2018 EE448, Big Data Mining, Lecture 4

Supervised Learning
(Part I)

Weinan Zhang
Shanghai Jiao Tong University
http://wnzhang.net

http://wnzhang.net/teaching/ee448/index.html

Content of Coming Lectures

- Introduction to Machine Learning
- Linear Models
- Support Vector Machines
- Neural Networks
- Tree Models
- Ensemble Methods

Content of This Lecture

- Introduction to Machine Learning

- Linear Models

What is Machine Learning

- Learning

"Learning is any process by which a system improves performance from experience."

--- Herbert Simon

Turing Award (1975)

artificial intelligence, the psychology of human cognition

Nobel Prize in Economics (1978)

decision-making process within economic organizations

What is Machine Learning

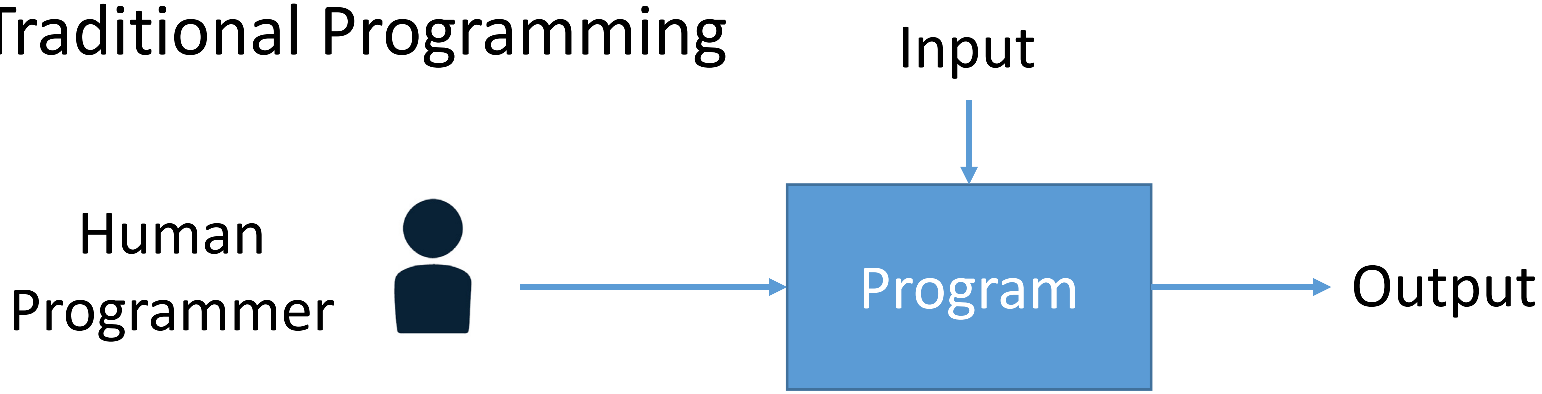
A more mathematical definition by Tom Mitchell

- Machine learning is the study of algorithms that
  - improvement their performance *P*
  - at some task *T*
  - based on experience *E*
  - with non-explicit programming

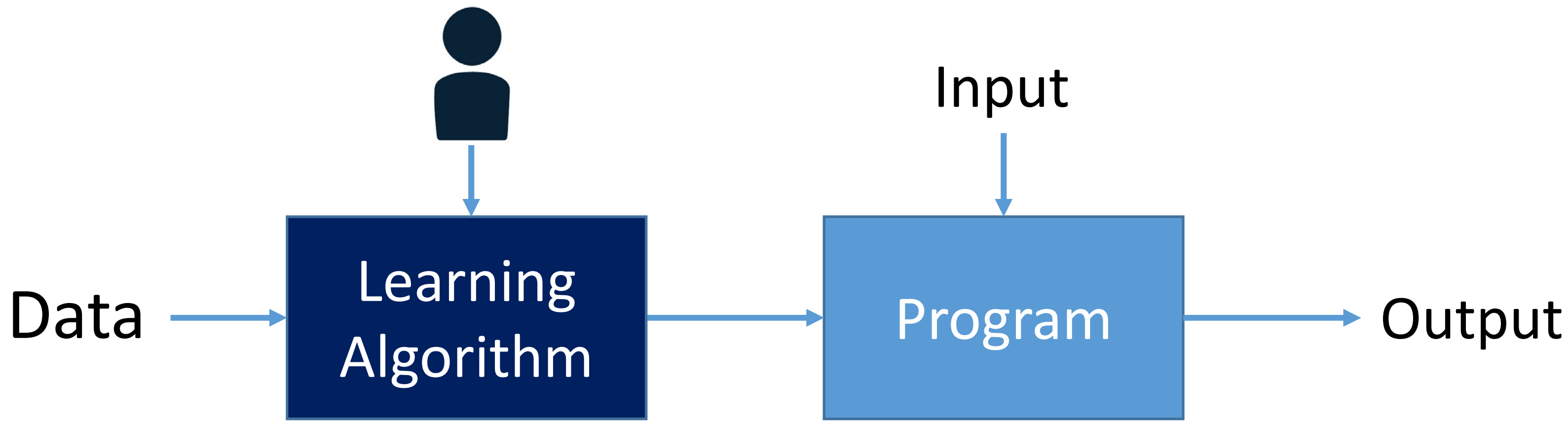
- A well-defined learning task is given by <*P, T, E*>

Programming vs. Machine Learning

- Traditional Programming



- Machine Learning



Slide credit: Feifei Li

When does ML Make Advantages

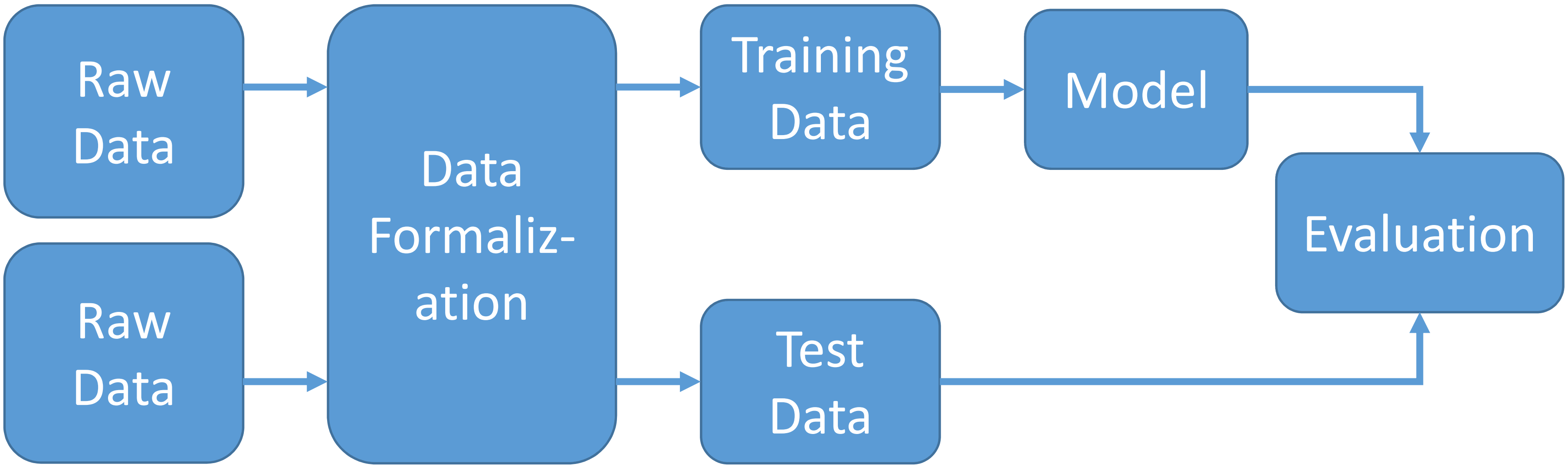
ML is used when

- Models are based on a huge amount of data
  - Examples: Google web search, Facebook news feed
- Output must be customized
  - Examples: News / item / ads recommendation
- Humans cannot explain the expertise
  - Examples: Speech / face recognition, game of Go
- Human expertise does not exist
  - Examples: Navigating on Mars

Machine Learning Categories

- Supervised Learning
  - To perform the desired output given the data and labels
- Unsupervised Learning
  - To analyze and make use of the underlying data patterns/structures
- Reinforcement Learning
  - To learn a policy of taking actions in a dynamic environment and acquire rewards

Machine Learning Process



- Basic assumption: there exist the same patterns across training and test data

Supervised Learning

- Given the training dataset of (data, label) pairs,

$$D = \{(x_i, y_i)\}_{i=1,2,\ldots,N}$$

  let the machine learn a function from data to label

$$y_i \simeq f_\theta(x_i)$$

- Function set $\{f_\theta(\cdot)\}$ is called hypothesis space
- Learning is referred to as updating the parameter θ
- How to learn?
  - Update the parameter to make the prediction close to the corresponding label
    - What is the learning objective?
    - How to update the parameters?

Learning Objective

- Make the prediction close to the corresponding label

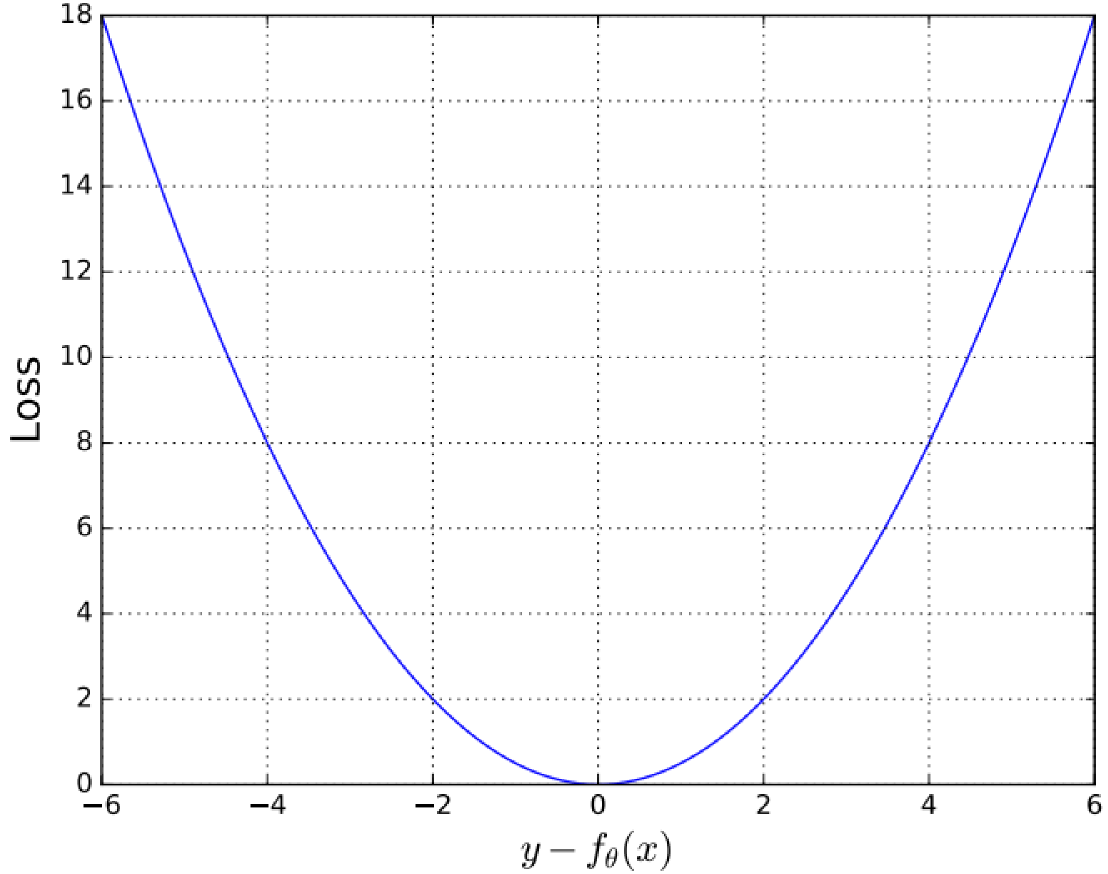
$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i))$$

