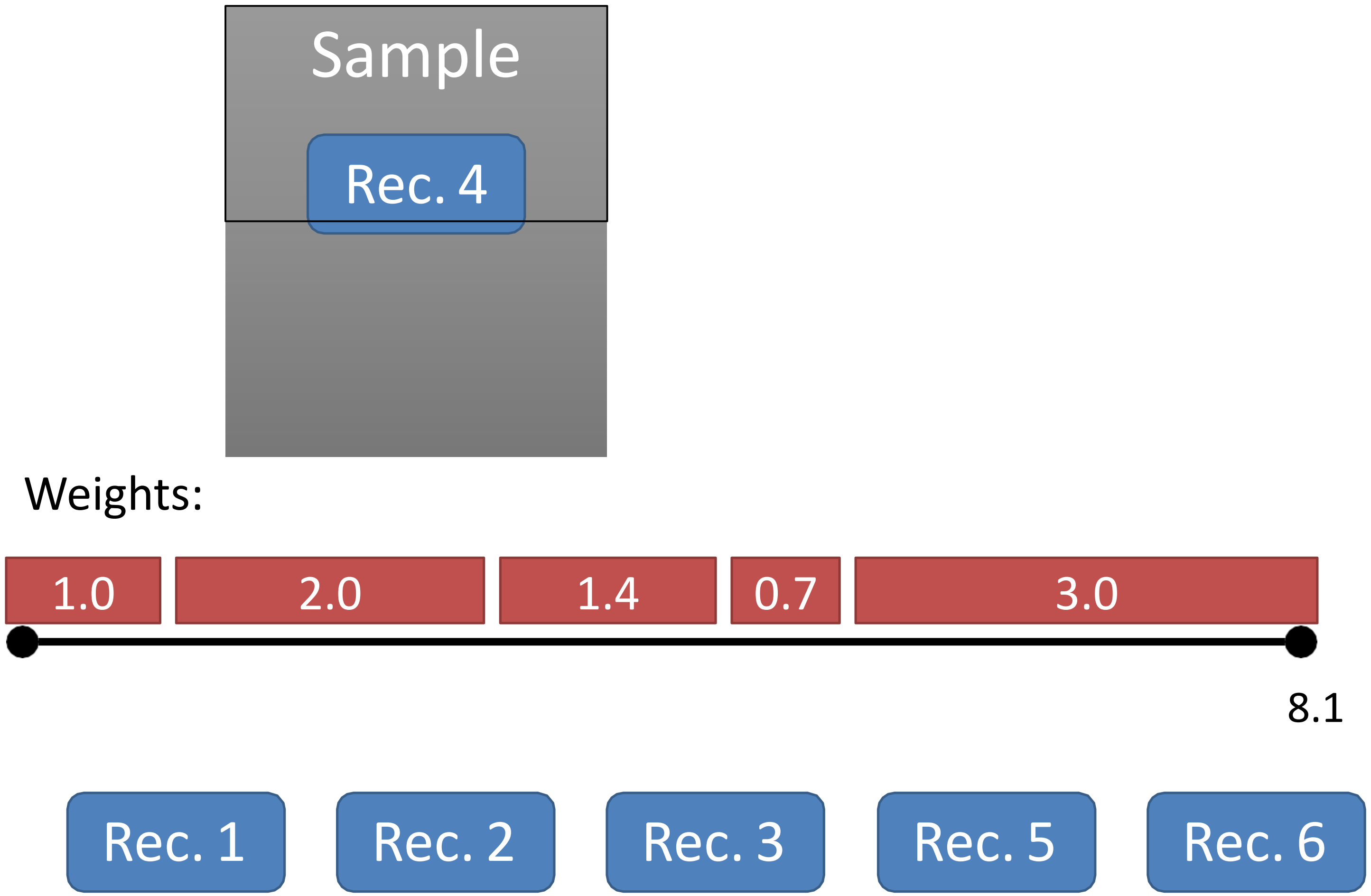
Illustrating Basic Weighted Sampling



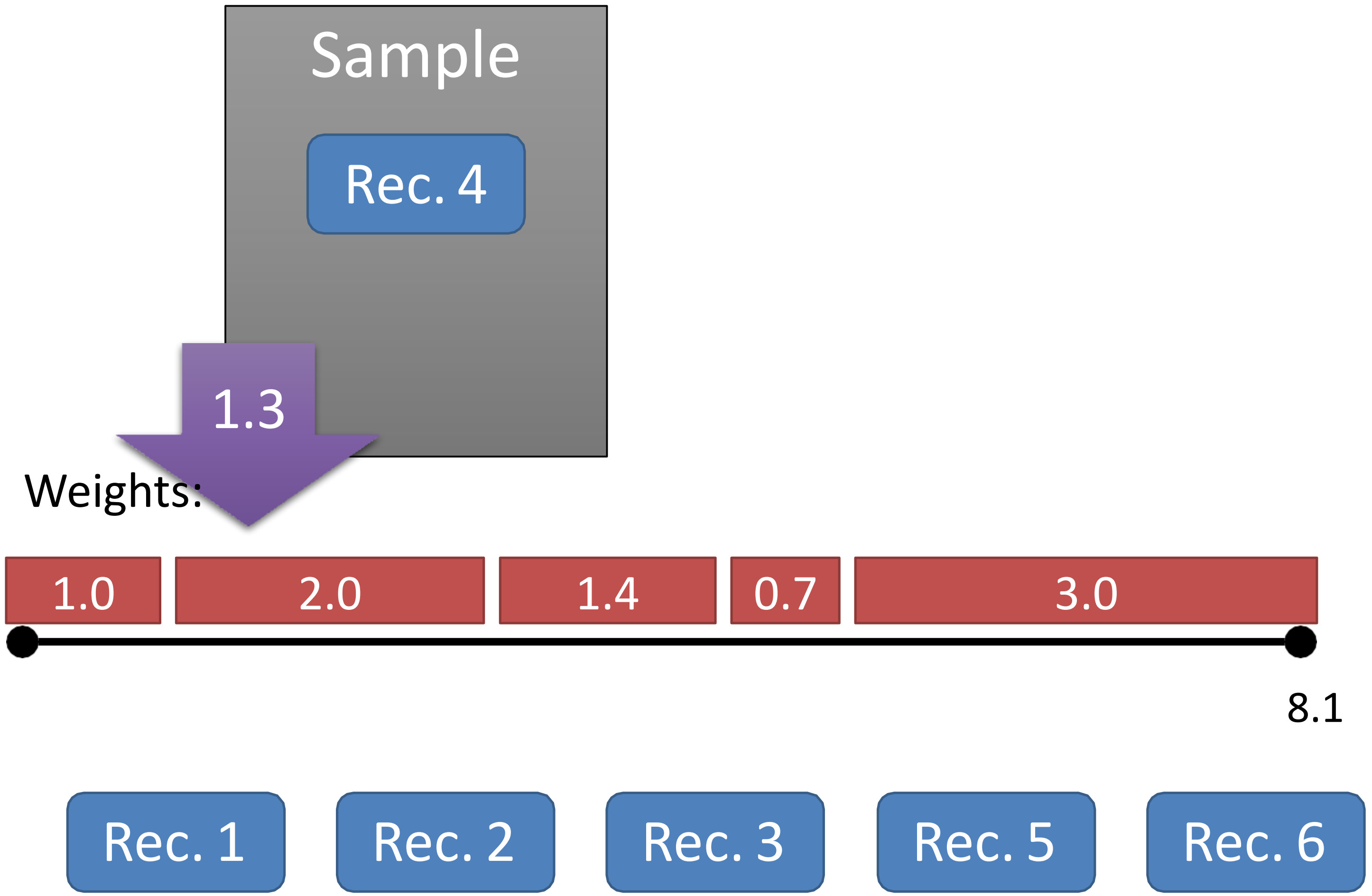
Illustrating Basic Weighted Sampling



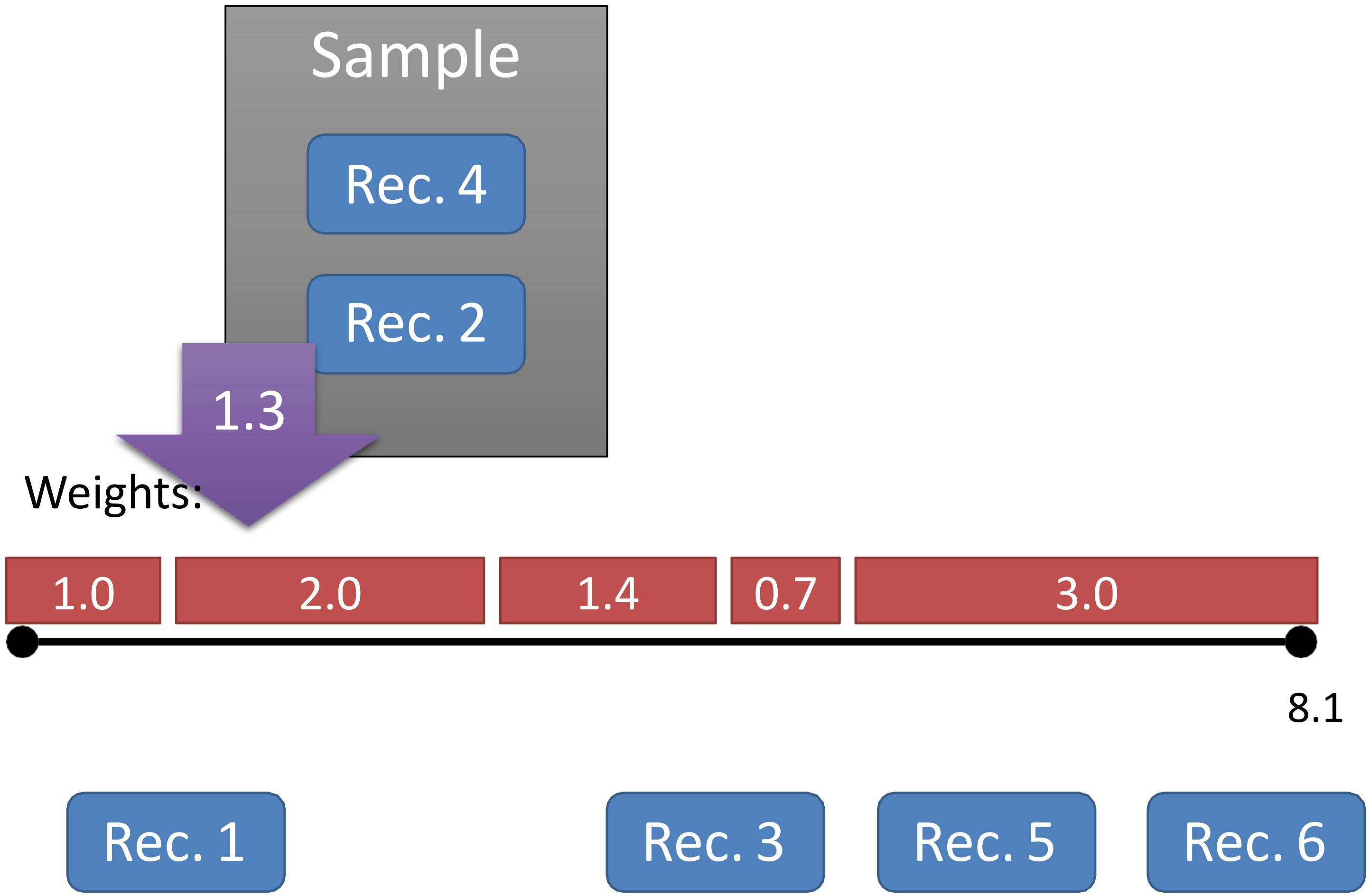
Illustrating Basic Weighted Sampling



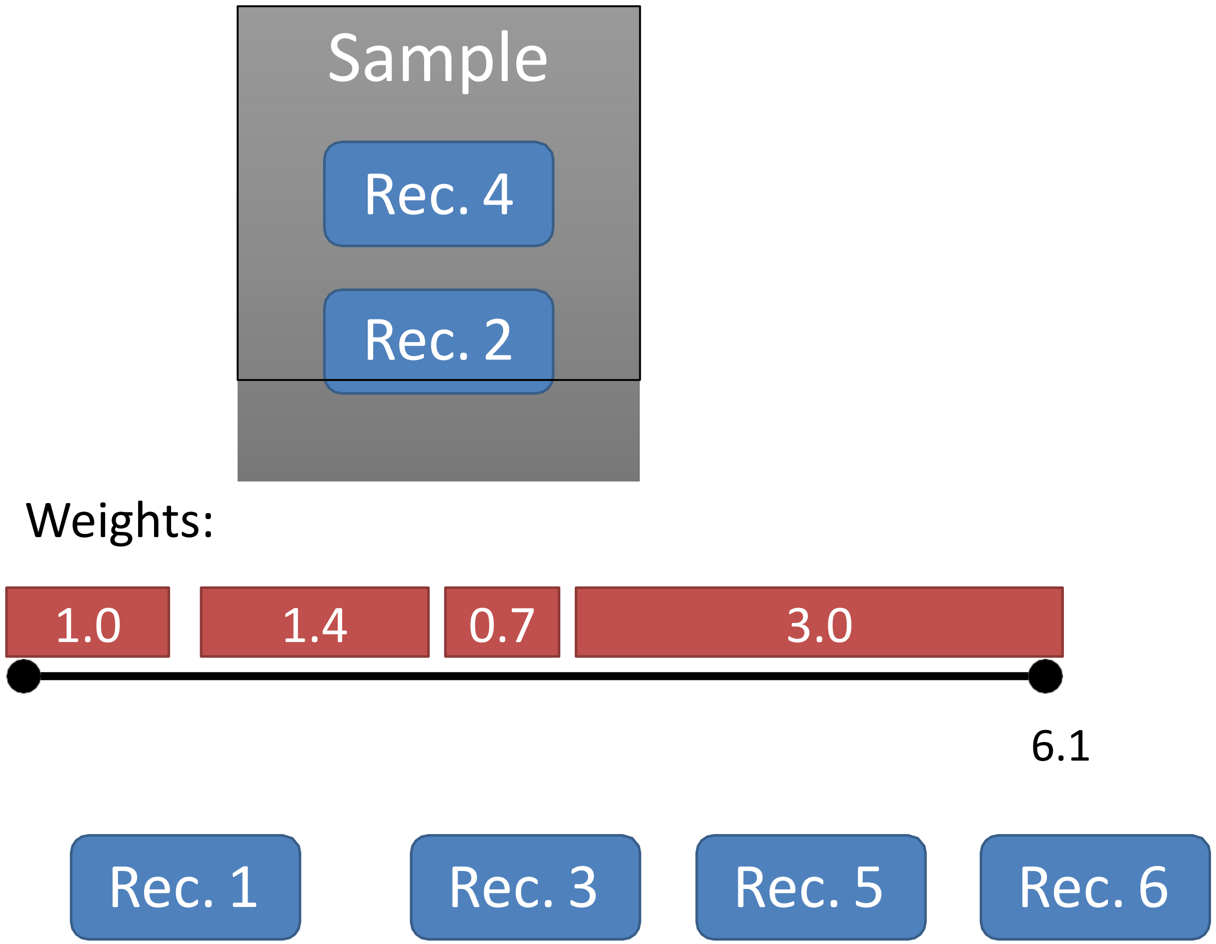
Illustrating Basic Weighted Sampling



Illustrating Basic Weighted Sampling



Illustrating Basic Weighted Sampling

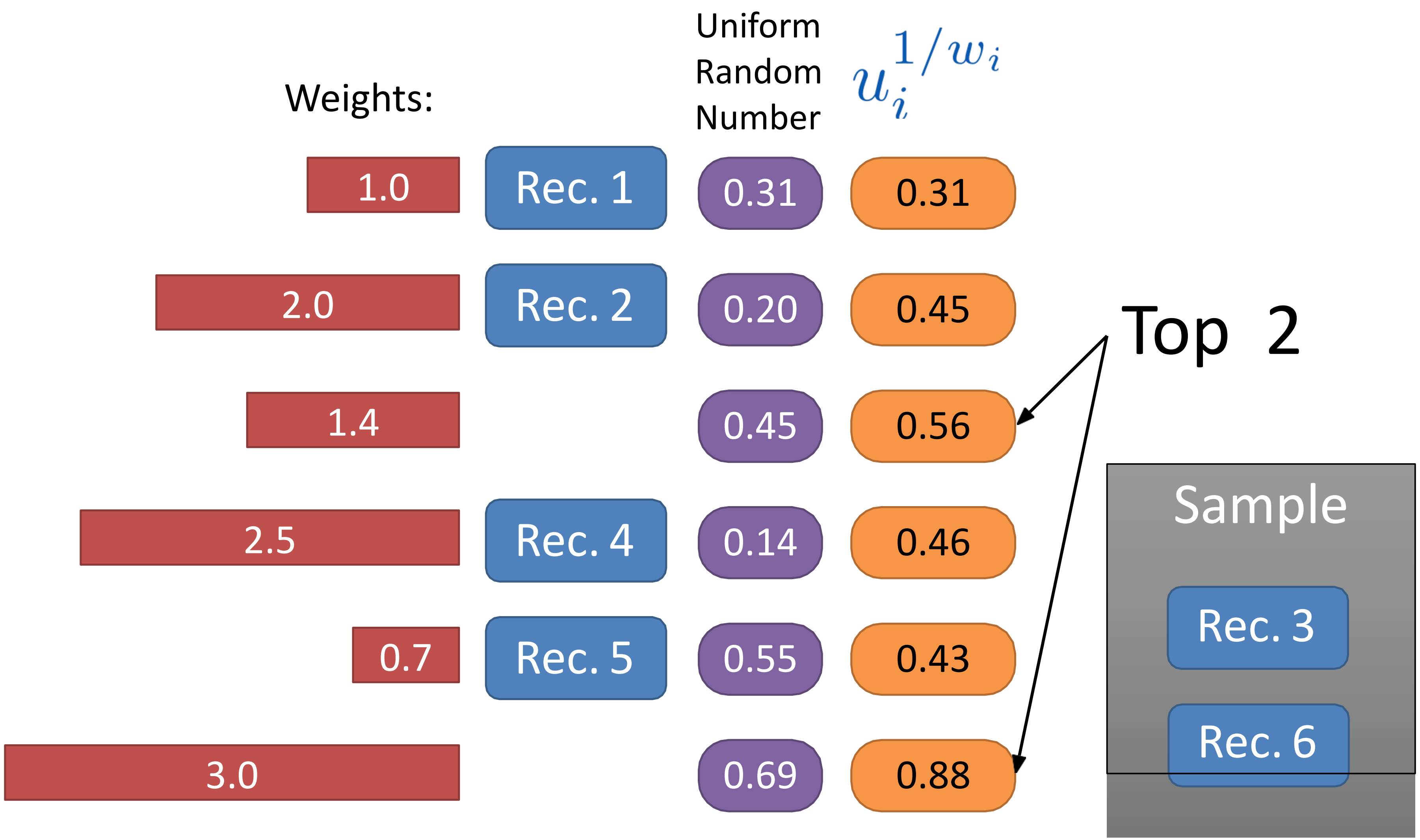


Illustrating Res-A Algorithm

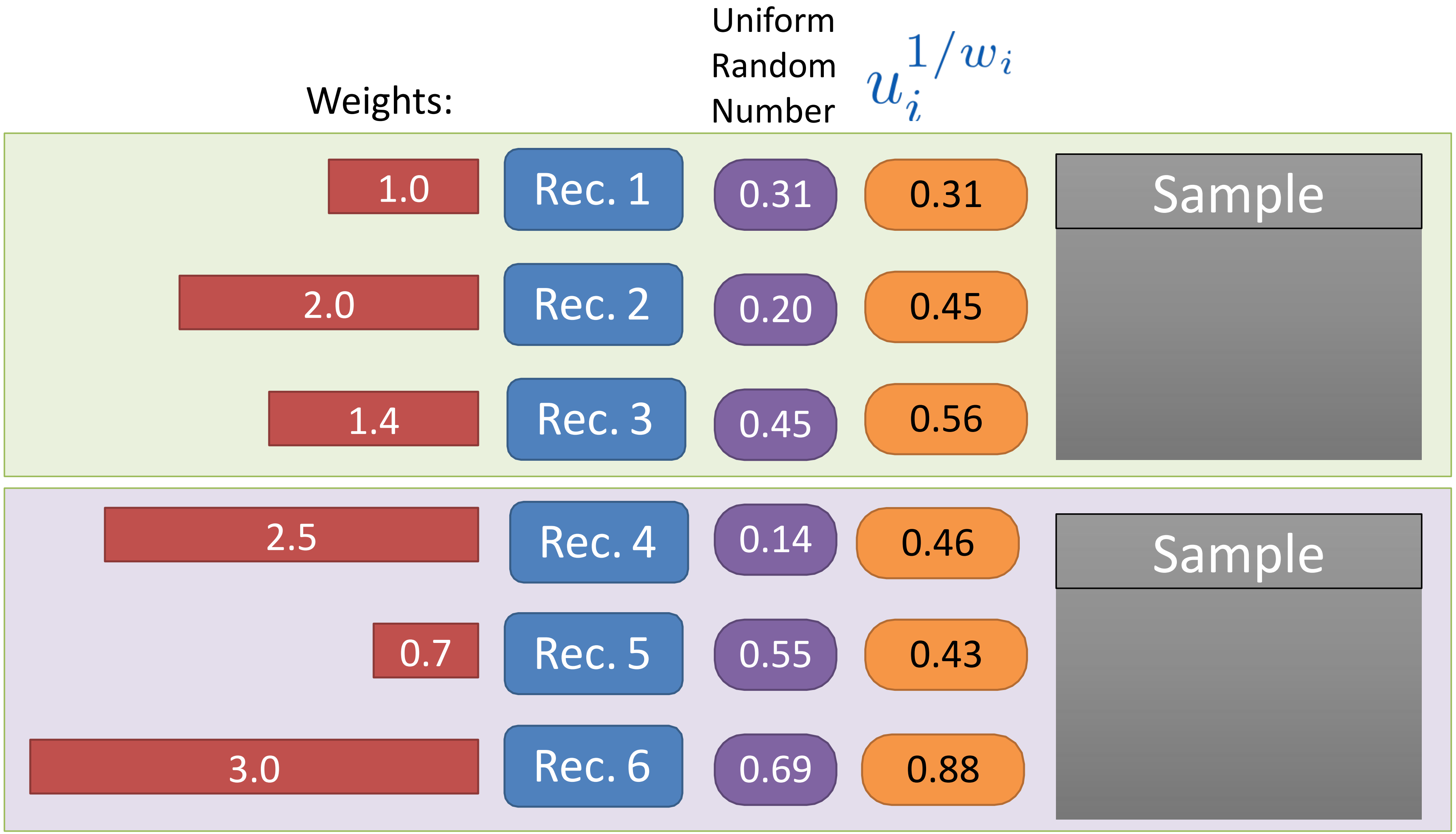
Weights:		Uniform Random Number	u_i^{1/w_i}
1.0	Rec. 1	0.31	0.31
2.0	Rec. 2	0.20	0.45
1.4	Rec. 3	0.45	0.56
2.5	Rec. 4	0.14	0.46
0.7	Rec. 5	0.55	0.43
3.0	Rec. 6	0.69	0.88

Top 2

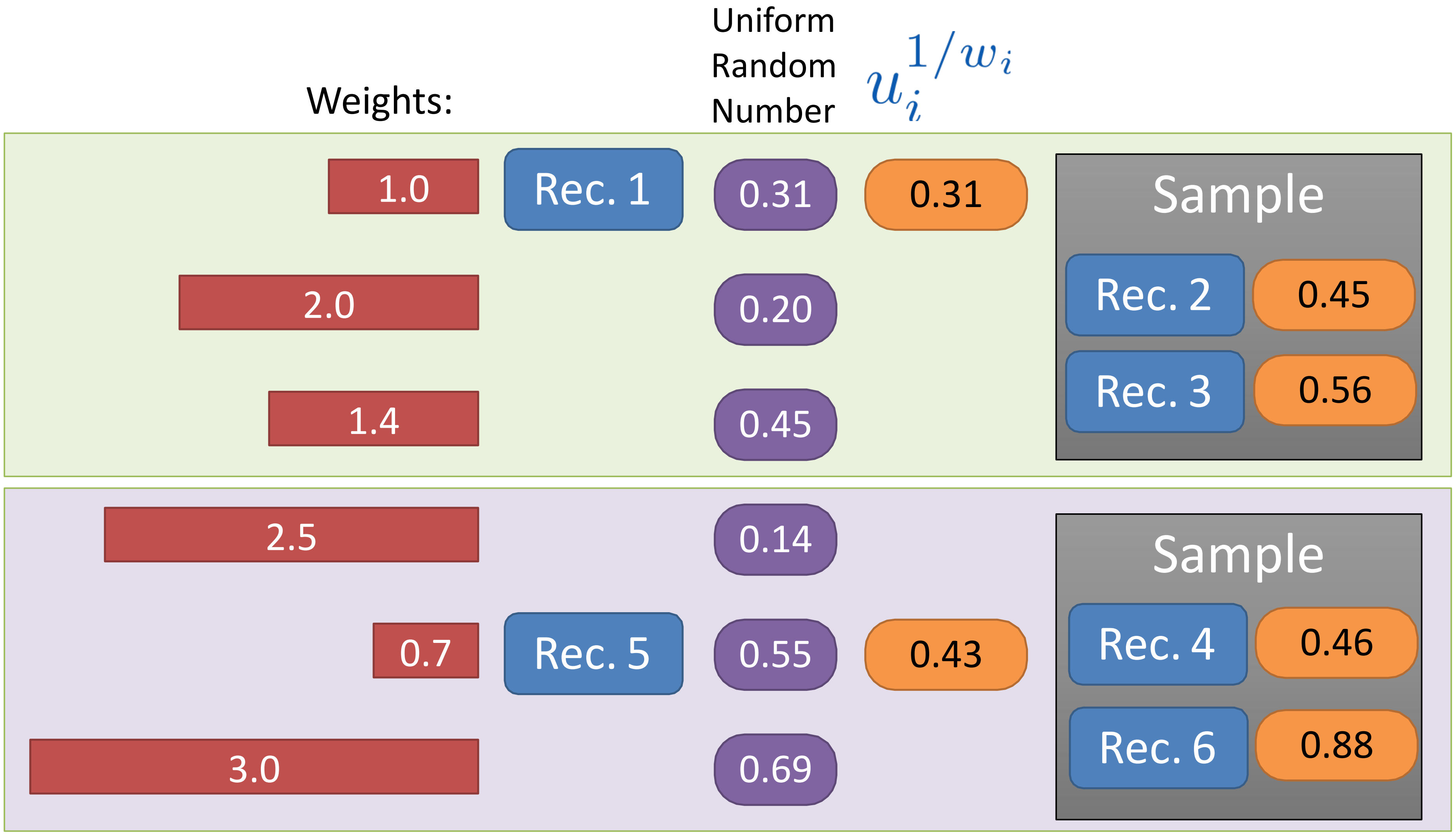
Illustrating Res-A Algorithm



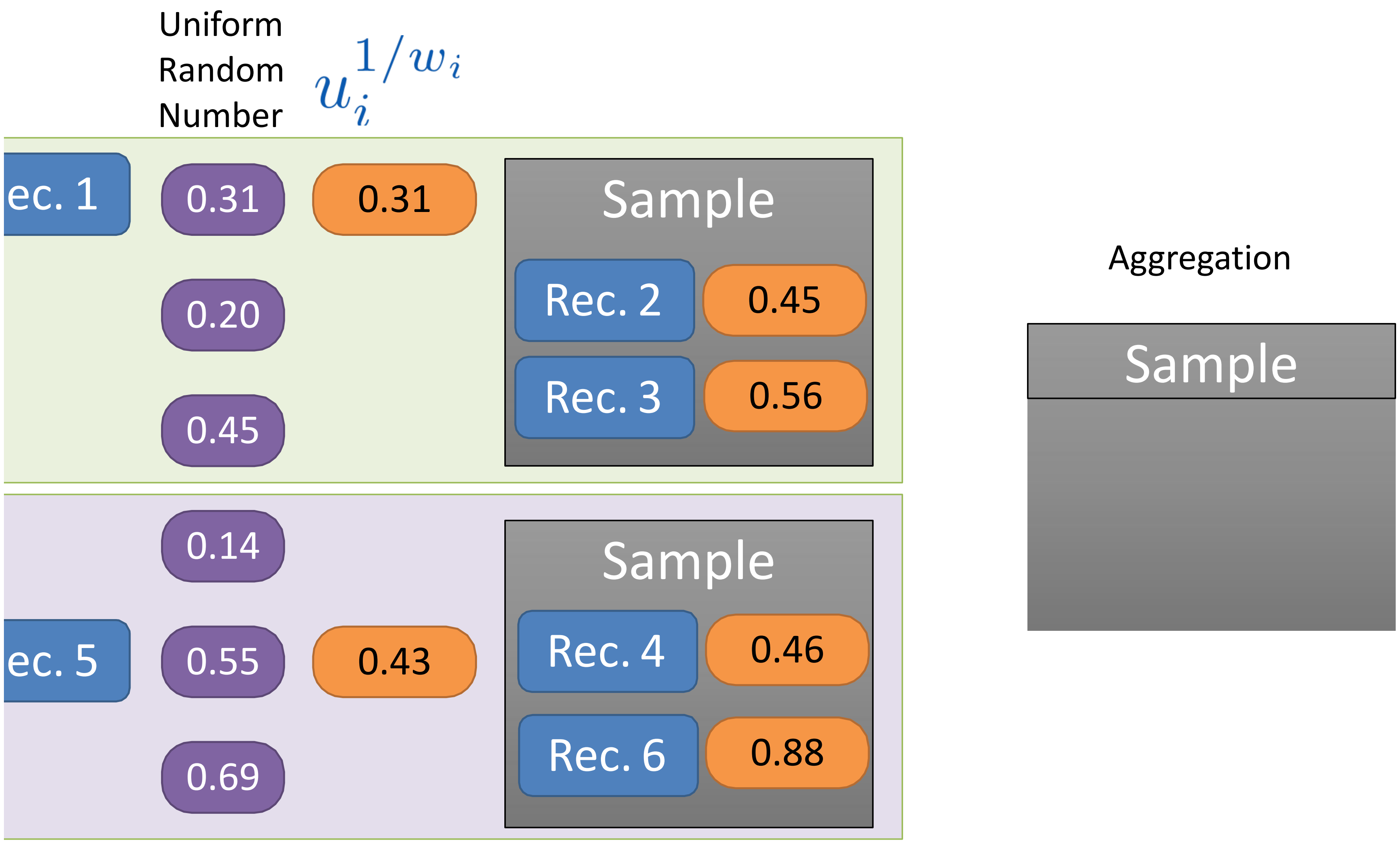
Parallelizing Res-A Algorithm



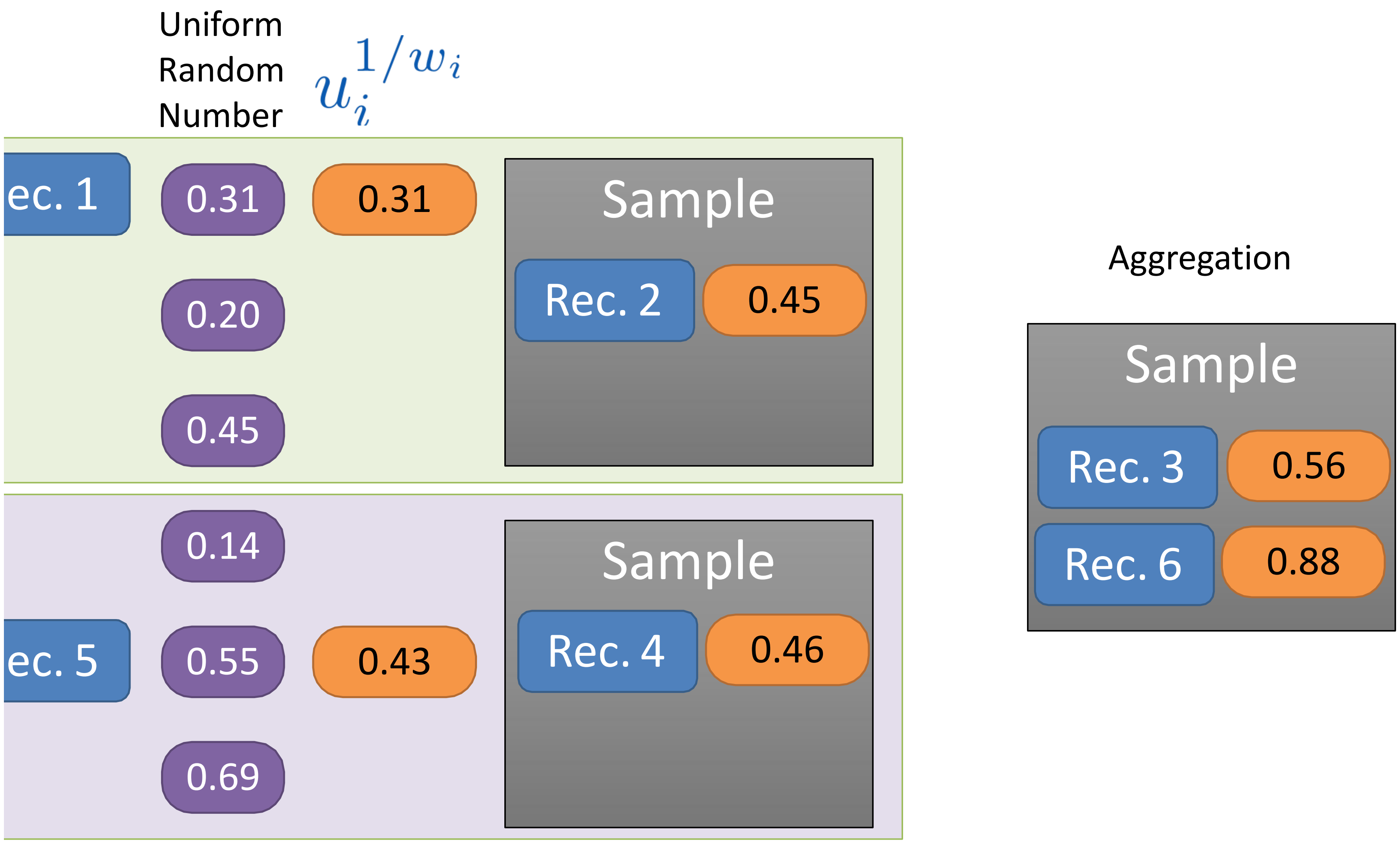
Parallelizing Res-A Algorithm



Parallelizing Res-A Algorithm



Parallelizing Res-A Algorithm



Taxonomy
of Machine Learning

Labeled Data

Indirect (reward)