- Loss function $\mathcal{L}(y_i, f_\theta(x_i))$ measures the error between the label and prediction
- The definition of loss function depends on the data and task
- Most popular loss function: squared loss

$$\mathcal{L}(y_i, f_\theta(x_i)) = \frac{1}{2}(y_i - f_\theta(x_i))^2$$

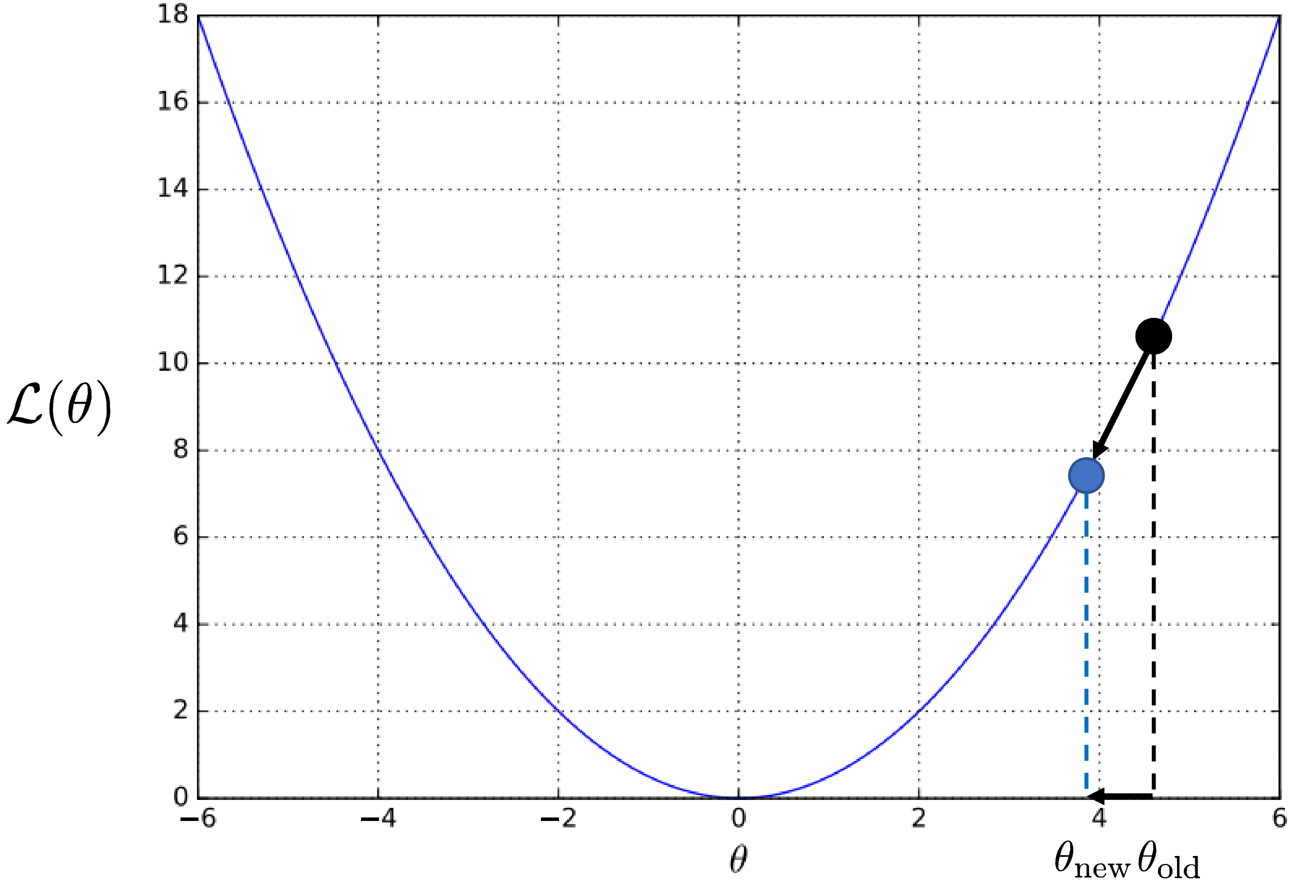
Squared Loss

$$\mathcal{L}(y_i, f_\theta(x_i)) = \frac{1}{2}(y_i - f_\theta(x_i))^2$$



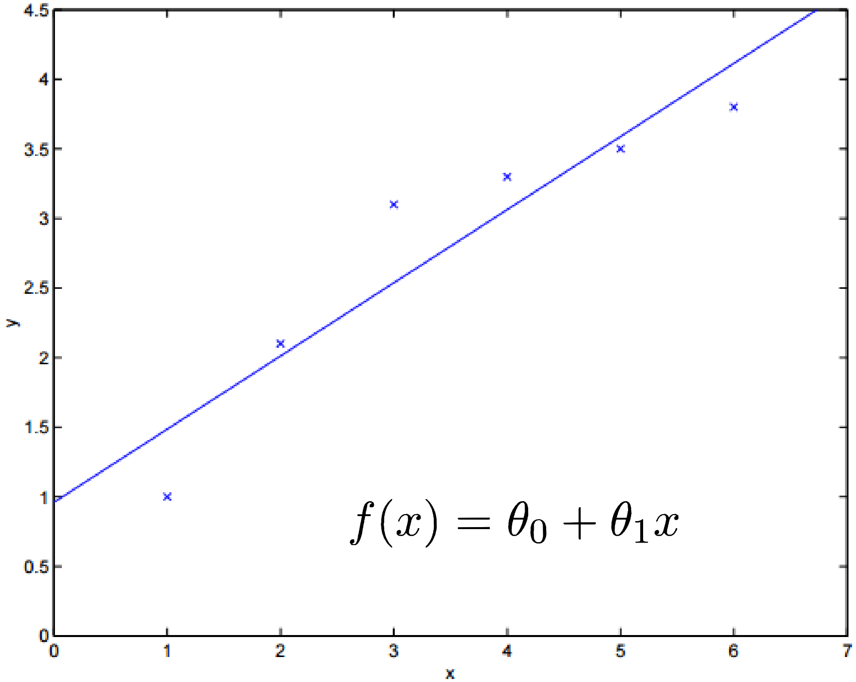
- Penalty much more on larger distances
- Accept small distance (error)
  - Observation noise etc.
  - Generalization

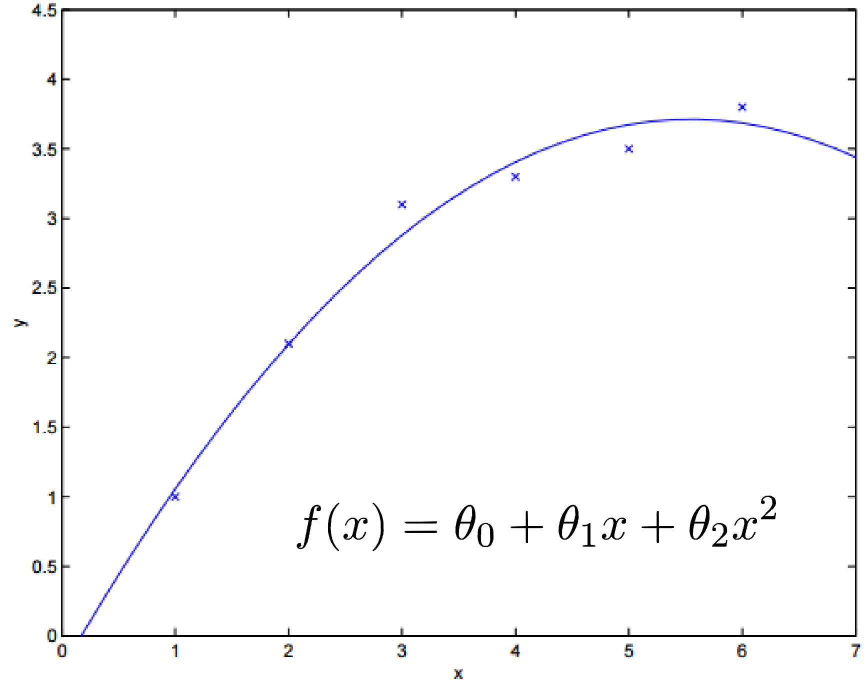
Gradient Learning Methods



$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

A Simple Example

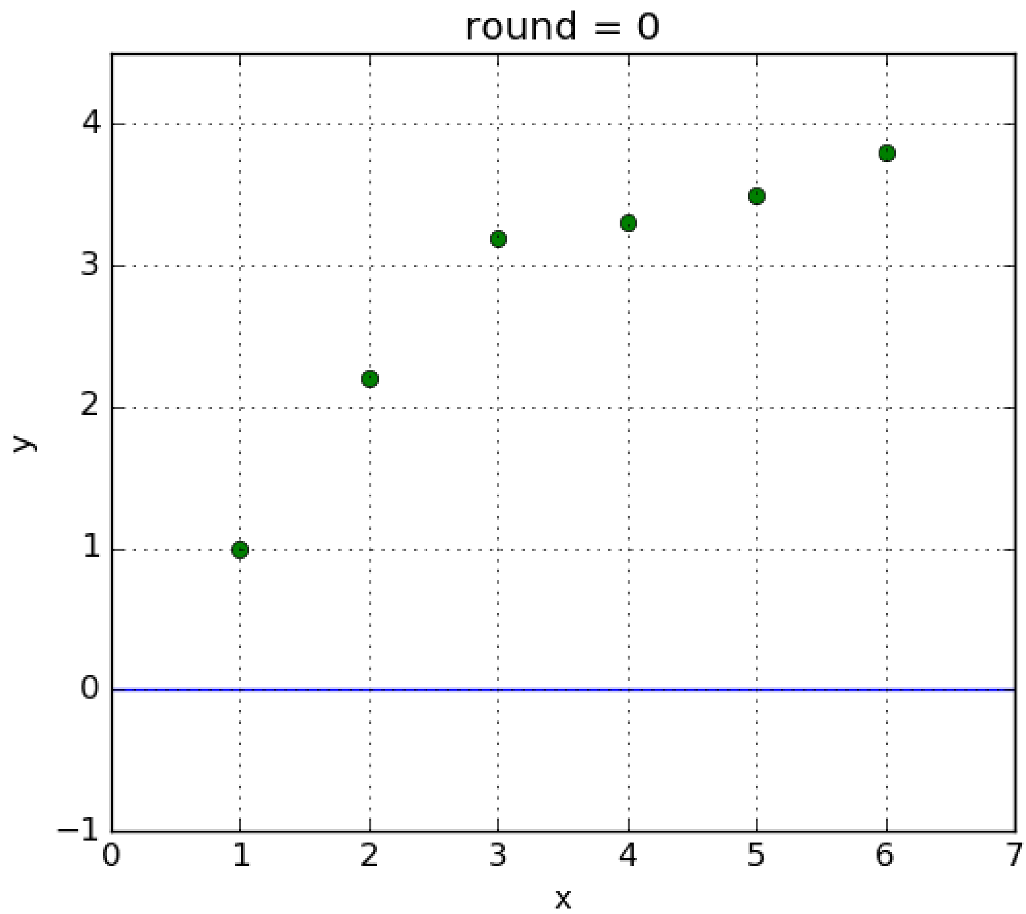




- Observing the data $\{(x_i, y_i)\}_{i=1,2,\ldots,N}$, we can use different models (hypothesis spaces) to learn
    - First, model selection (linear or quadratic)
    - Then, learn the parameters