Unlabeled Data

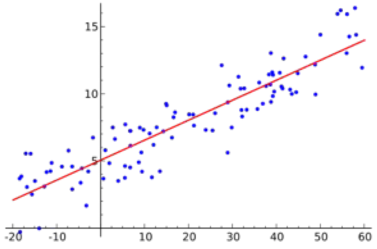
Supervised Learning

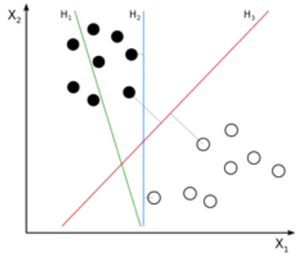
- Regression
- Classification

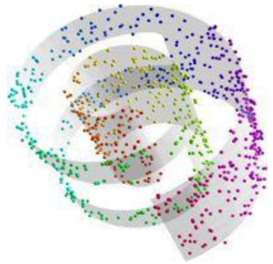
Reinforcement & Bandit Learning

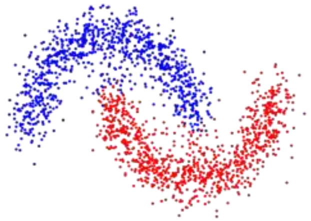
Unsupervised Learning

- Dimensionality Reduction
- Clustering









Taxonomy
of Machine Learning

Labeled Data

Indirect (reward)

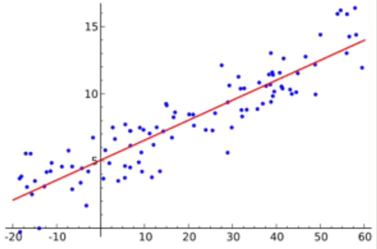
Unlabeled Data

Supervised Learning

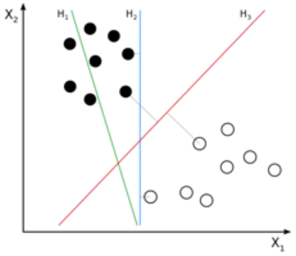
Reinforcement & Bandit Learning

Unsupervised Learning

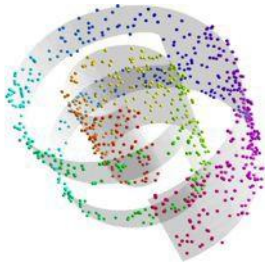
Regression



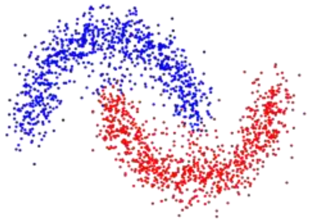
Classification



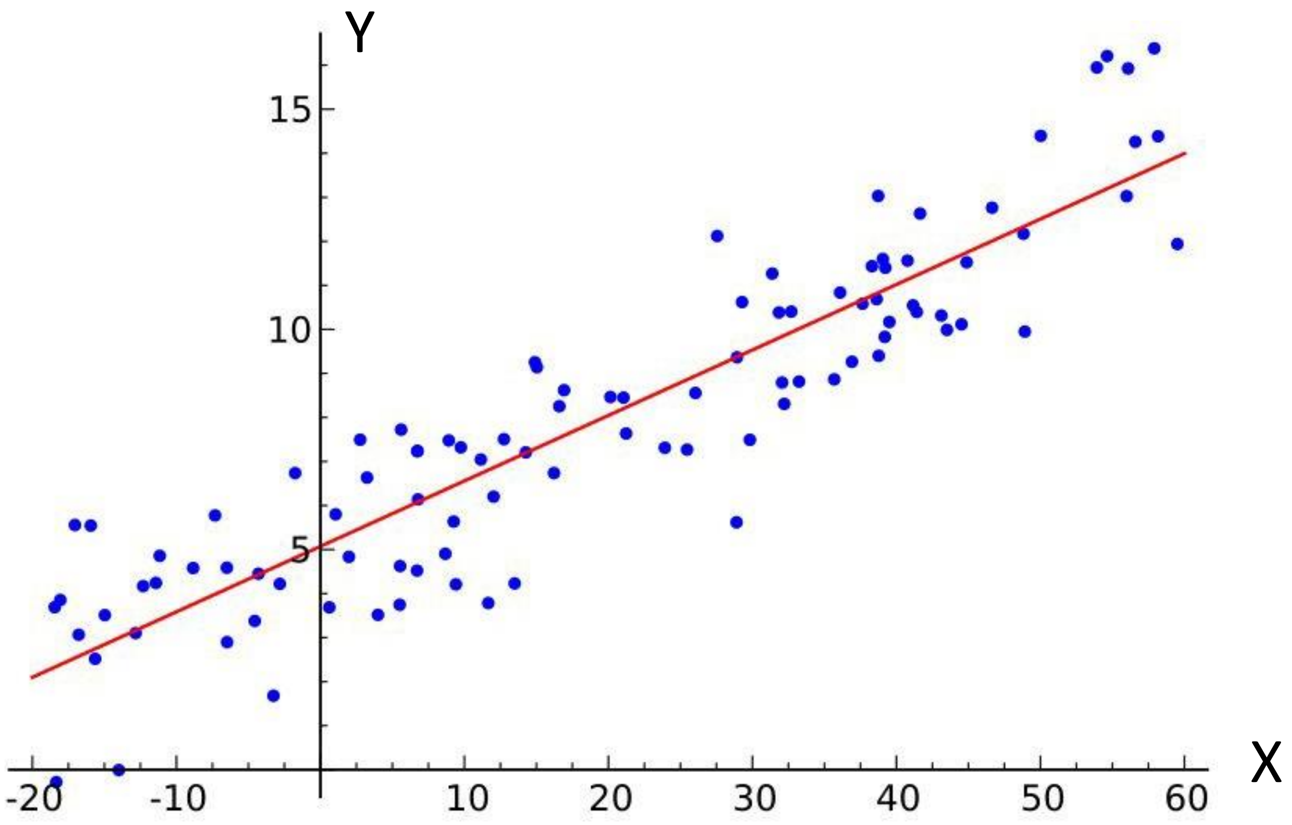
Dimensionality Reduction



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

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

Real Value Noise

Linear Combination of Covariates

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

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

Data:

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

$$\underbrace{x}_{\text{Vector of Features}} \in \mathbb{R}^p$$

"Real" data doesn't typically consist of entirely **real** valued features.

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 ...)

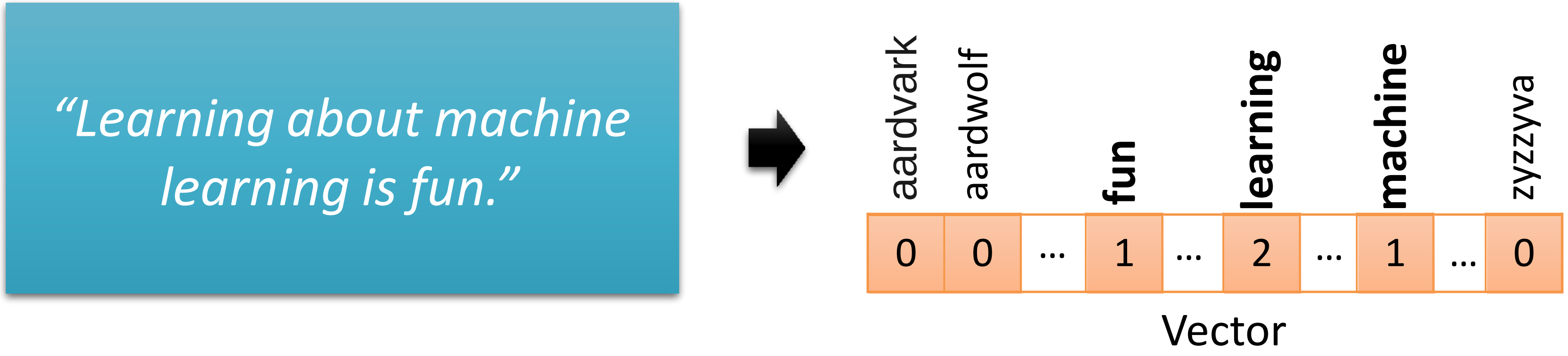
One-hot encoding

- □ 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?



□ 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")

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

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

Vector

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
…	…	…

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

- □ 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: $$f_\theta(x) := \theta^T x$$

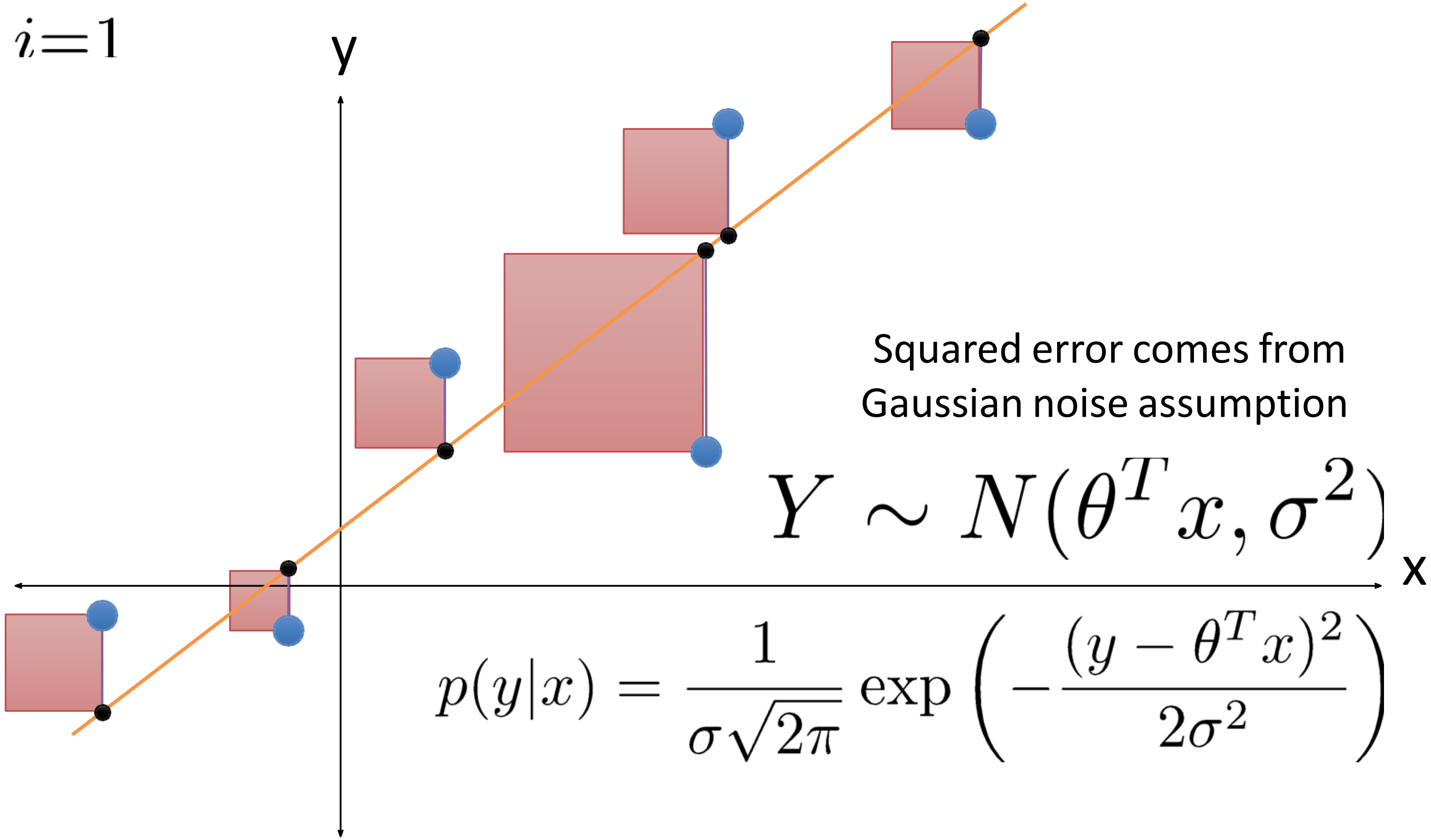
Step 1: define a **Loss Function:** *Average Prediction Error*

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

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

The meaning of Squared Loss (Error)

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



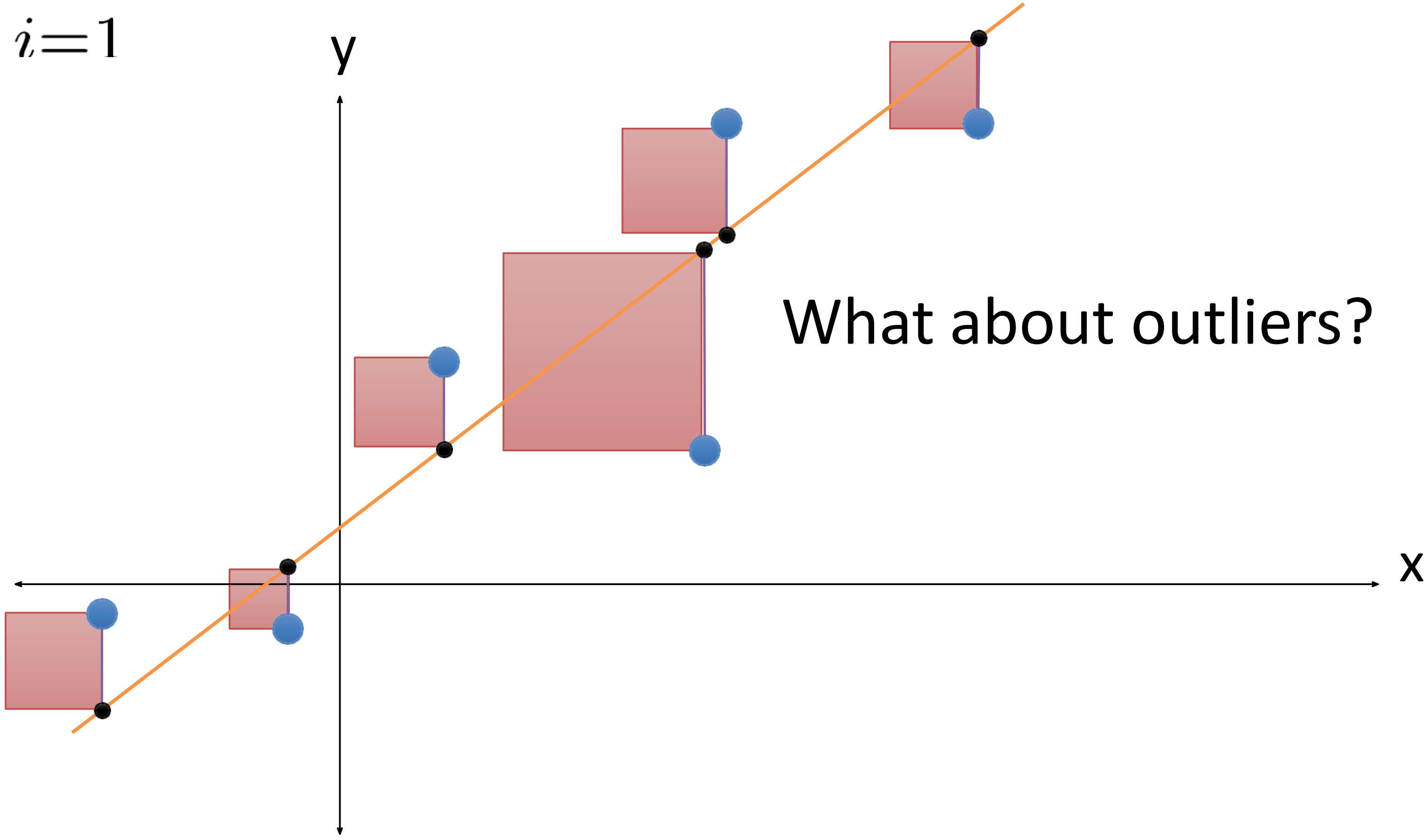
Squared error comes from
Gaussian noise assumption

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

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

The meaning of Squared Loss (Error)

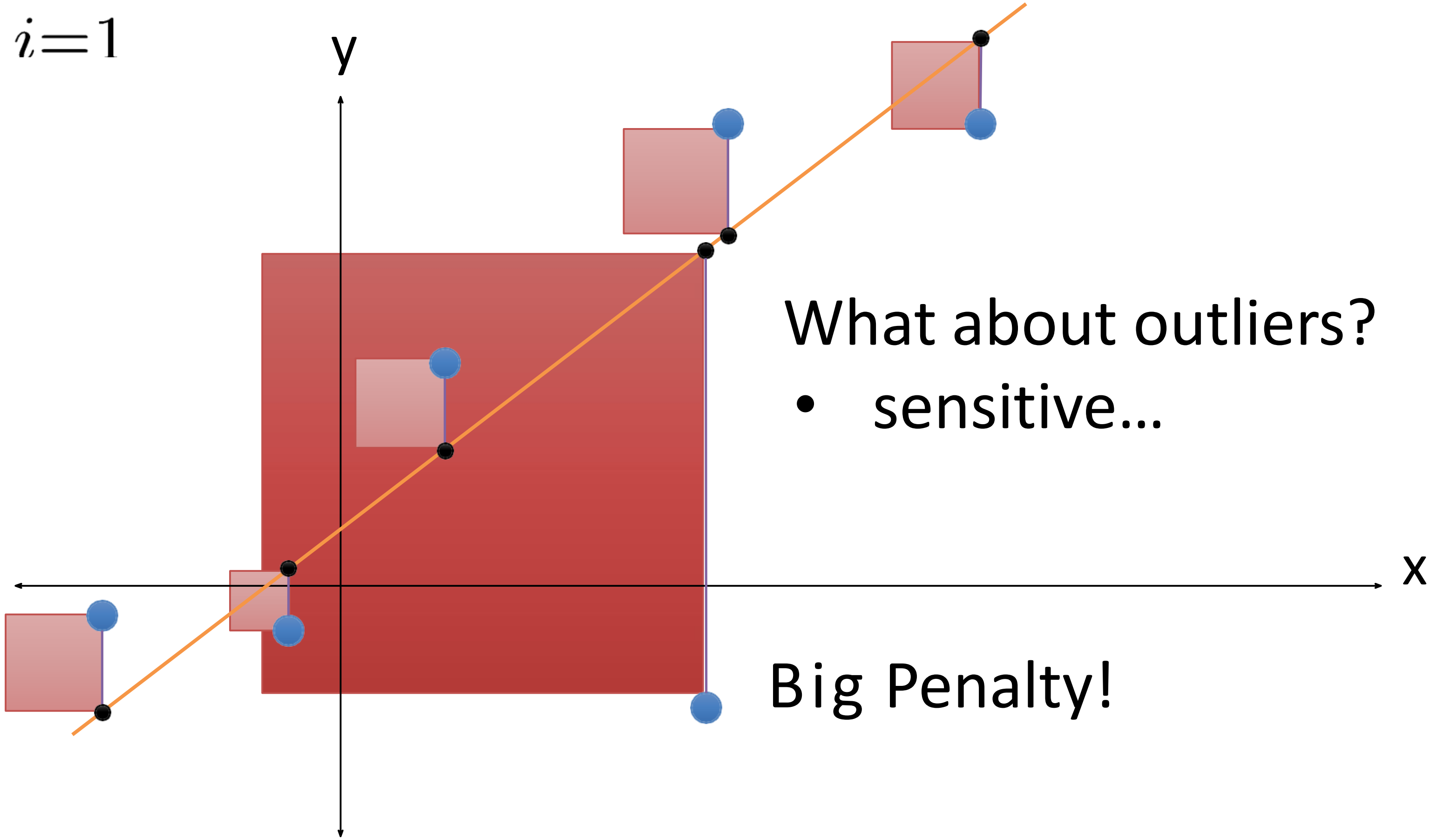
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^T x_i)^2$$



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:**

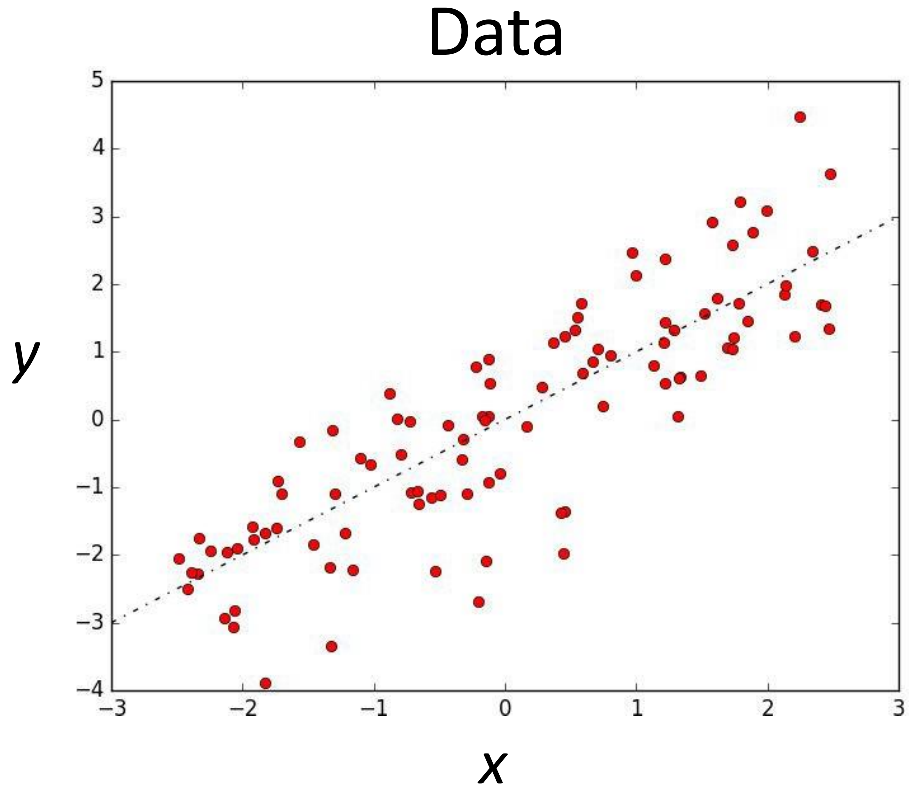
$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f_\theta(x_i)\right)^2$$

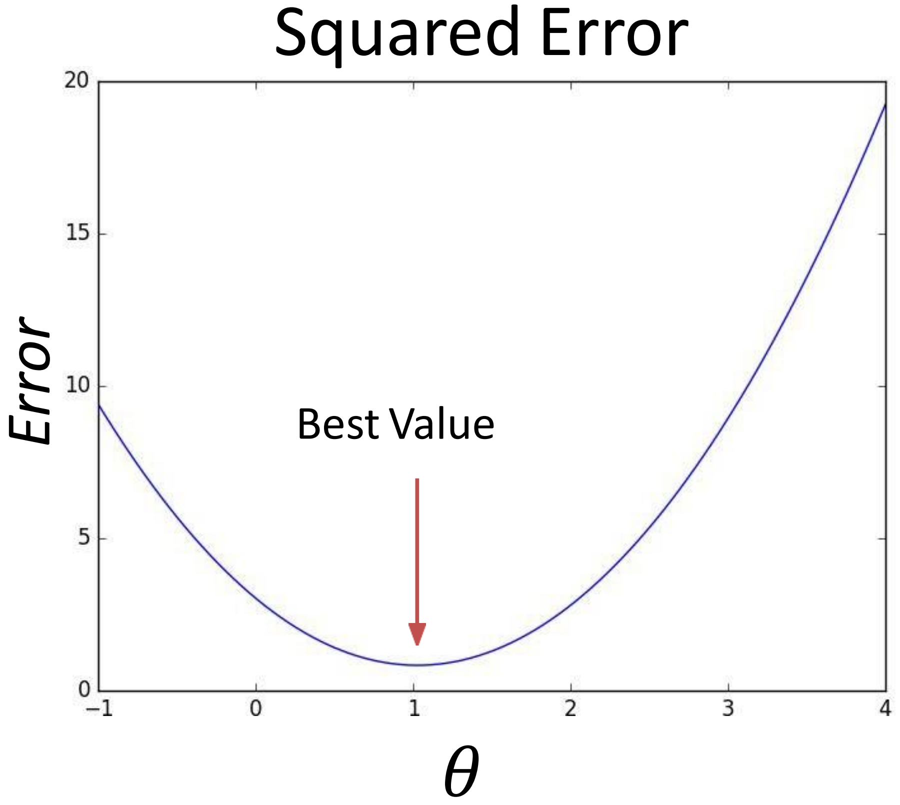
Step 2: Search for best model parameters θ

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

Minimizing the Squared Error

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

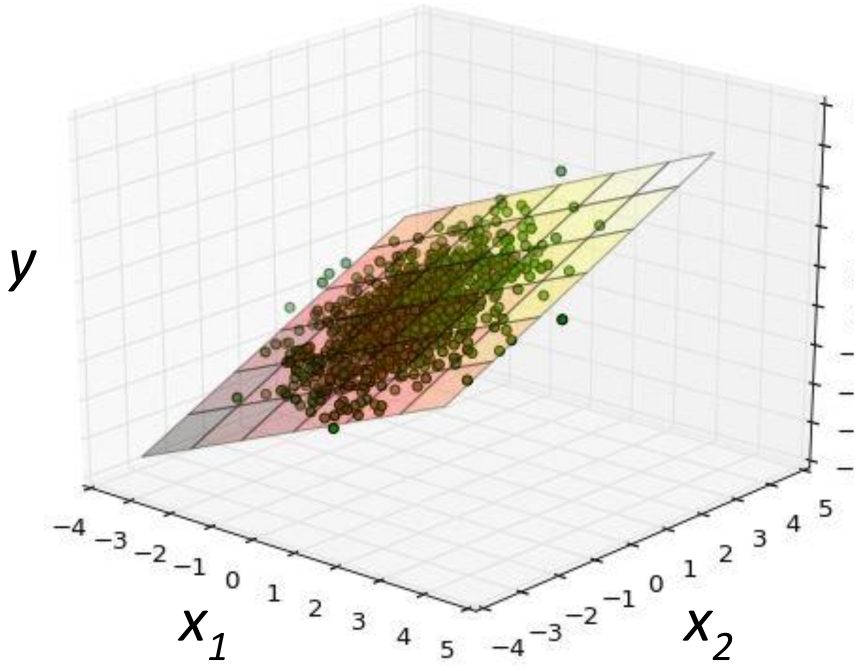




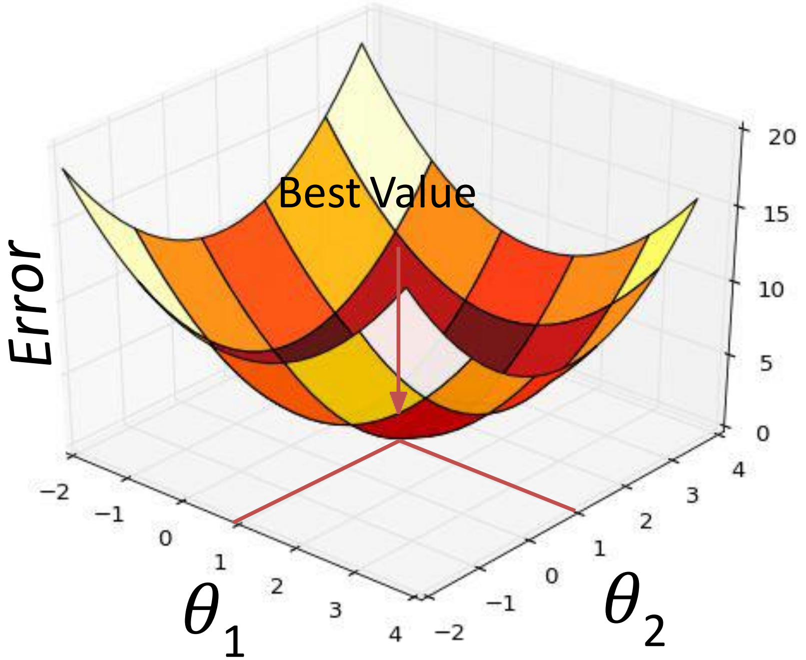
Minimizing the Squared Error

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

Data

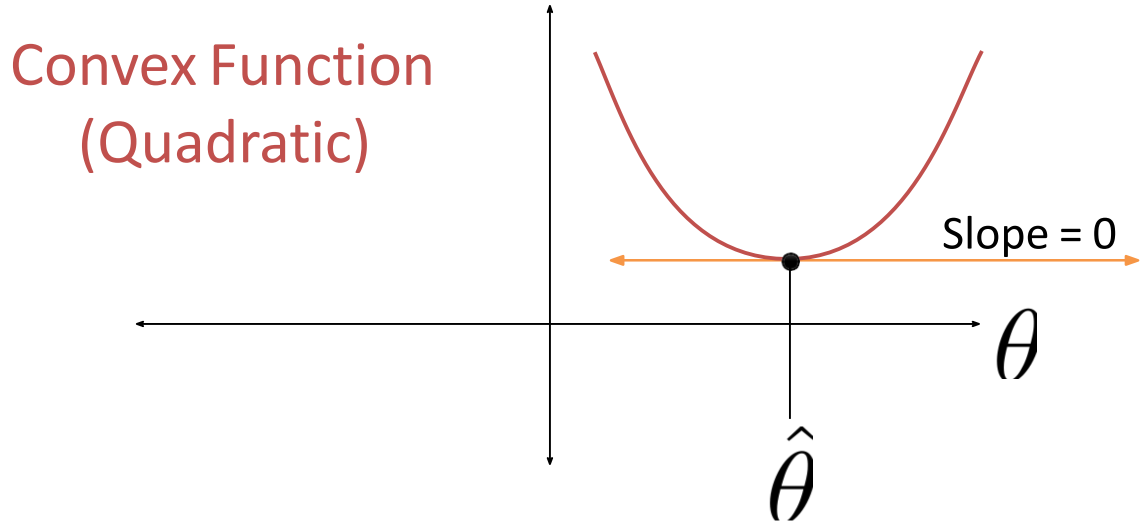


Squared Error



Minimizing the Squared Error

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



- □ 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

$$X = n\left\{ \underbrace{\begin{pmatrix} \text{---}\; x_1 \;\text{---} \\ \text{---}\; x_2 \;\text{---} \\ \dots \\ \text{---}\; x_n \;\text{---} \end{pmatrix}}_{p} \right. \in \mathbb{R}^{np}$$

Response Vector

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

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:

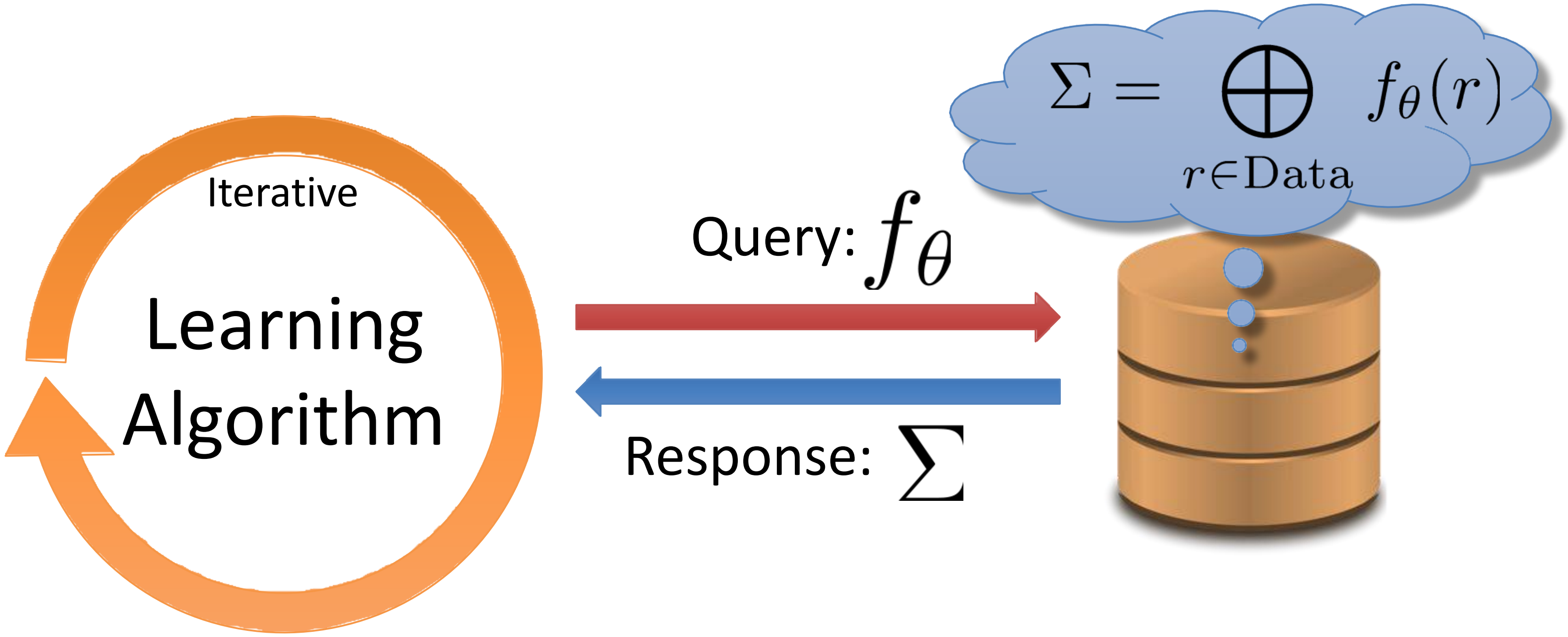
$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

□ Solved using any standard linear algebra library

Can we compute

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

using the statistical query pattern?