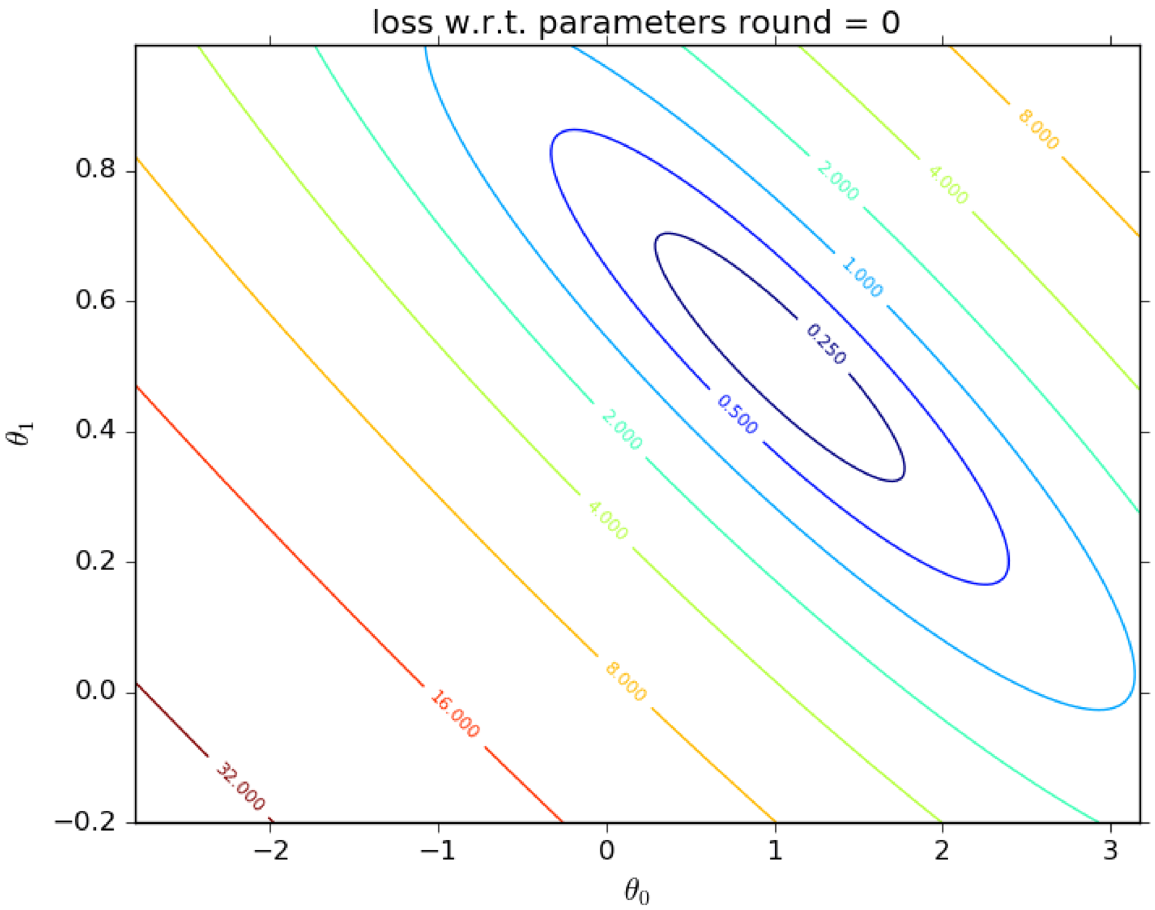
An example from Andrew Ng

Learning Linear Model - Curve

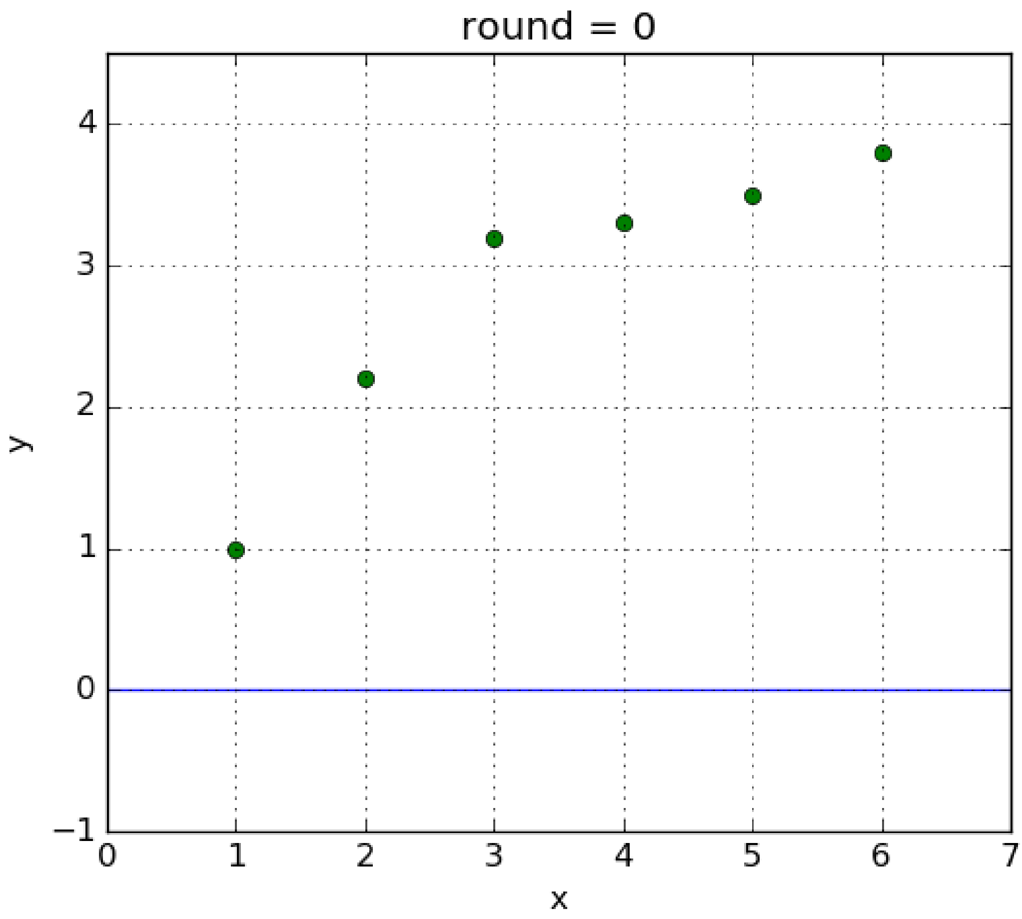


$$f(x) = \theta_0 + \theta_1 x$$

Learning Linear Model - Weights

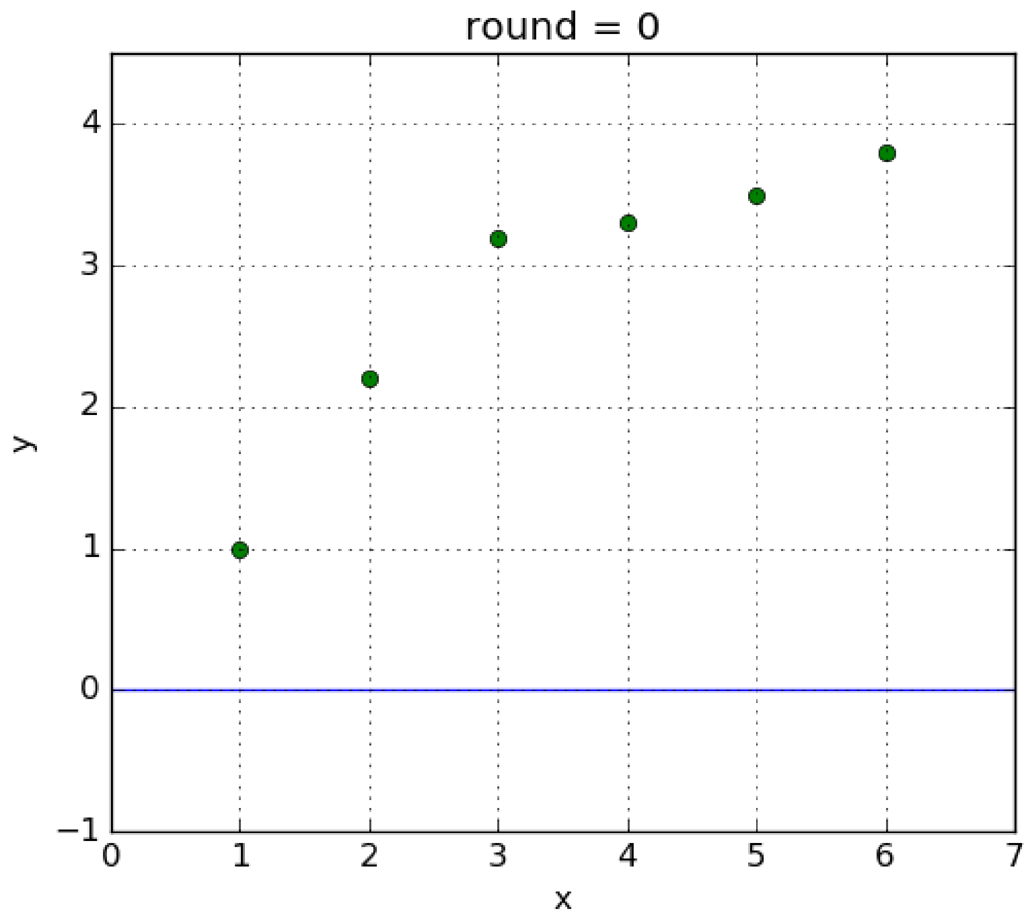


Learning Quadratic Model



$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

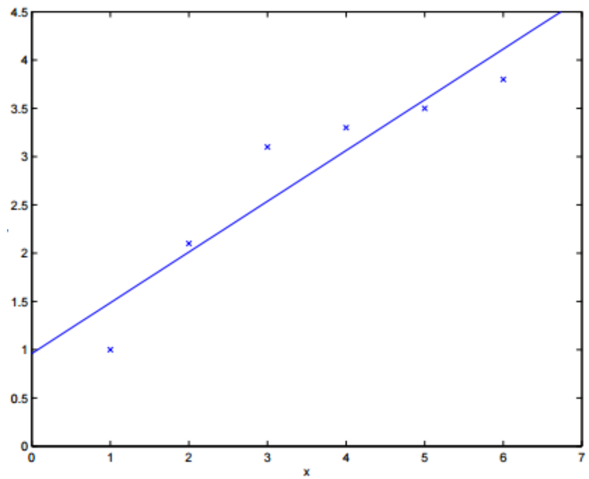
Learning Cubic Model



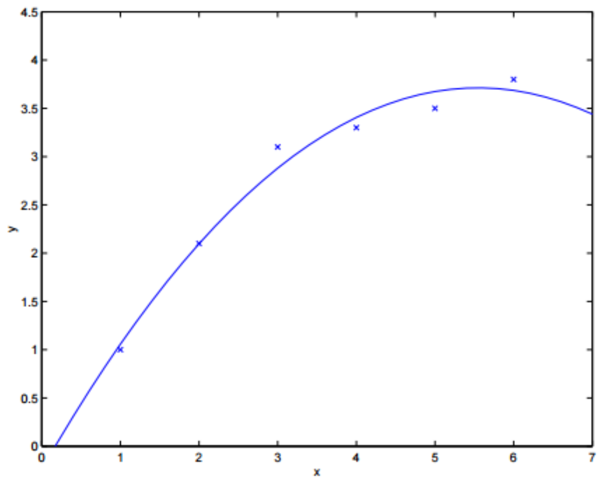
$$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

Model Selection

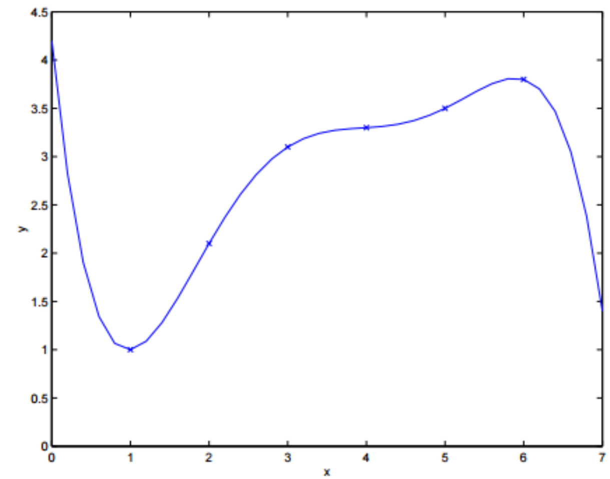
- Which model is the best?



Linear model: underfitting



Quadratic model: well fitting

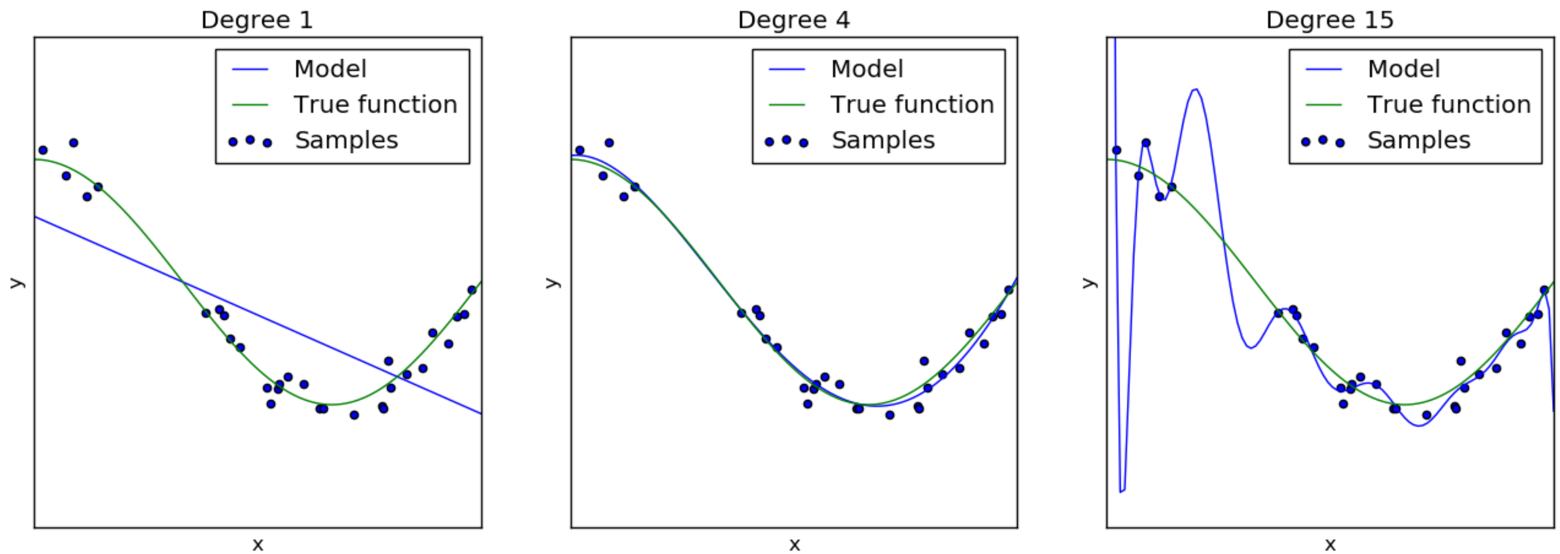


5^{th}-order model: overfitting

- Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data.
- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship

Model Selection

- Which model is the best?



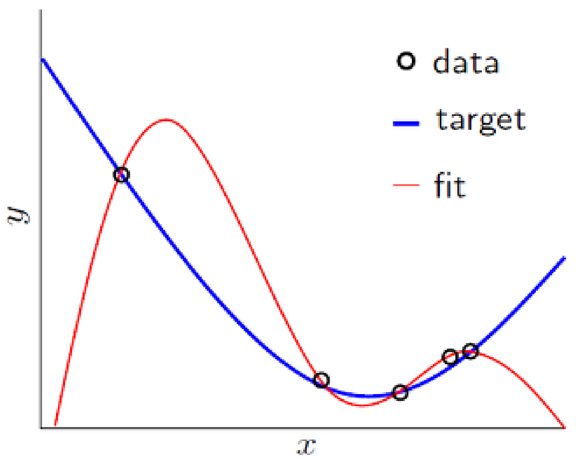
Linear model: underfitting | 4^{th}-order model: well fitting | 15^{th}-order model: overfitting

- Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data.