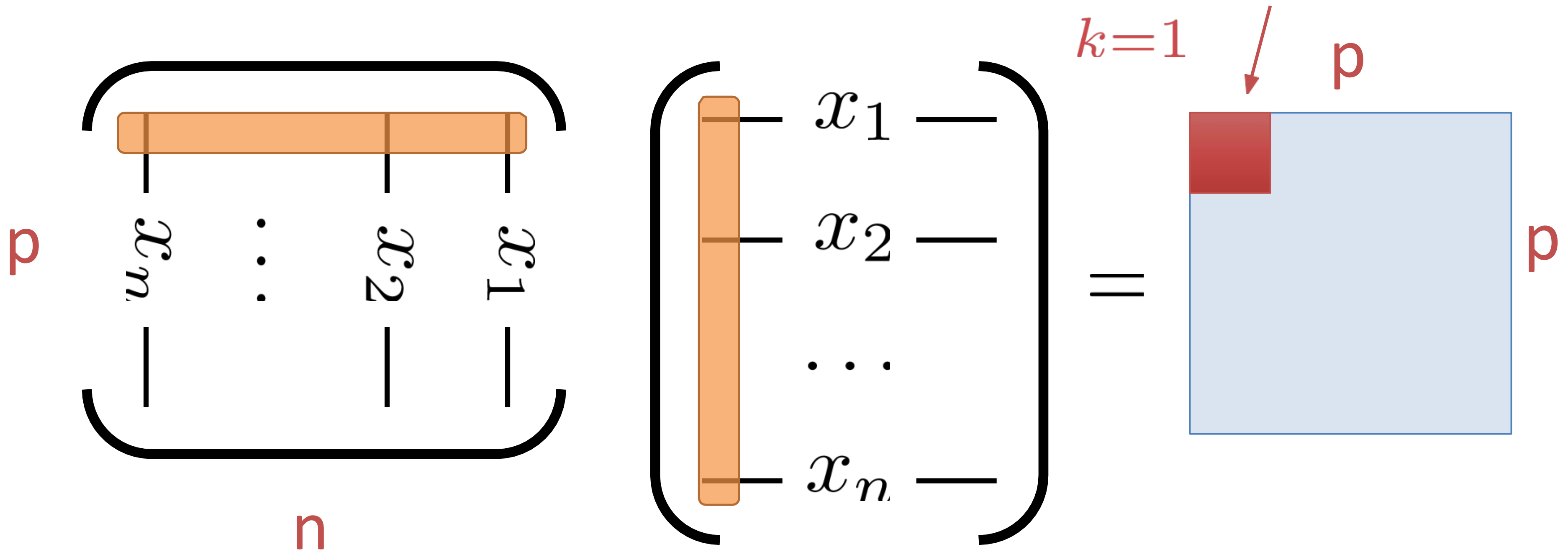


$$\hat{\theta} = (\underbrace{X^T X}_{\text{Query 1}})^{-1} \underbrace{X^T Y}_{\text{Query 2}}$$

Break
computation
into two queries

Computing Query 1

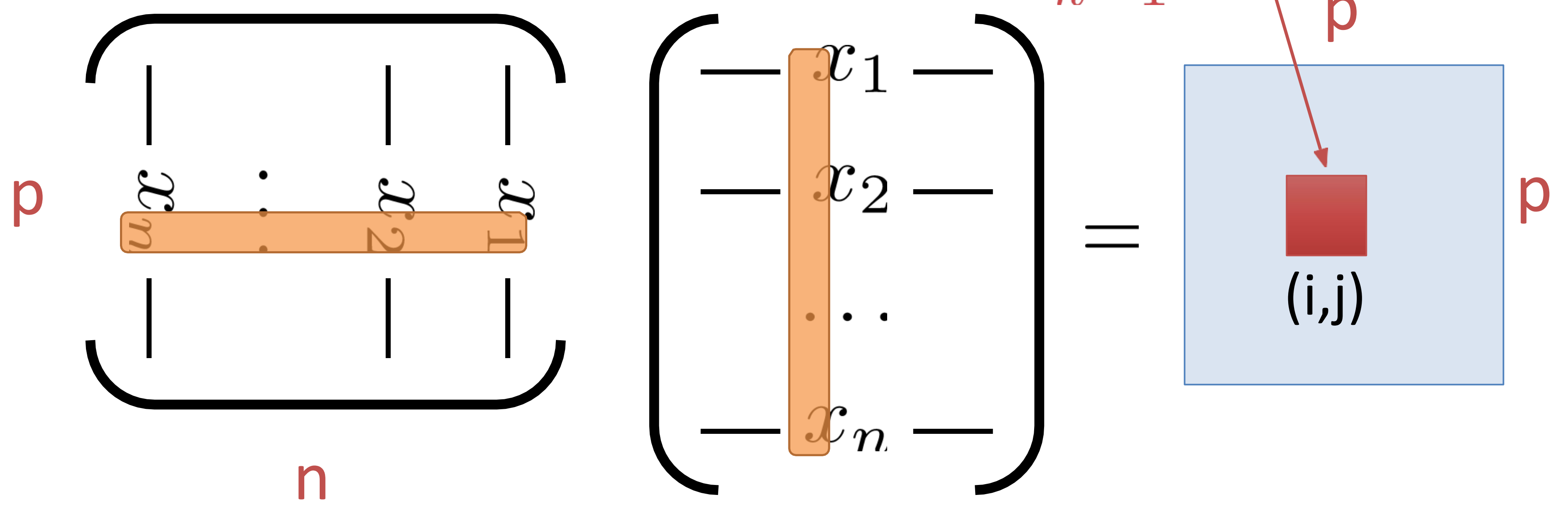
$$X^T X =$$



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:

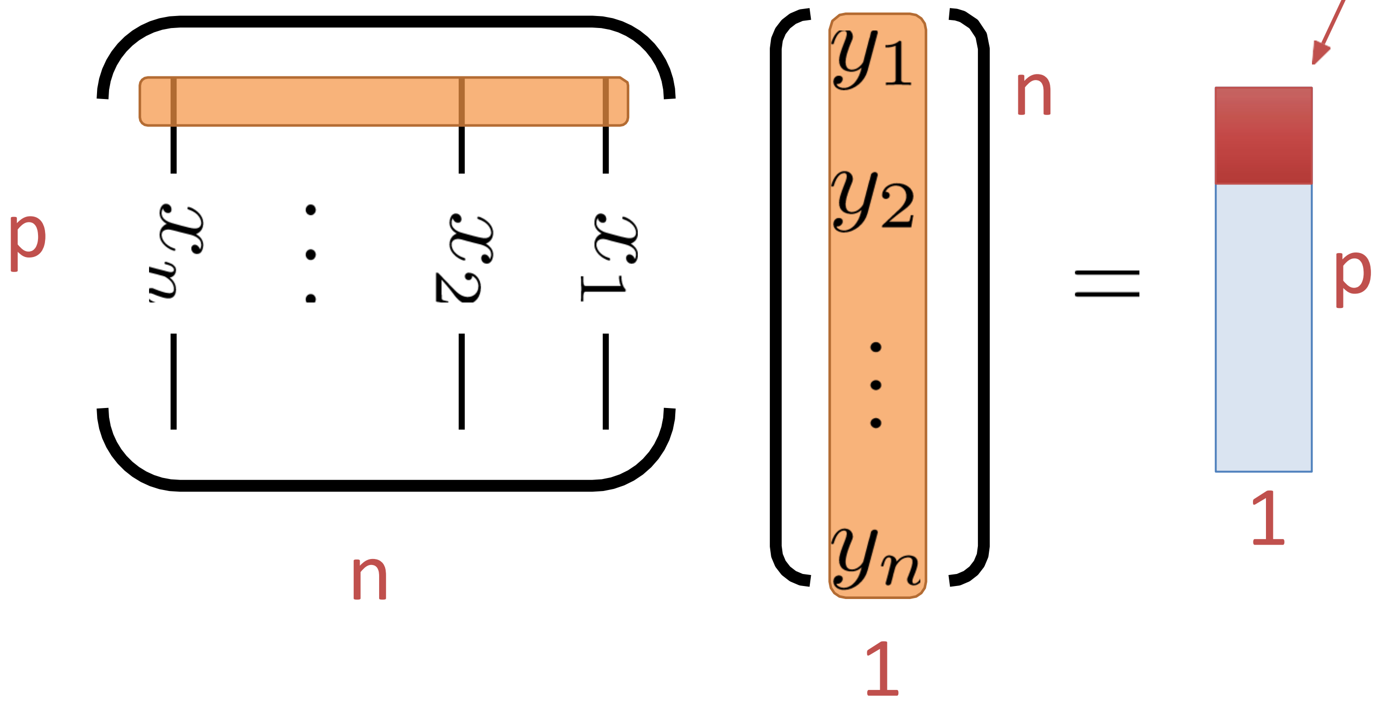
- Pure SQL Expression:

```
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

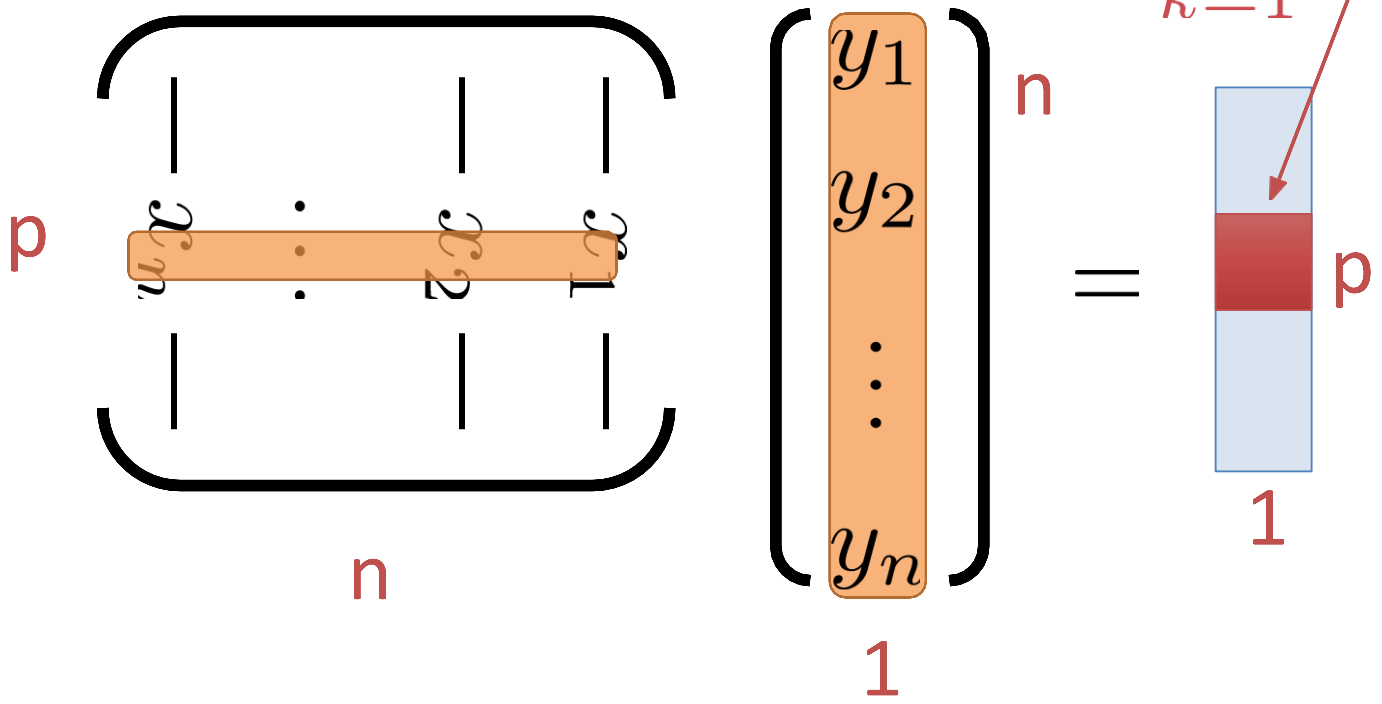
$$X^T y =$$

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



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:

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

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

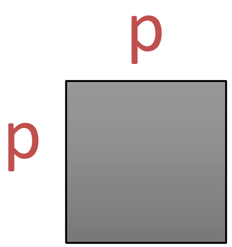
- □ Pure SQL Expression:

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

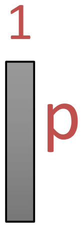
Least Squares Regression using the **Statistical Query Pattern**

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

- In database compute sums:



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

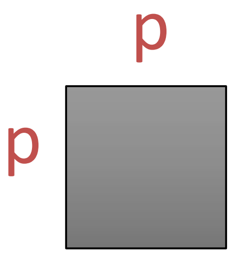


$$d = X^T y = \sum_{i=1}^{n} x_i y_i \qquad O(np)$$

- On client compute:

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

What if p is large?



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

… could be expensive …

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

□ Rather than directly solving:

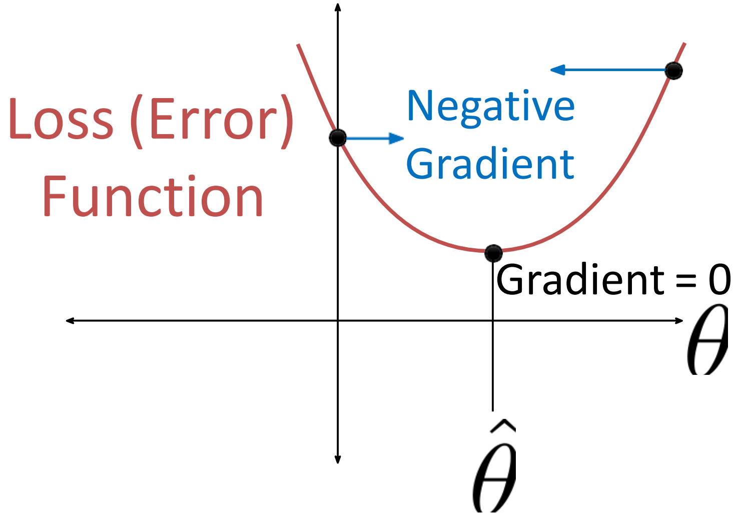
$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} L(y_i, \theta^T x_i)$$

□ Instead we compute the gradient of the loss:

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

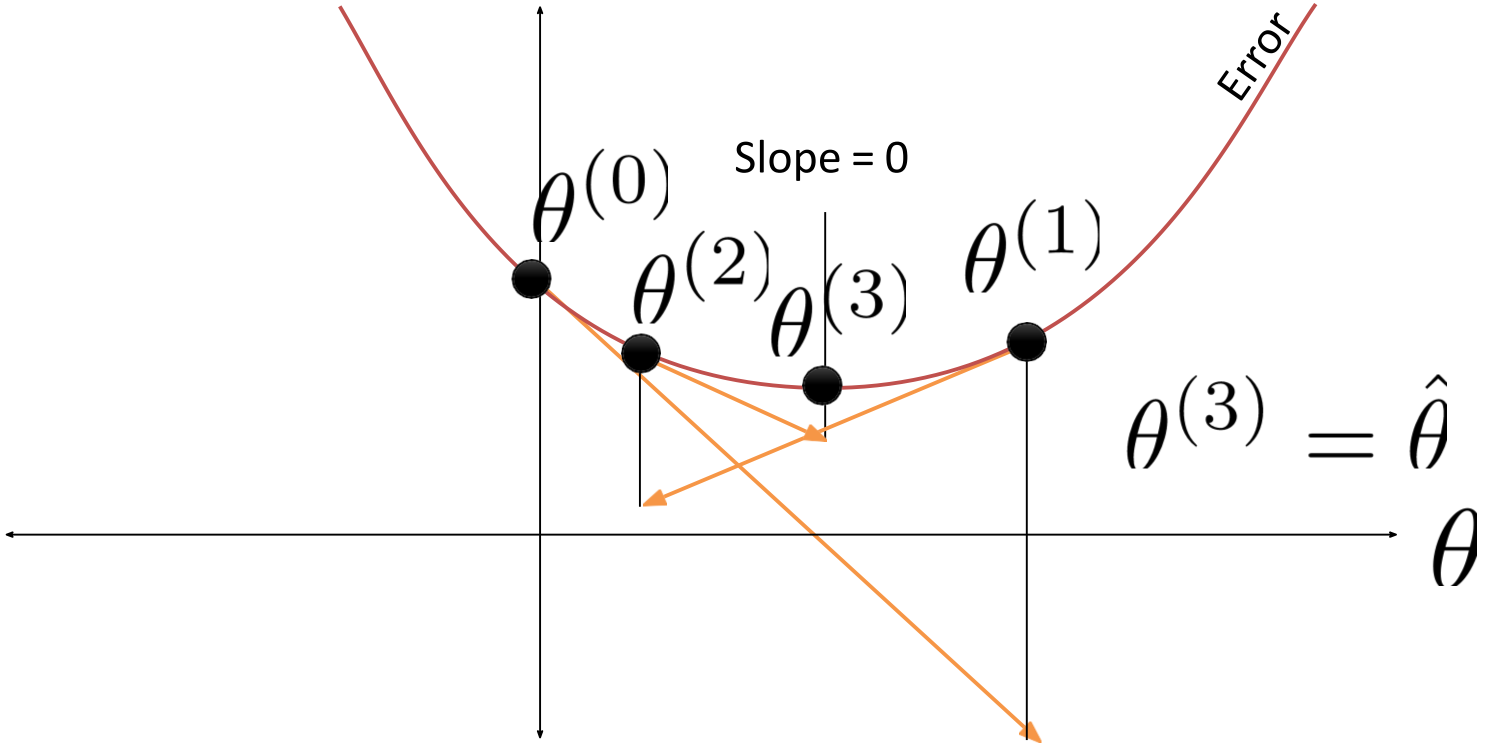
$$= \frac{1}{n} \sum_{i=1}^{n} (y - \theta^T x_i) x_i$$

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



Gradient Descent Algorithm

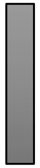
```
t ← 0
θ^(0) ← Vec(0)
while (not converged):
    θ^(t+1) ← θ^(t) – η * G(θ; X,Y)
    t ← t + 1
```



Gradient Descent Algorithm

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

- Does this fit the statistical query pattern
  - Yes! Only dependence on data is:

Size? 

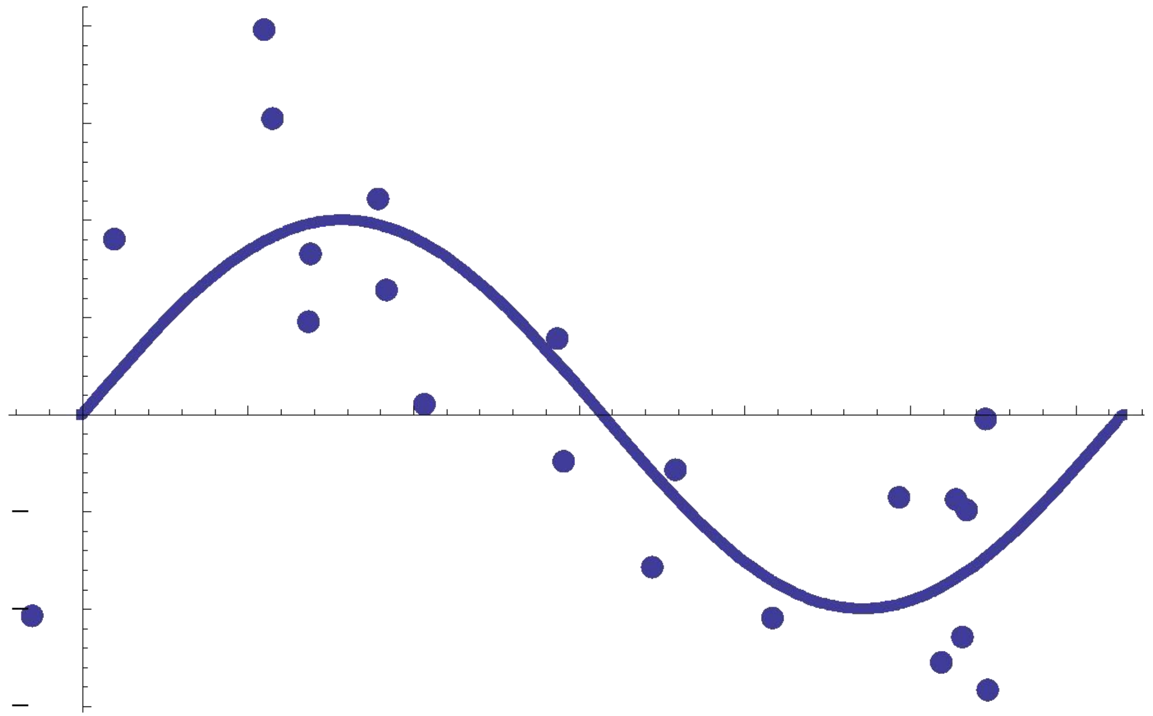
$$G(\theta; X, y) = \frac{1}{n}\sum_{i=1}^{n}(y - \theta^T x_i)x_i$$