- Overfitting occurs when a statistical model describes random error or noise instead of the underlying relationship

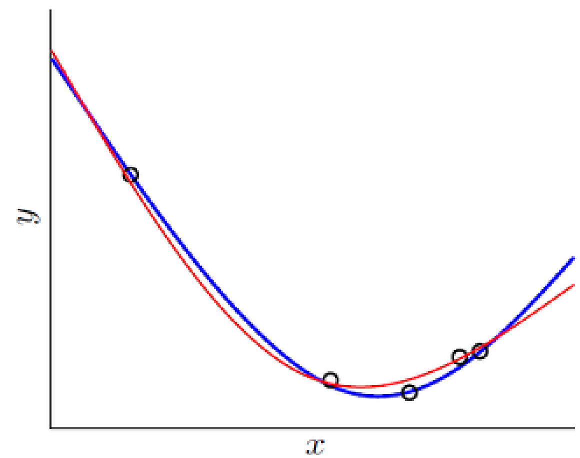
Regularization

- Add a penalty term of the parameters to prevent the model from overfitting the data

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i)) + \lambda\Omega(\theta)$$



(a) without regularization



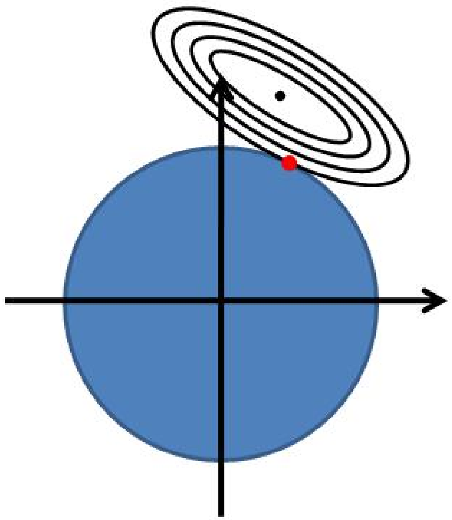
(b) with regularization

Typical Regularization

- L2-Norm (Ridge)

$$\Omega(\theta) = ||\theta||_2^2 = \sum_{m=1}^{M} \theta_m^2$$

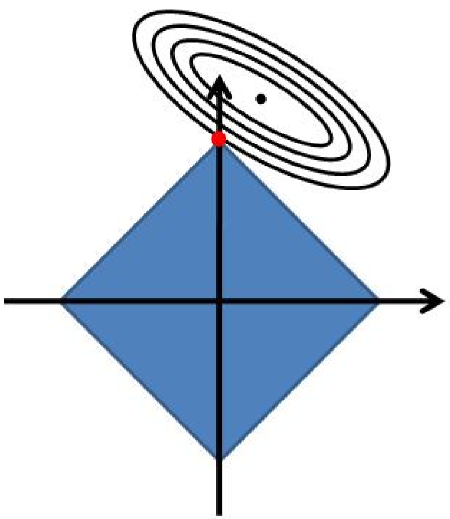
$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i)) + \lambda ||\theta||_2^2$$



- L1-Norm (LASSO)

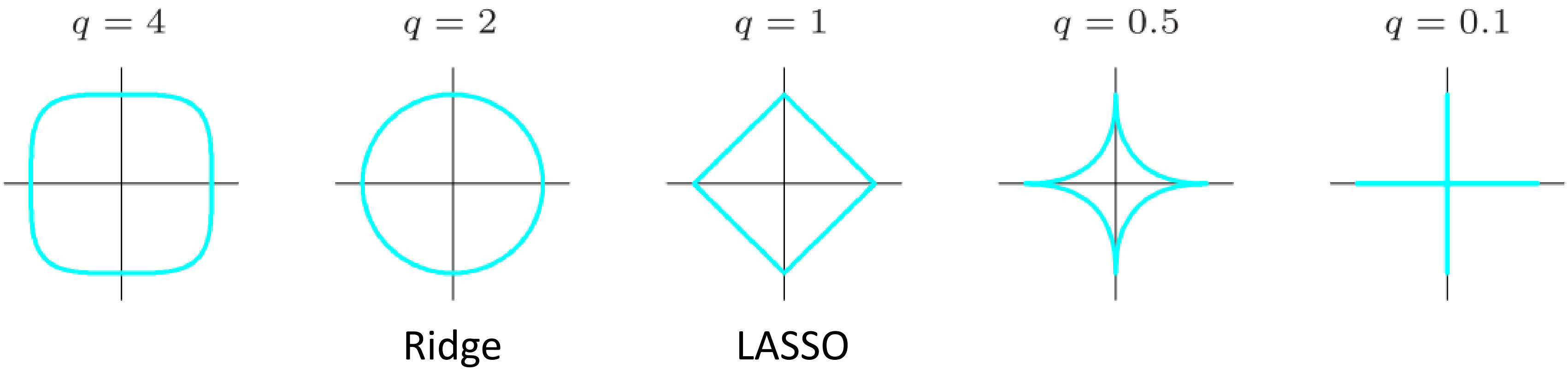
$$\Omega(\theta) = ||\theta||_1 = \sum_{m=1}^{M} |\theta_m|$$

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i)) + \lambda ||\theta||_1$$



More Normal-Form Regularization

- Contours of constant value of $\sum_j |\theta_j|^q$



- Sparse model learning with *q* not higher than 1
- Seldom use of *q* > 2
- Actually, 99% cases use *q* = 1 or 2

Principle of Occam's razor

Among competing hypotheses, the one with the fewest assumptions should be selected.