Complexity? $O(np)$

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

Fitting Non-linear Data

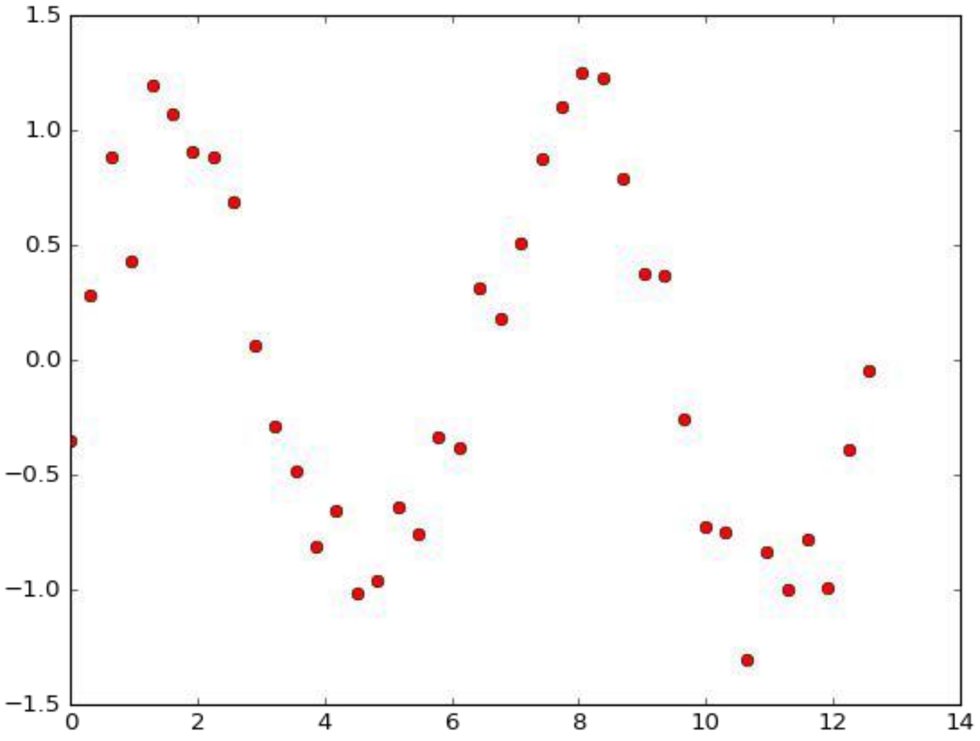
- What if Y has a non-linear response?



- Can we still use a linear model?

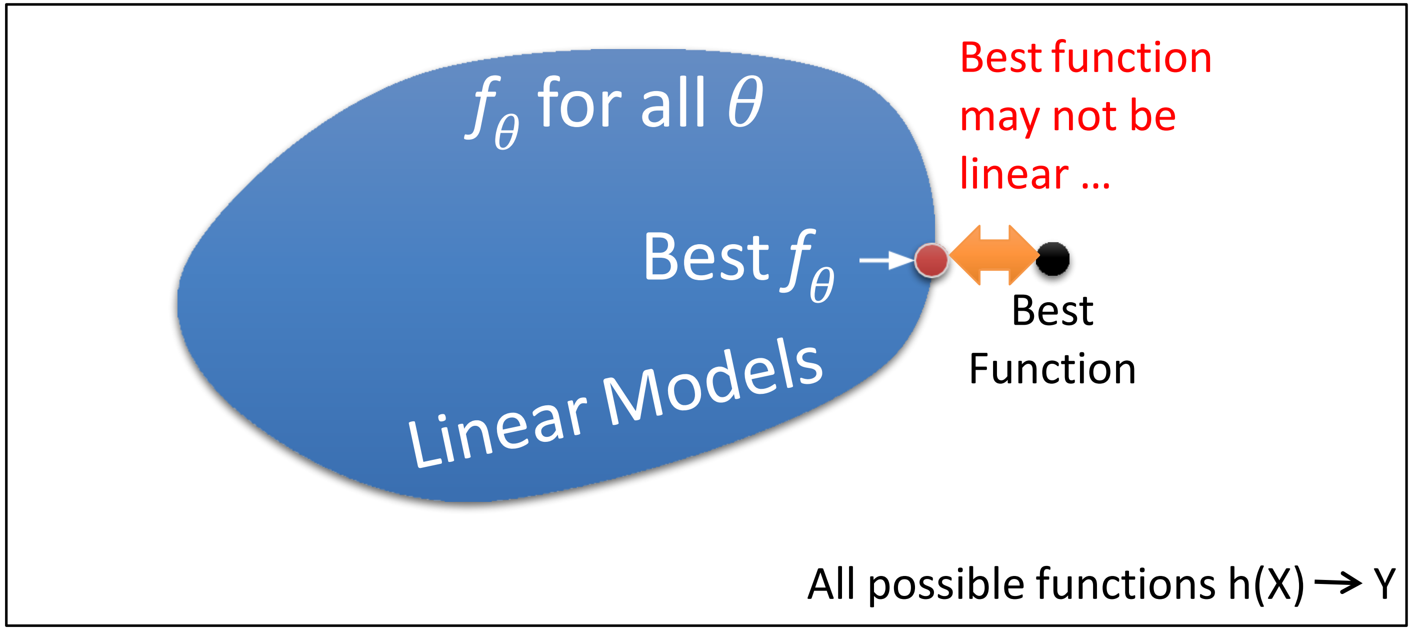
Fitting Non-linear Data

□ What if Y has a non-linear response?

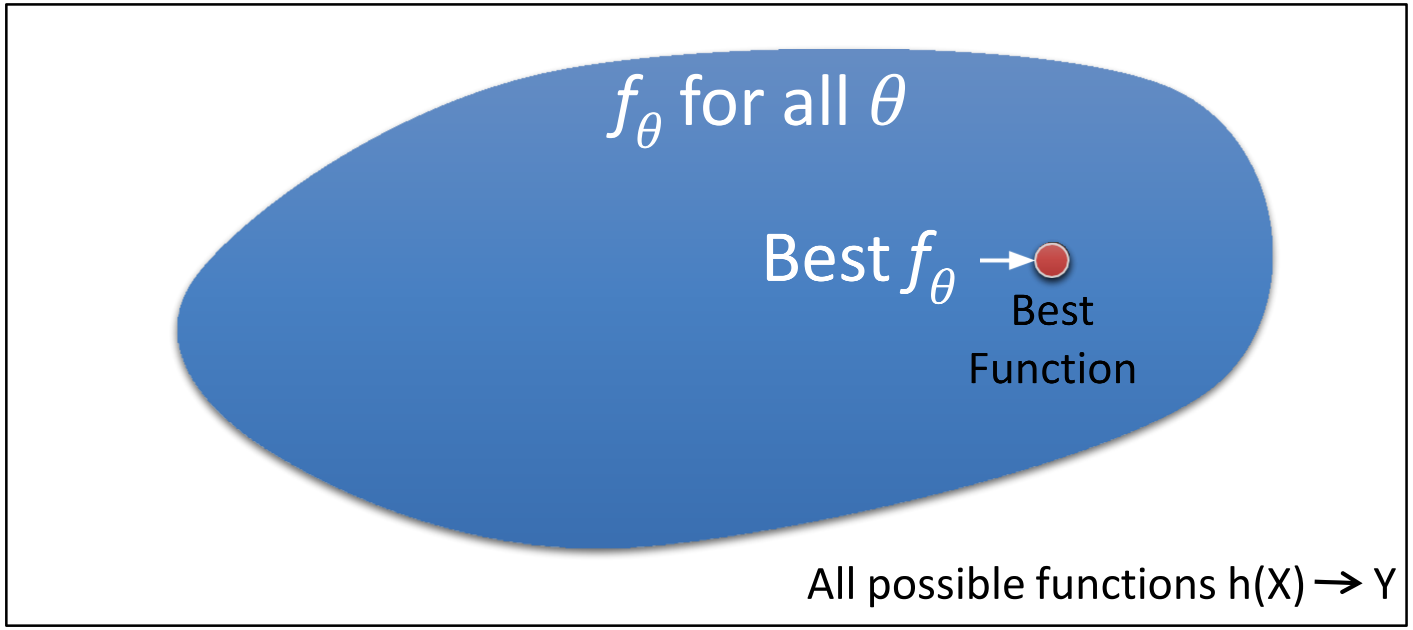


□ Can we still use a linear model?

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:

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

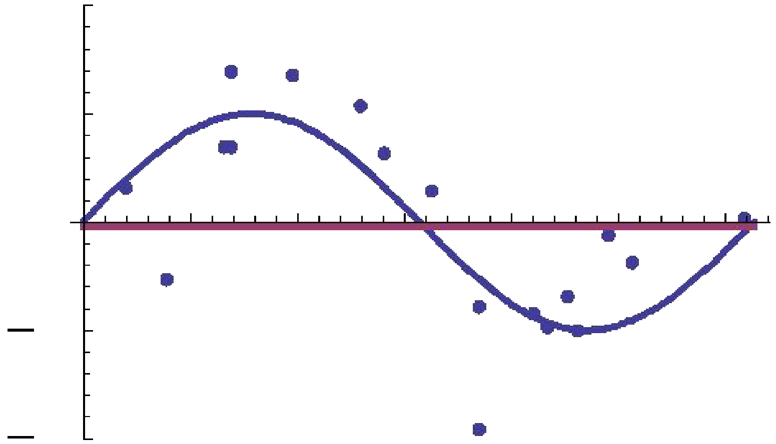
x_1	x_2	y
1.1	2.7	3.6
4.2	3.2	7.5
9.8	9.2	17

$\xrightarrow{\phi}$

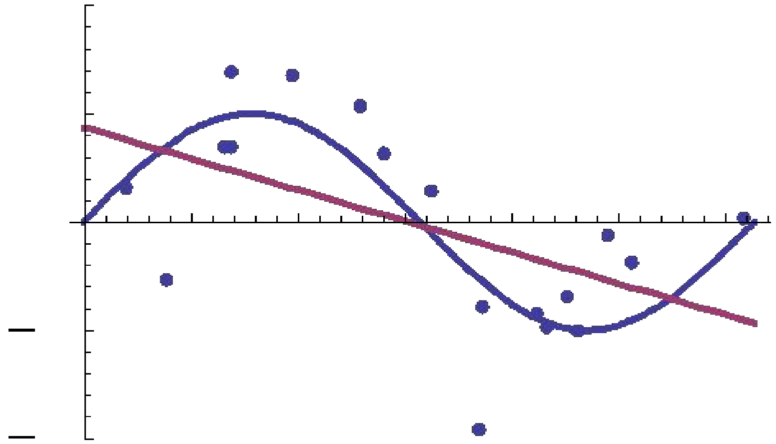
x_1	x_2	x_1*x_1	x_2*x_2	x_1*x_2	y
1.1	2.7	1.21	7.29	2.97	3.6
4.2	3.2	17.64	10.24	13.44	7.5
9.8	9.2	90.04	84.64	90.16	17

Under-fitting

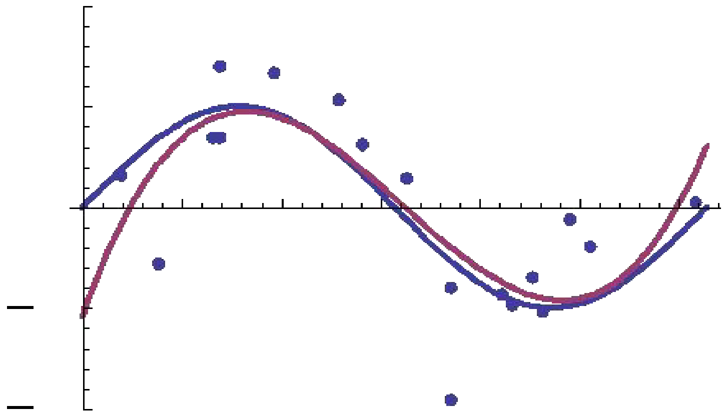
$\{\, 1 \,\}$



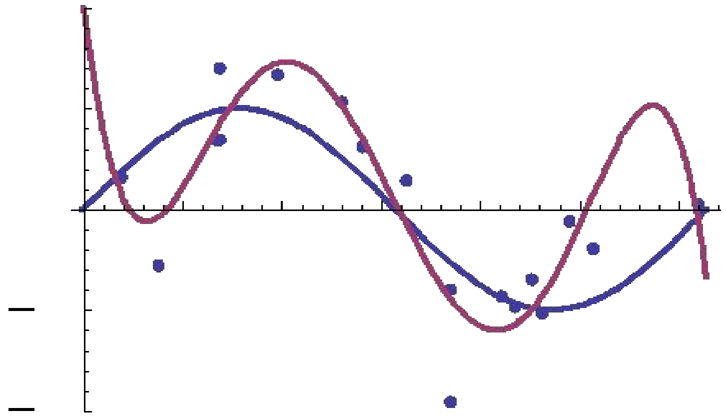
$\{\, 1, x \,\}$



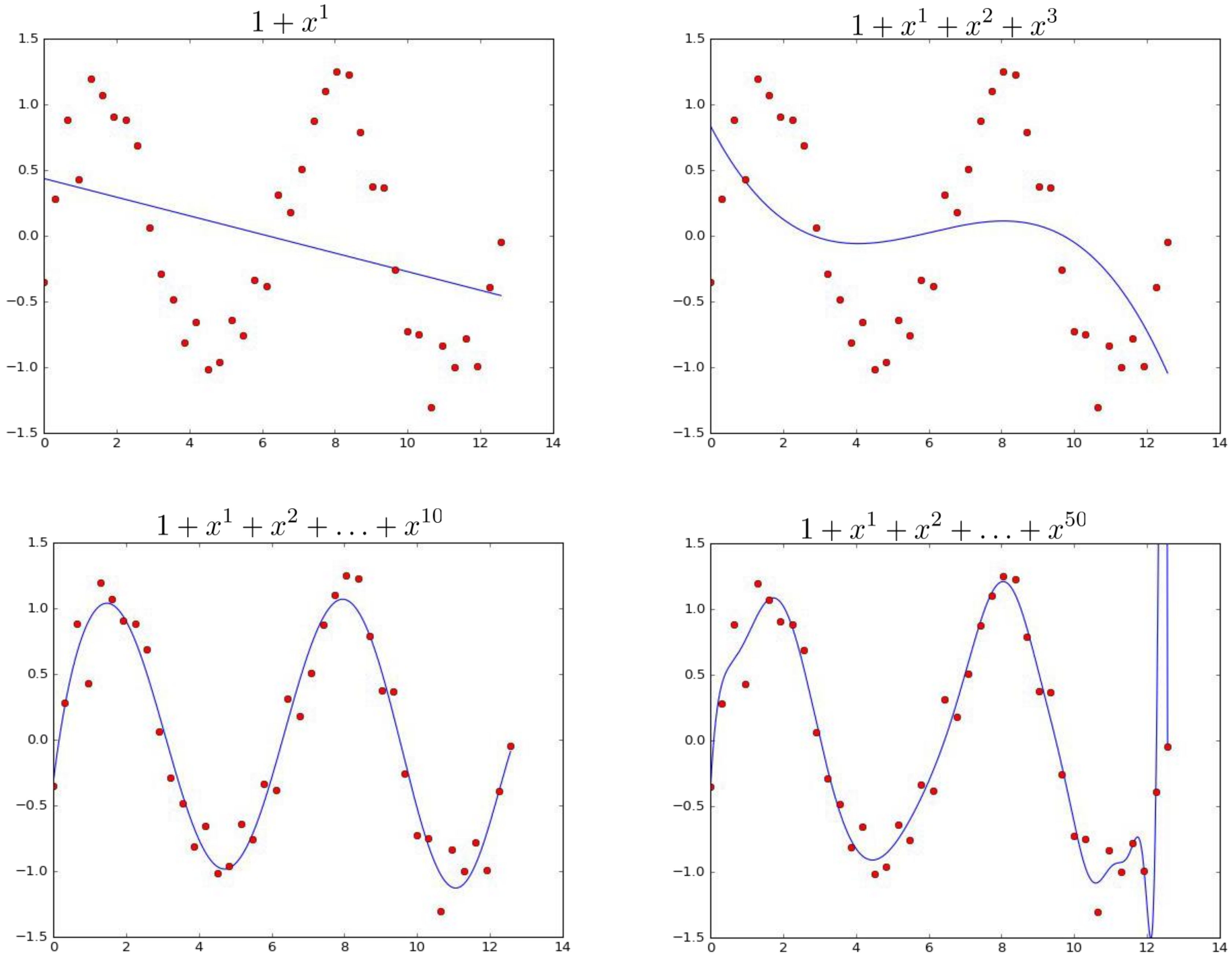
$\{\, 1, x, x^2, x^3 \,\}$



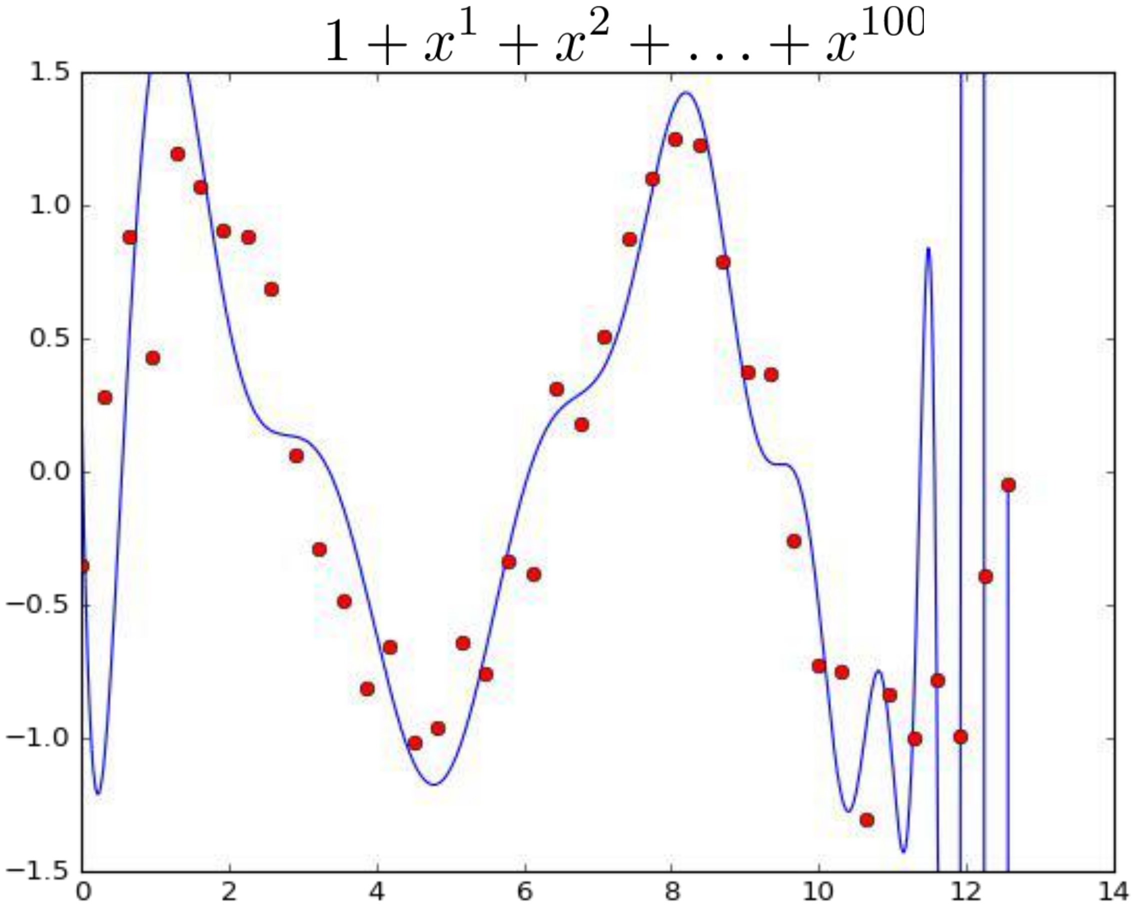
$\{\, 1, x, x^2, x^3, x^4, x^5 \,\}$



Over-fitting


$1 + x^1$
$1 + x^1 + x^2 + x^3$
$1 + x^1 + x^2 + \ldots + x^{10}$
$1 + x^1 + x^2 + \ldots + x^{50}$

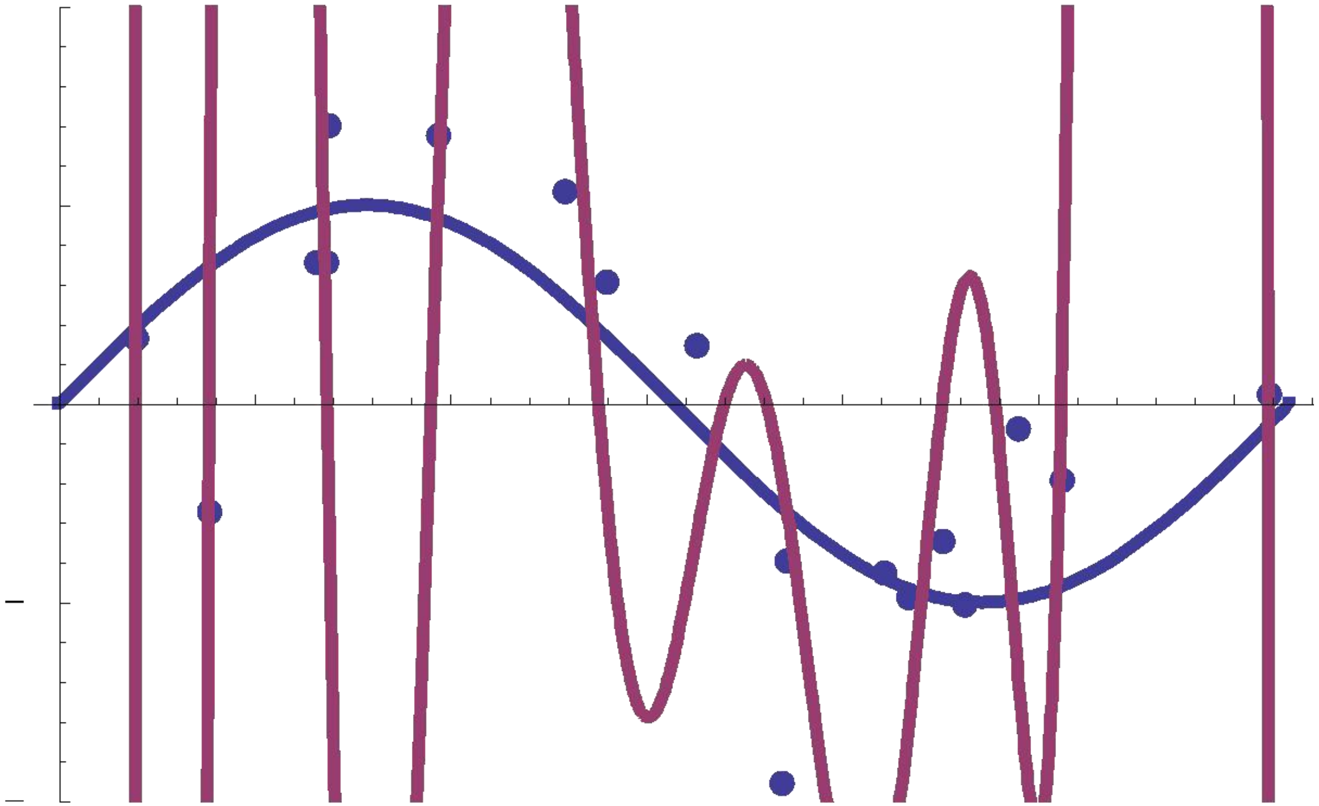
Over-fitting!



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

Really Over-fitting!

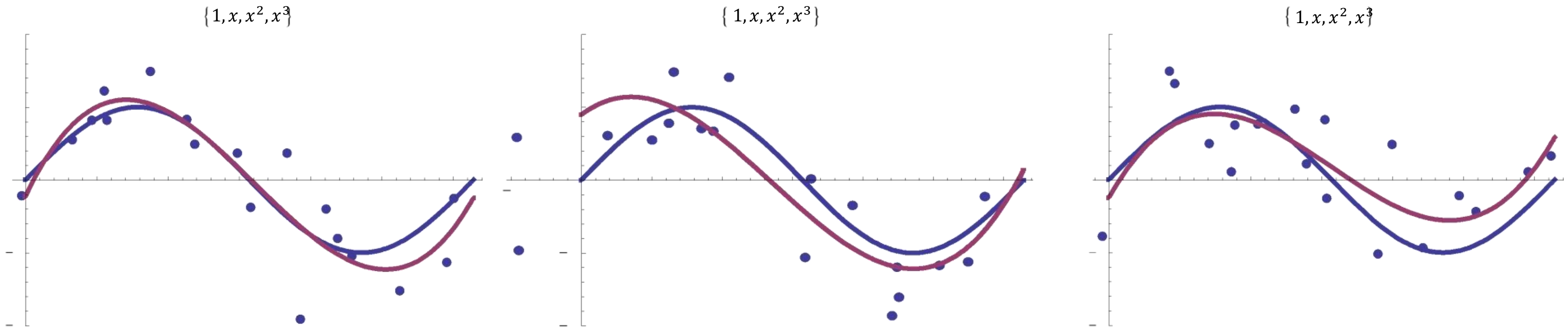
$$\{\ 1, x, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}, x^{11}, x^{12}, x^{13}, x^{14}\ \}$$



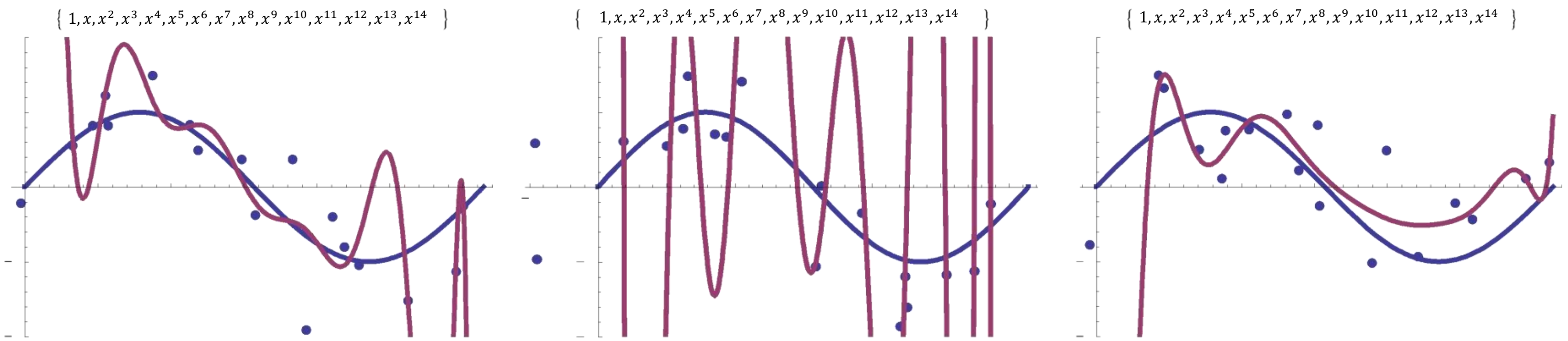
- Errors on training data are small
- But errors on new points are likely to be large

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**

Theorem:

$$\text{Test Error} = \text{Noise} + \text{Bias}^2 + \text{Variance}$$

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

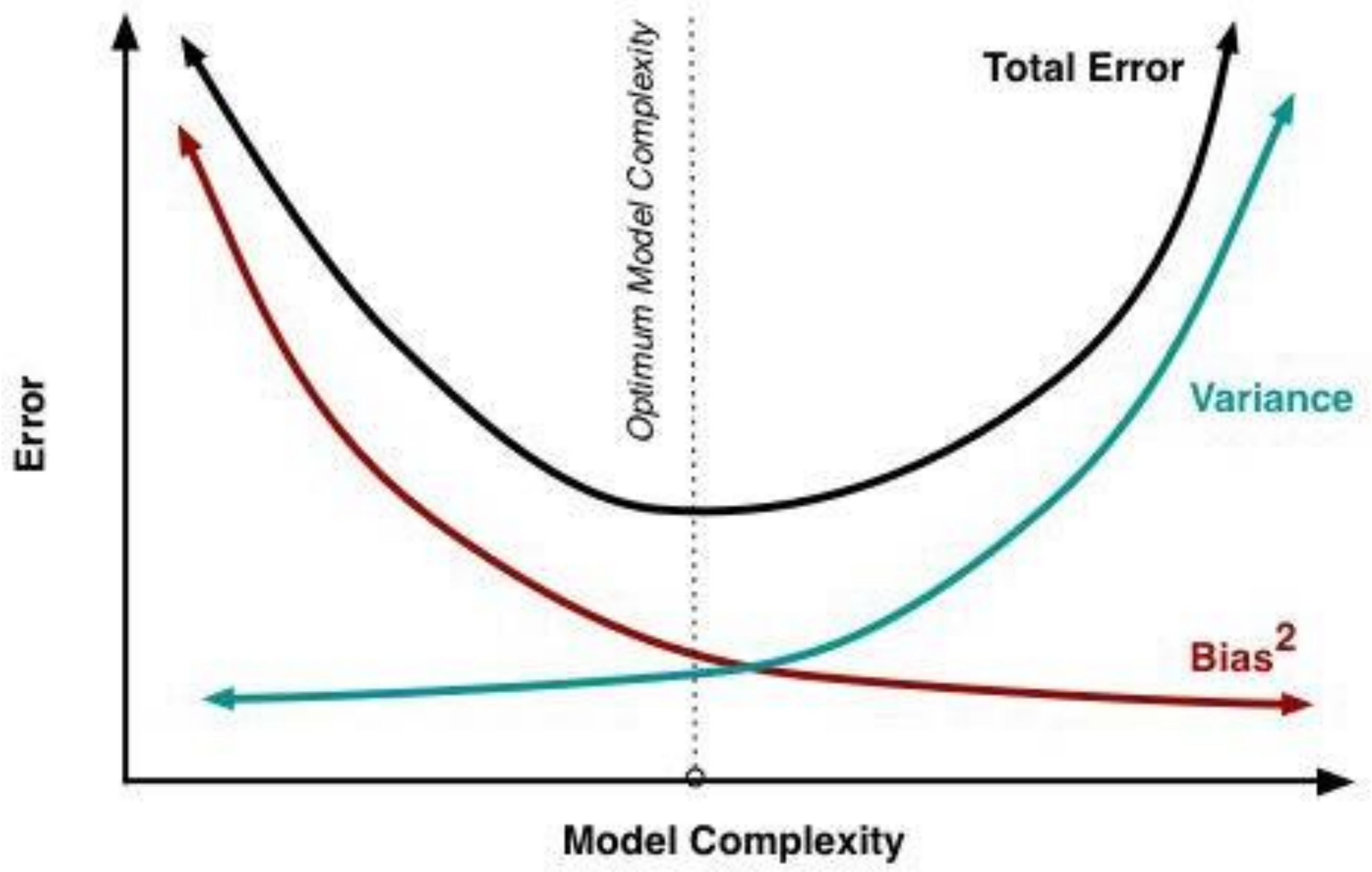


Image from

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")