- Recall the function set $\{f_\theta(\cdot)\}$ is called hypothesis space

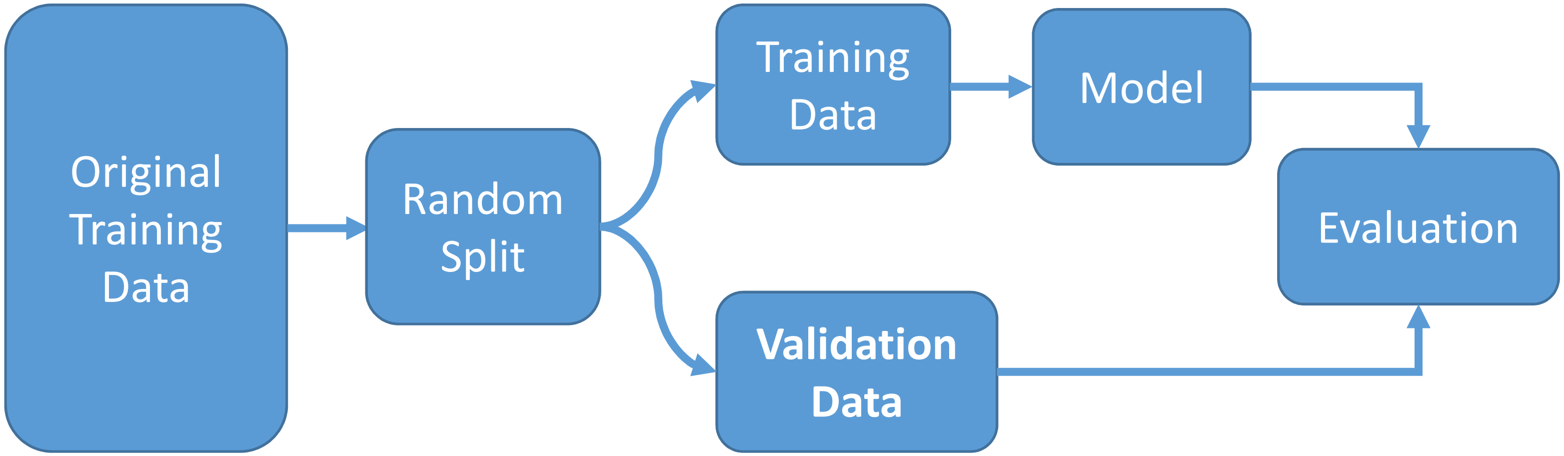
$$\min_\theta \underbrace{\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, f_\theta(x_i))}_{\text{Original loss}} + \lambda \underbrace{\Omega(\theta)}_{\text{Penalty on assumptions}}$$

Model Selection

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i)) + \lambda ||\theta||_2^2$$

- An ML solution has model parameters θ and optimization hyperparameters λ
- Hyperparameters
  - Define higher level concepts about the model such as complexity, or capacity to learn.
  - **Cannot be learned directly from the data** in the standard model training process and need to be predefined.
  - Can be decided by setting different values, training different models, and choosing the values that test better
- Model selection (or hyperparameter optimization) cares how to select the optimal hyperparameters.

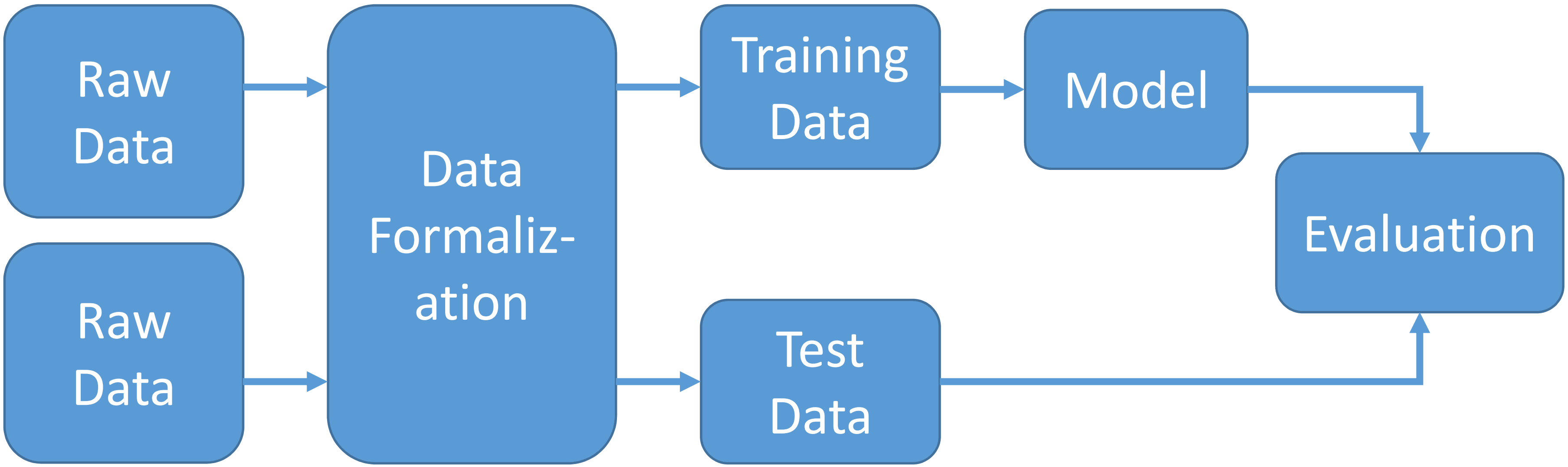
Cross Validation for Model Selection



K-fold Cross Validation

1. Set hyperparameters
2. For *K* times repeat:
   - Randomly split the original training data into training and validation datasets
   - Train the model on training data and evaluate it on validation data, leading to an evaluation score
3. Average the *K* evaluation scores as the model performance

Machine Learning Process



- After selecting ‘good’ hyperparameters, we train the model over the whole training data and the model can be used on test data.

Generalization Ability

- Generalization Ability is the model prediction capacity on unobserved data
  - Can be evaluated by Generalization Error, defined by

$$R(f) = \mathbb{E}[\mathcal{L}(Y, f(X))] = \int_{X \times Y} \mathcal{L}(y, f(x)) p(x, y) dx dy$$

  - where $p(x, y)$ is the underlying (probably unknown) joint data distribution

- Empirical estimation of GA on a training dataset is

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

A Simple Case Study on Generalization Error

- Finite hypothesis set $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$
- Theorem of generalization error bound:

  For any function $f \in \mathcal{F}$, with probability no less than $1 - \delta$, it satisfies

$$R(f) \leq \hat{R}(f) + \epsilon(d, N, \delta)$$

  where

$$\epsilon(d, N, \delta) = \sqrt{\frac{1}{2N}\Big(\log d + \log \frac{1}{\delta}\Big)}$$

  - *N*: number of training instances
  - *d:* number of functions in the hypothesis set

Section 1.7 in Dr. Hang Li's text book.

Content of This Lecture

- Introduction to Machine Learning

- Linear Models
    - Linear regression, linear classification, applications

Linear Regression

Linear Models for Supervised Learning

Linear Discriminative Models

- Discriminative model
  - modeling the dependence of unobserved variables on observed ones
  - also called conditional models.
  - **Deterministic**: $y = f_\theta(x)$
  - Probabilistic: $p_\theta(y|x)$

- Focus of this course
  - Linear regression model
  - Linear classification model

Linear Discriminative Models

- Discriminative model
  - modeling the dependence of unobserved variables on observed ones
  - also called conditional models.
  - **Deterministic**: $y = f_\theta(x)$
  - Probabilistic: $p_\theta(y|x)$

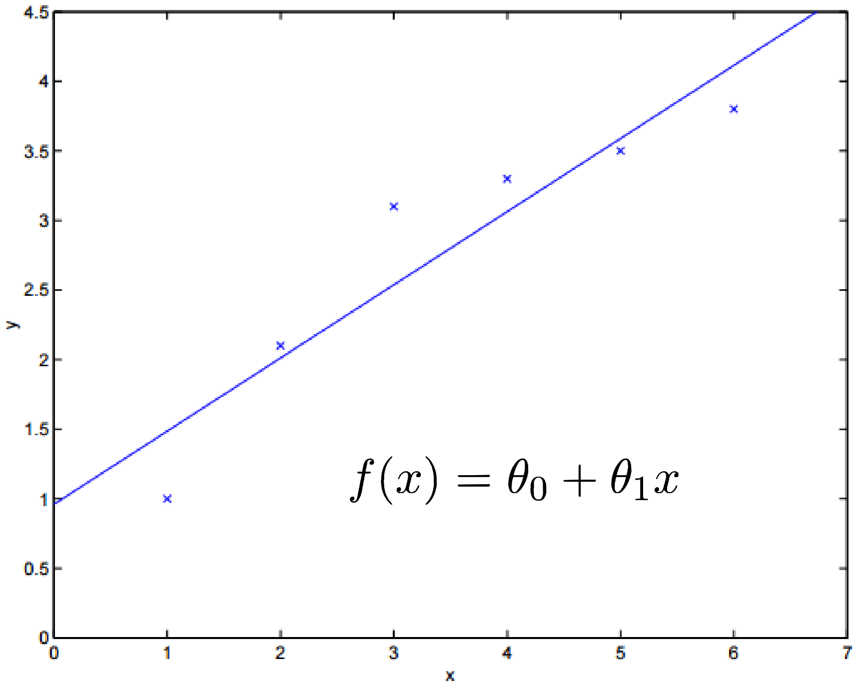
- Linear regression model

$$y = f_\theta(x) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j = \theta^\top x$$

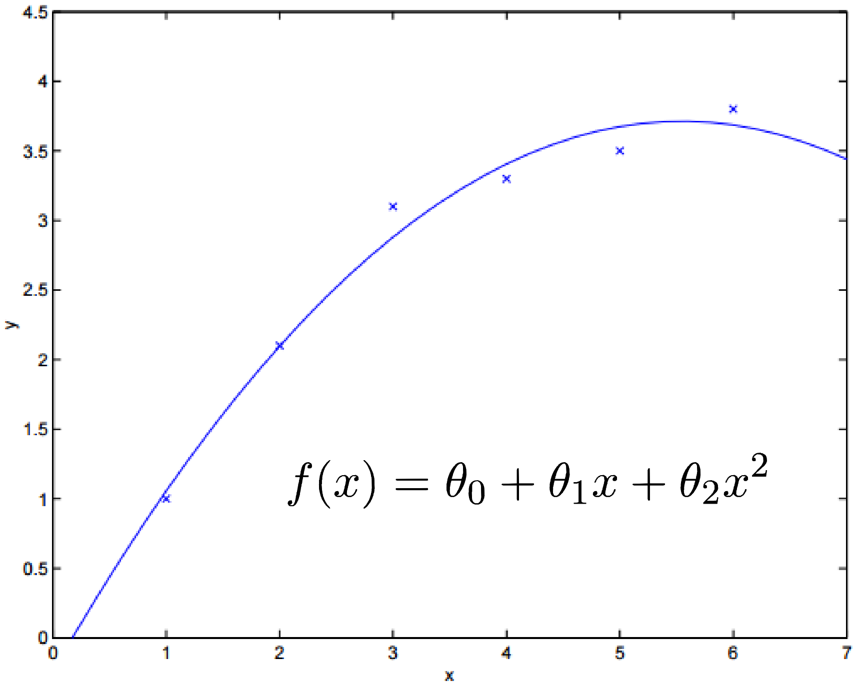
$$x = (1, x_1, x_2, \ldots, x_d)$$

Linear Regression

- One-dimensional linear & quadratic regression



Linear Regression

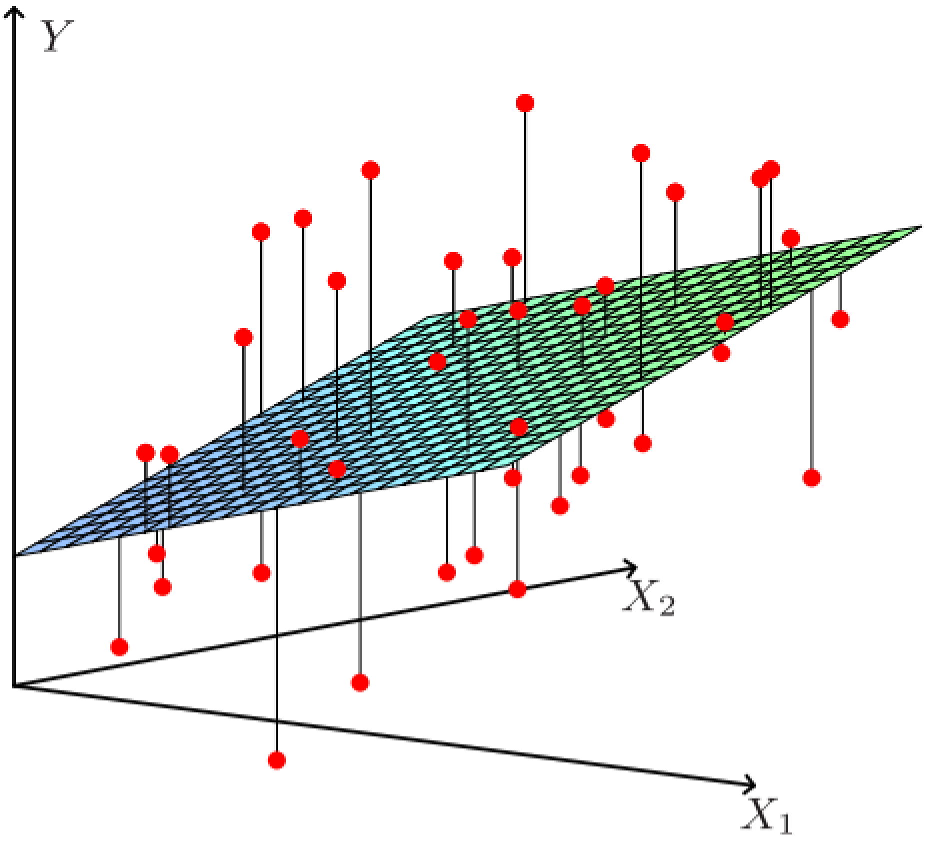


Quadratic Regression
(A kind of generalized
linear model)

Linear Regression

- Two-dimensional linear regression

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



Learning Objective

- Make the prediction close to the corresponding label

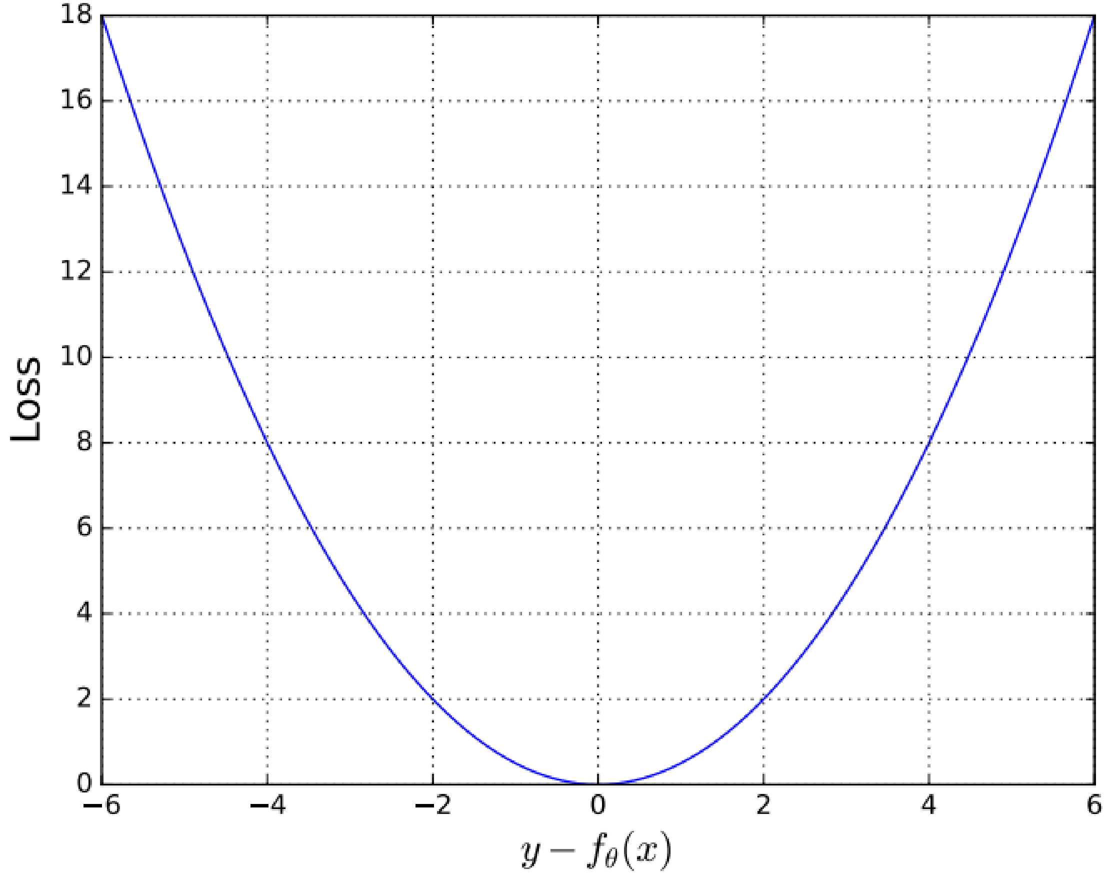
$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y_i, f_\theta(x_i))$$

- Loss function $\mathcal{L}(y_i, f_\theta(x_i))$ measures the error between the label and prediction
- The definition of loss function depends on the data and task
- Most popular loss function: squared loss

$$\mathcal{L}(y_i, f_\theta(x_i)) = (y_i - f_\theta(x_i))^2$$

Squared Loss

$$\mathcal{L}(y_i, f_\theta(x_i)) = \frac{1}{2}(y_i - f_\theta(x_i))^2$$

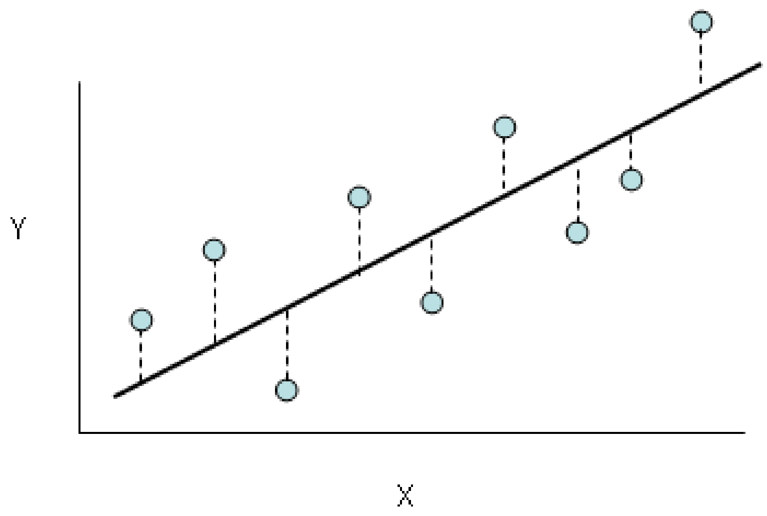


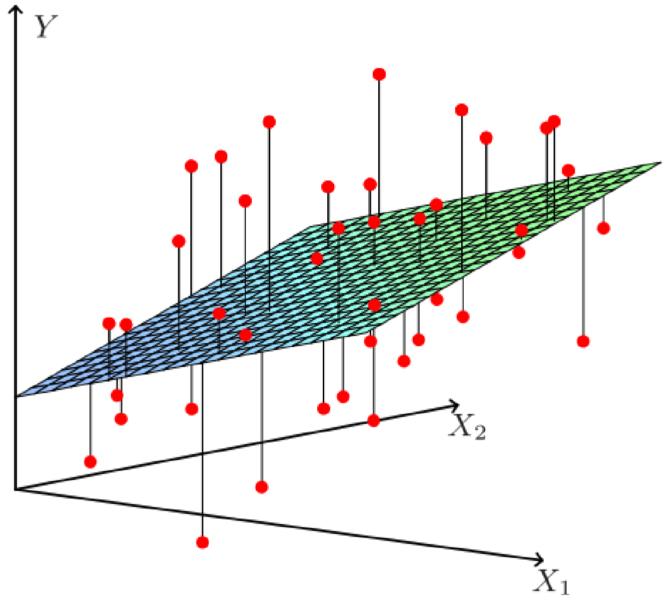
- Penalty much more on larger distances
- Accept small distance (error)
  - Observation noise etc.
  - Generalization

Least Square Linear Regression

- Objective function to minimize

$$J_\theta = \frac{1}{2N}\sum_{i=1}^{N}(y_i - f_\theta(x_i))^2 \qquad \min_\theta J_\theta$$