$$f(x) = \theta_1 x_1 + \theta_2 x_2 + 0 x_3 + \ldots + 0 x_d$$
$$= \theta_1 x_1 + \theta_2 x_2$$

  - 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:

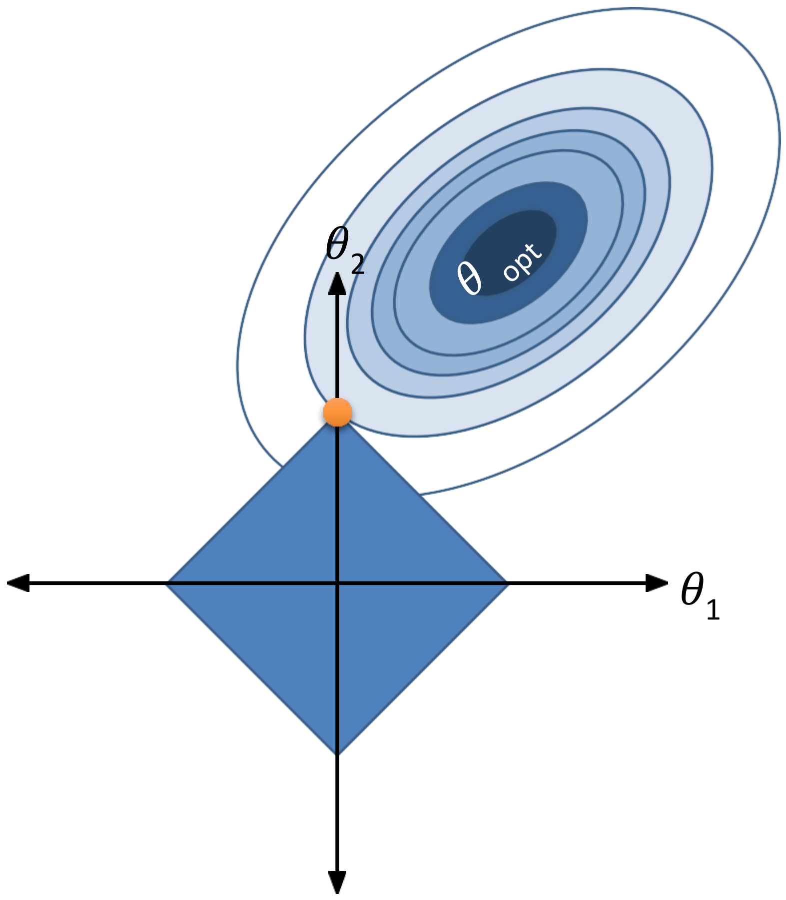
Ridge (L2-Reg) Regression $R_{\text{Ridge}}(\theta) = \sum_{i=1}^{d} \theta_i^2$

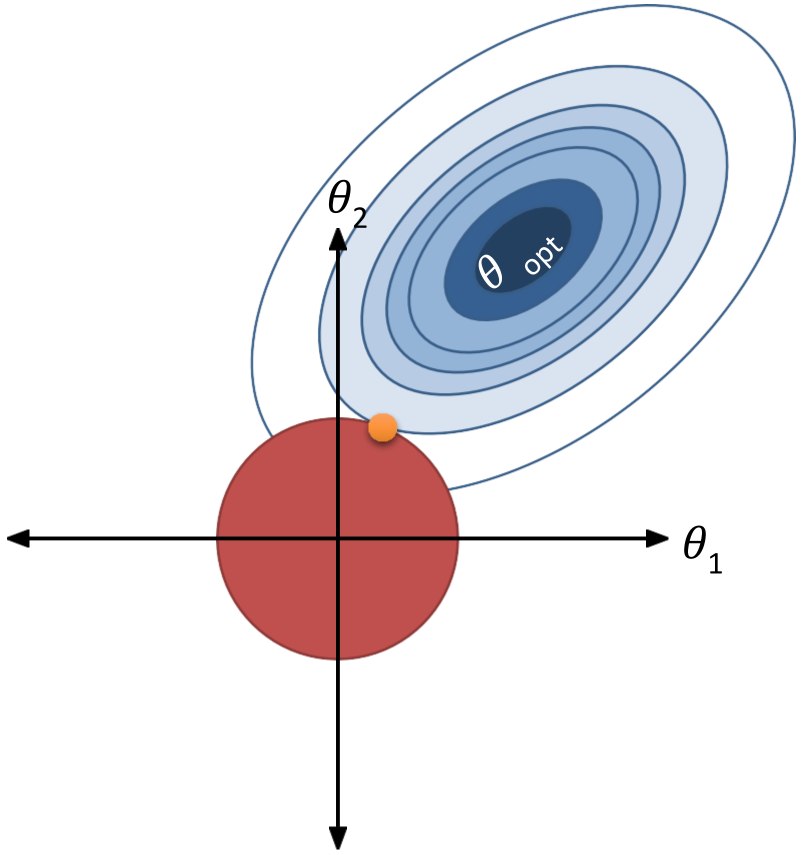
Lasso (L1-Reg) $R_{\text{Lasso}}(\theta) = \sum_{i=1}^{d} |\theta_i$

  - 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

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

  - Trivial solution $\rightarrow \lambda = 0$
- 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

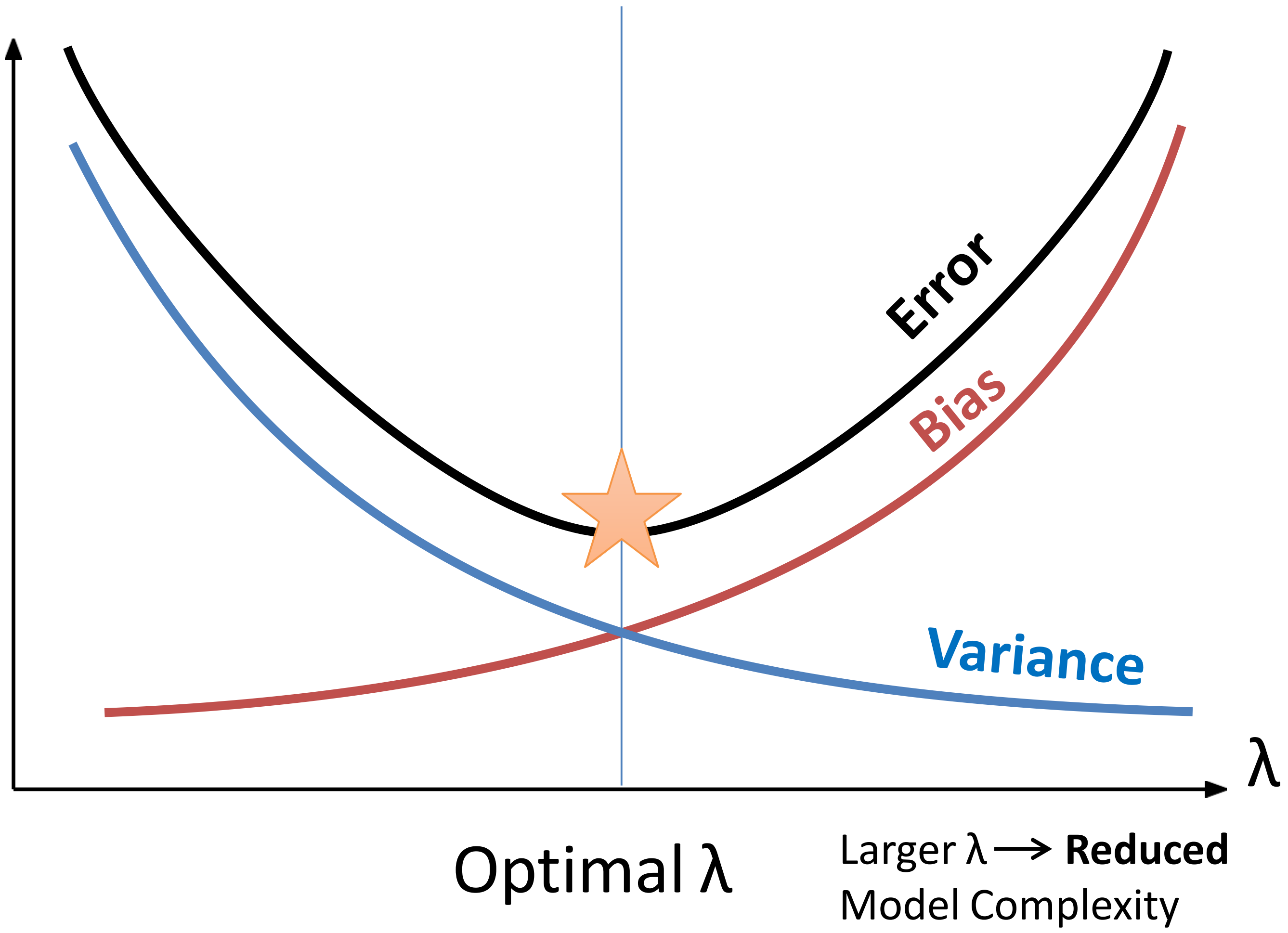
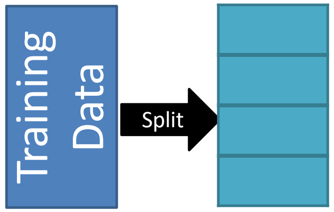
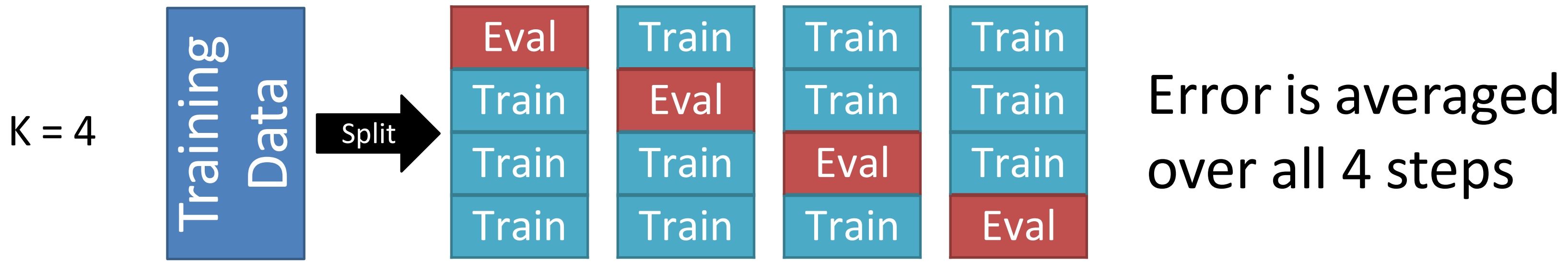


Image from

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:



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

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:

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

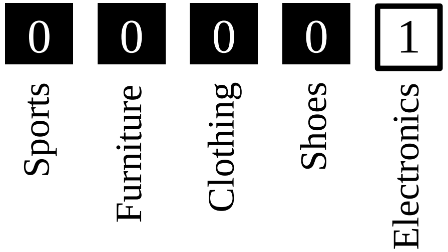
- Common of Regularization Functions:

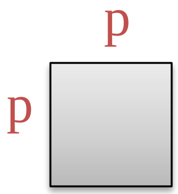
Ridge (L2-Reg) Regression $R_{\text{Ridge}}(\theta) = \sum_{i=1}^{d} \theta_i^2$ Lasso (L1-Reg) $R_{\text{Lasso}}(\theta) = \sum_{i=1}^{d} |\theta_i|$

  - Encourage small parameter values

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

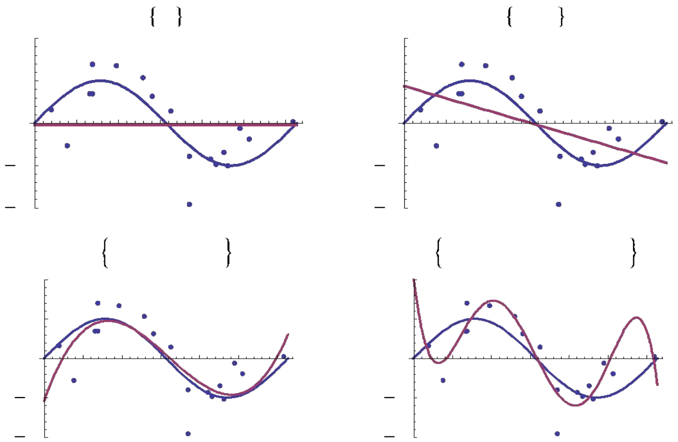
Summary of Regression

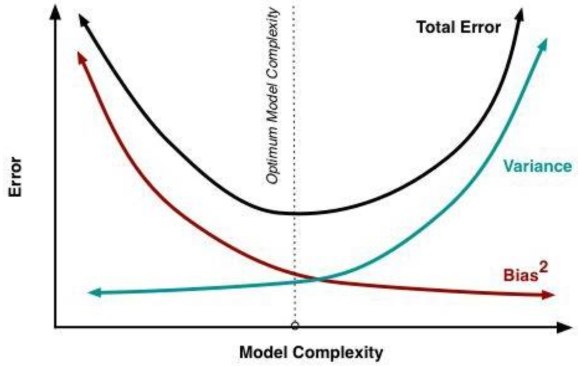


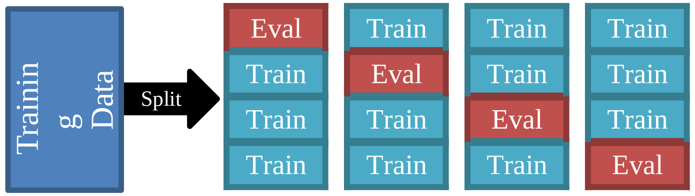


$$C = X^T X = \sum_{i=1}^{n} x_i x_i^T$$

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







Taxonomy

of Machine Learning

Labeled Data

Indirect (reward)

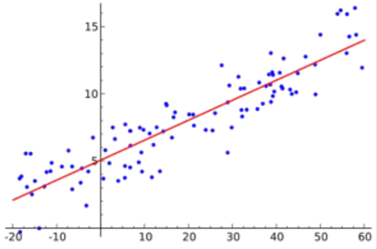
Unlabeled Data

Supervised Learning

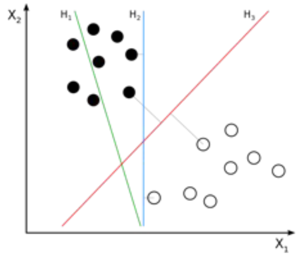
Reinforcement & Bandit Learning

Unsupervised Learning

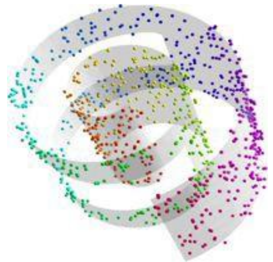
Regression



Classification



Dimensionality Reduction



Clustering