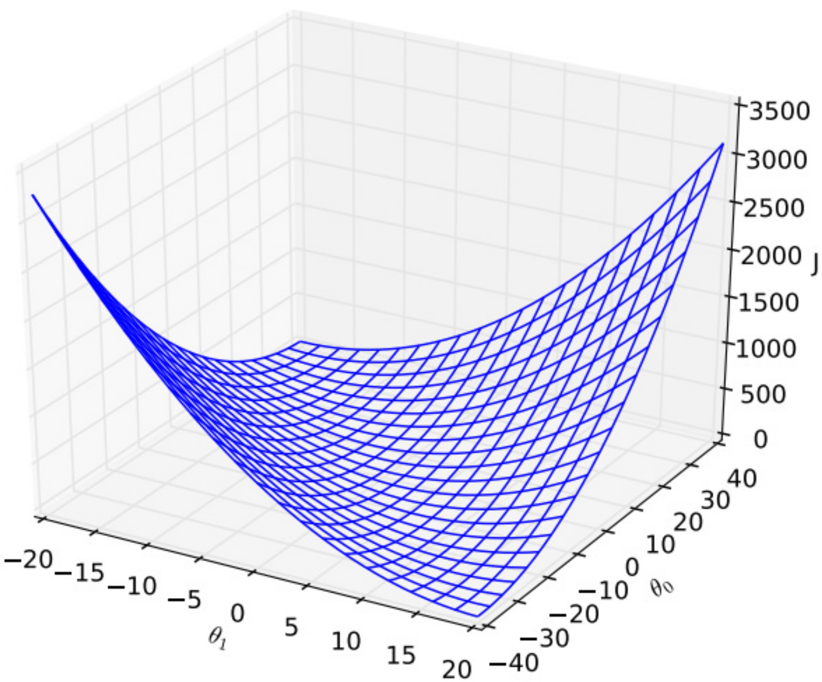


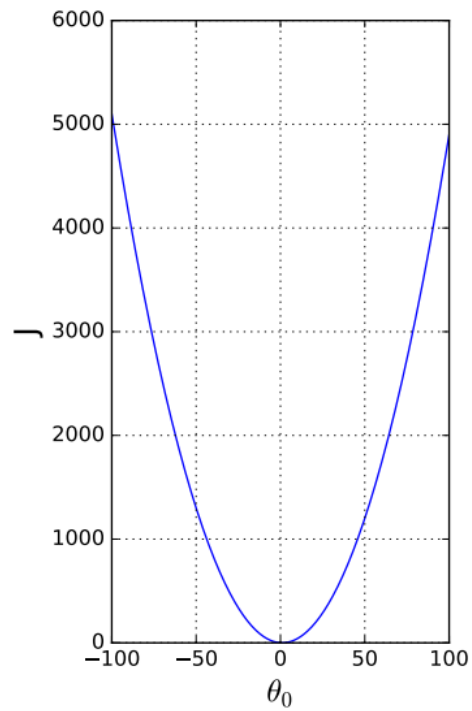


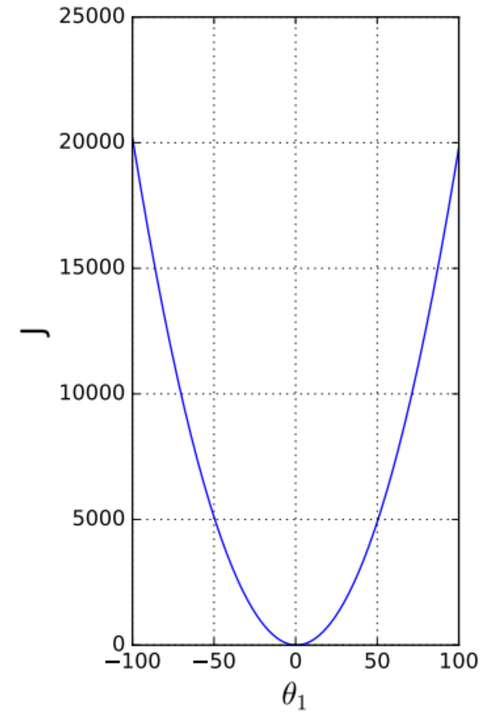
Minimize the Objective Function

- Let N=1 for a simple case, for (x,y)=(2,1)

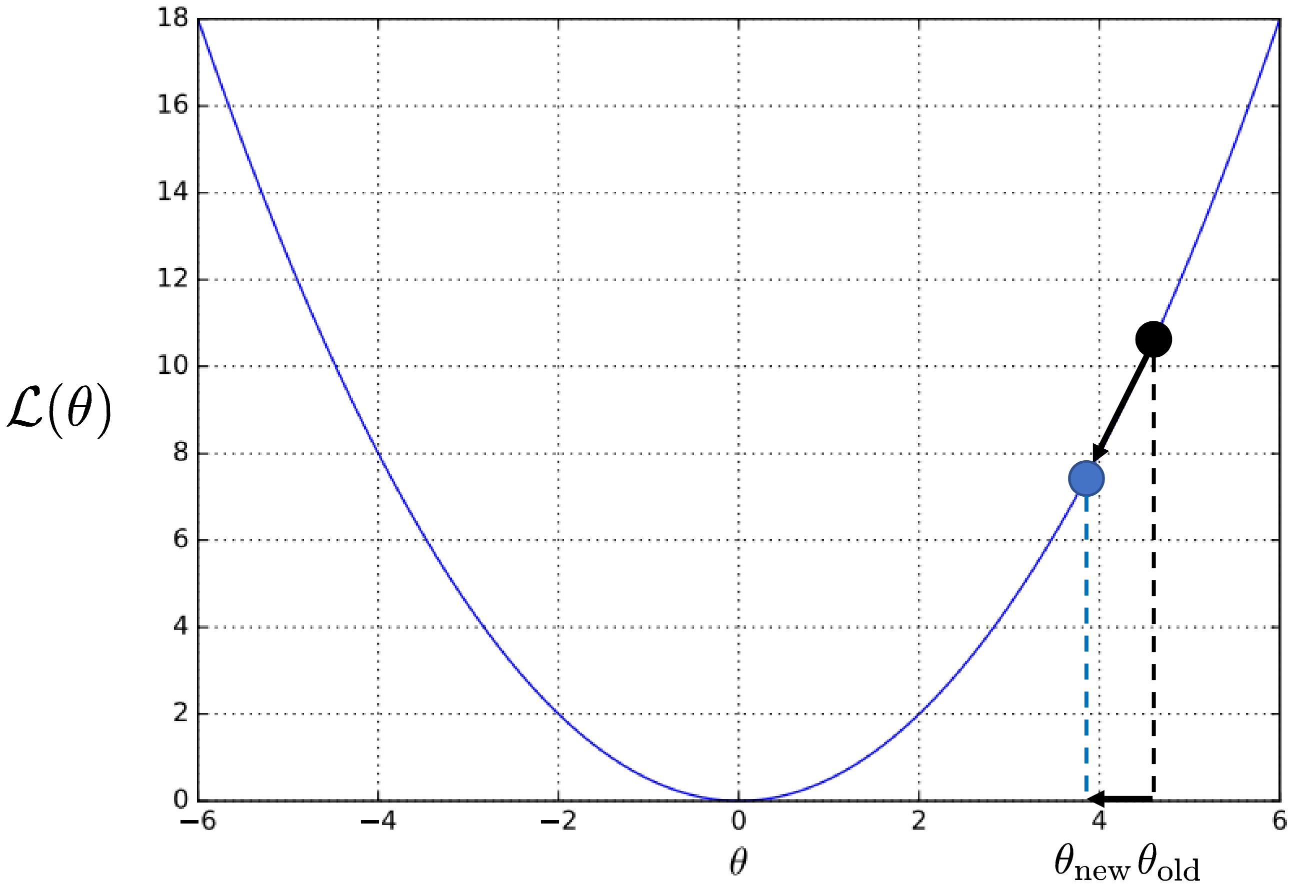
$$J(\theta) = \frac{1}{2}(y - \theta_0 - \theta_1 x)^2 = \frac{1}{2}(1 - \theta_0 - 2\theta_1)^2$$







Gradient Learning Methods



$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$

Batch Gradient Descent

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2 \qquad \min_\theta J(\theta)$$

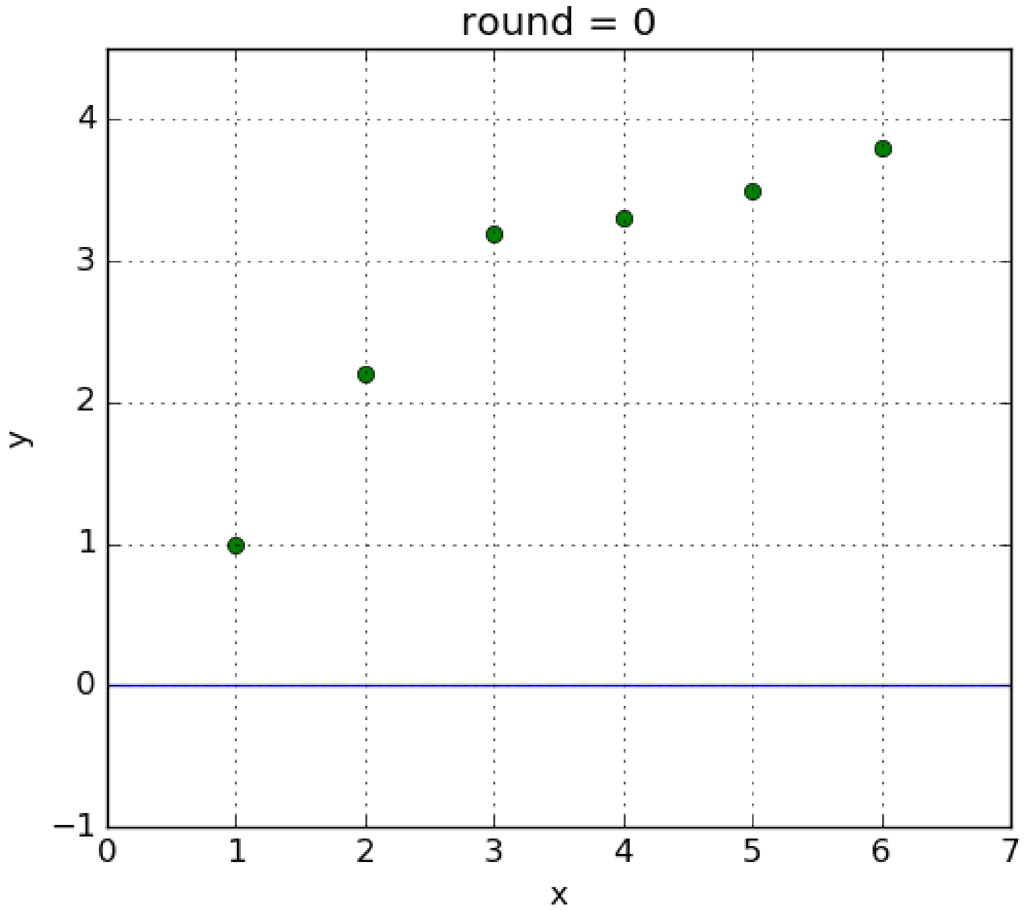
- Update $\theta_{\text{new}} \leftarrow \theta_{\text{old}} - \eta \frac{\partial J(\theta)}{\partial \theta}$ for the whole batch

$$\frac{\partial J(\theta)}{\partial \theta} = -\frac{1}{N} \sum_{i=1}^{N} (y_i - f_\theta(x_i)) \frac{\partial f_\theta(x_i)}{\partial \theta}$$

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

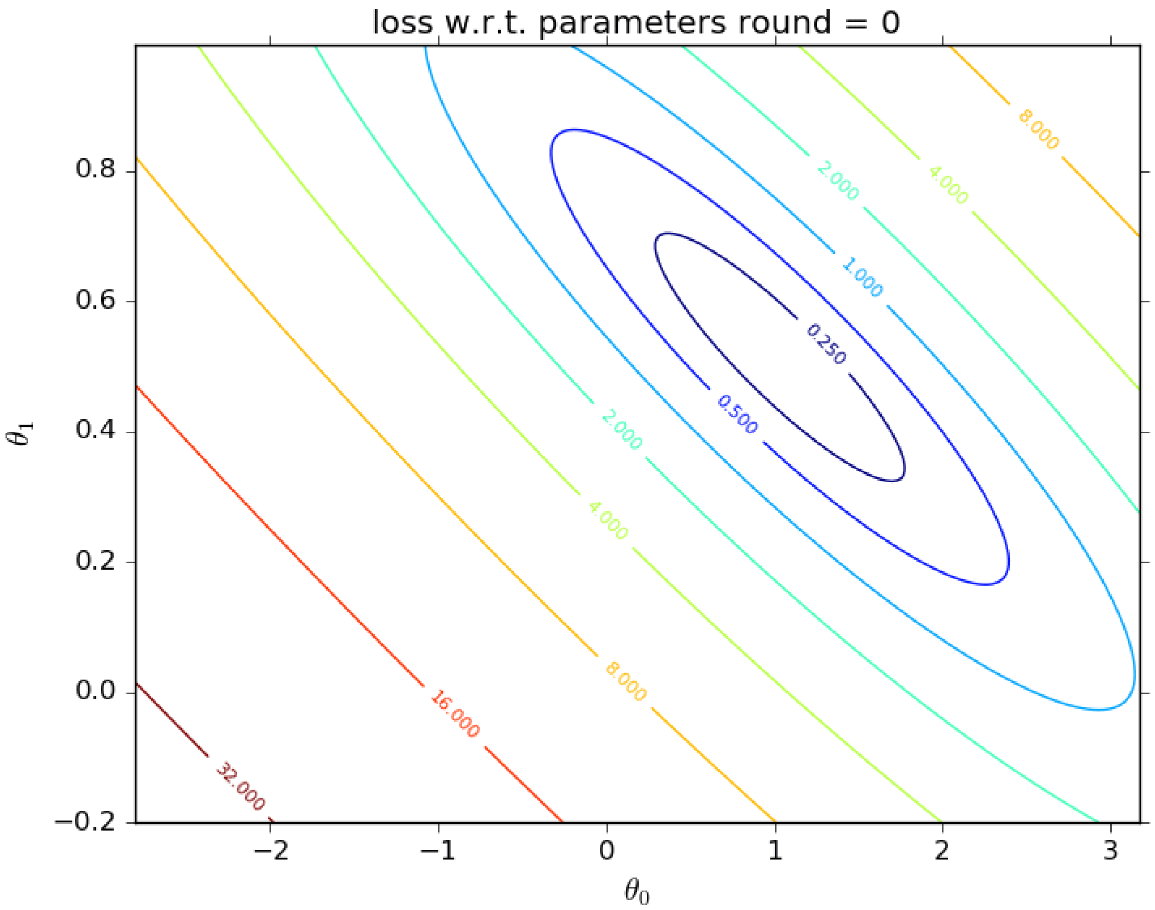
$$\theta_{\text{new}} = \theta_{\text{old}} + \eta \frac{1}{N} \sum_{i=1}^{N} (y_i - f_\theta(x_i)) x_i$$

Learning Linear Model - Curve



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

Learning Linear Model - Weights



Stochastic Gradient Descent

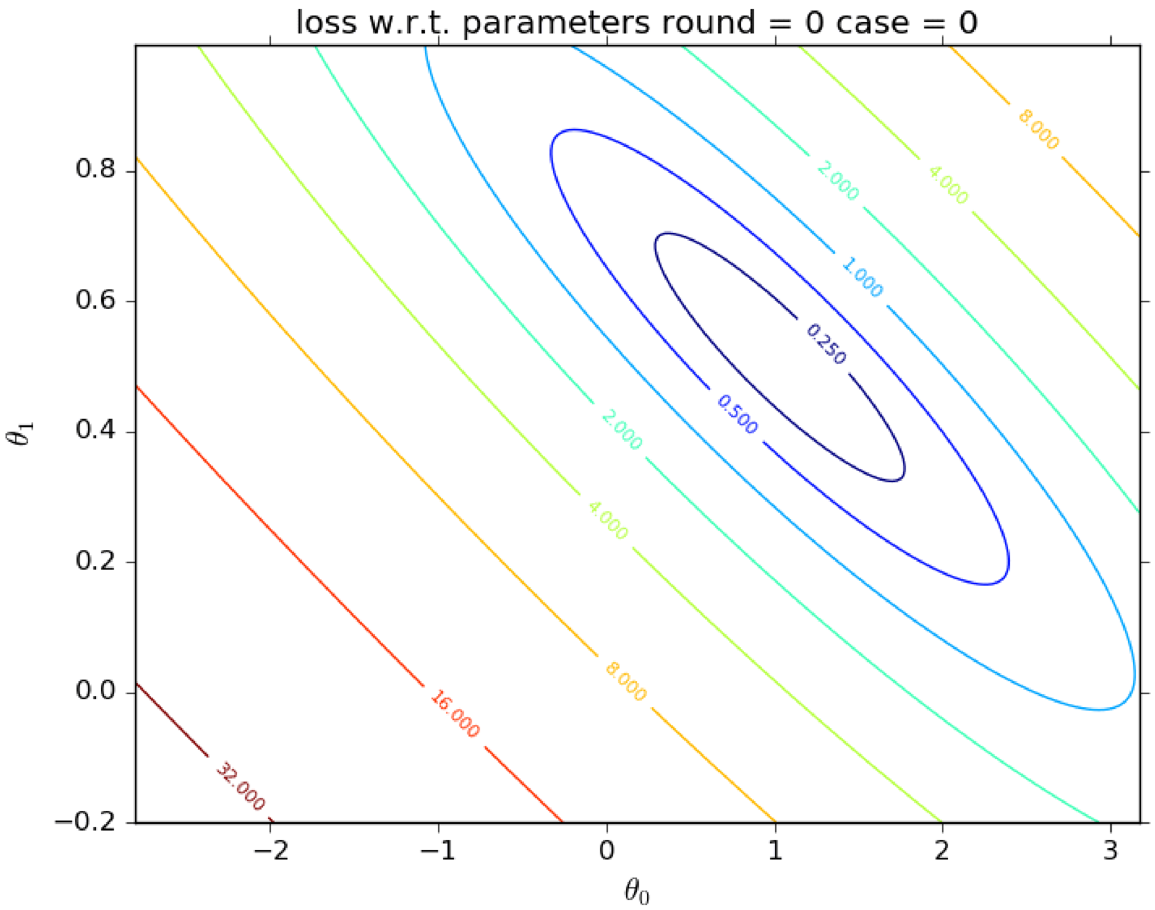
$$J^{(i)}(\theta) = \frac{1}{2}(y_i - f_\theta(x_i))^2 \qquad \min_\theta \frac{1}{N} \sum_i J^{(i)}(\theta)$$

- Update $\theta_{\text{new}} = \theta_{\text{old}} - \eta \frac{\partial J^{(i)}(\theta)}{\partial \theta}$ for every single instance

$$\begin{aligned}
\frac{\partial J^{(i)}(\theta)}{\partial \theta} &= -(y_i - f_\theta(x_i)) \frac{\partial f_\theta(x_i)}{\partial \theta} \\
&= -(y_i - f_\theta(x_i)) x_i \\
\theta_{\text{new}} &= \theta_{\text{old}} + \eta (y_i - f_\theta(x_i)) x_i
\end{aligned}$$

- Compare with BGD
  - Faster learning
  - Uncertainty or fluctuation in learning

Linear Classification Model



Mini-Batch Gradient Descent

- A combination of batch GD and stochastic GD

- Split the whole dataset into *K* mini-batches

$$\{1, 2, 3, \ldots, K\}$$

- For each mini-batch *k*, perform one-step BGD toward minimizing

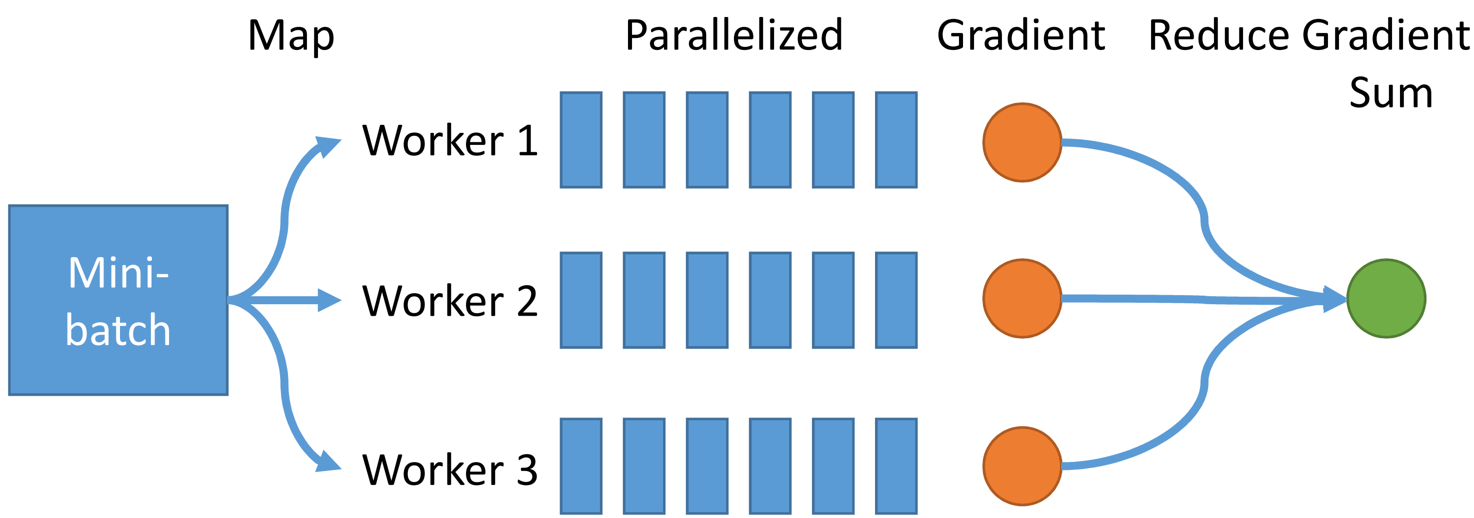
$$J^{(k)}(\theta) = \frac{1}{2N_k} \sum_{i=1}^{N_k} (y_i - f_\theta(x_i))^2$$

- Update $\theta_{\text{new}} = \theta_{\text{old}} - \eta \frac{\partial J^{(k)}(\theta)}{\partial \theta}$ for each mini-batch

Mini-Batch Gradient Descent

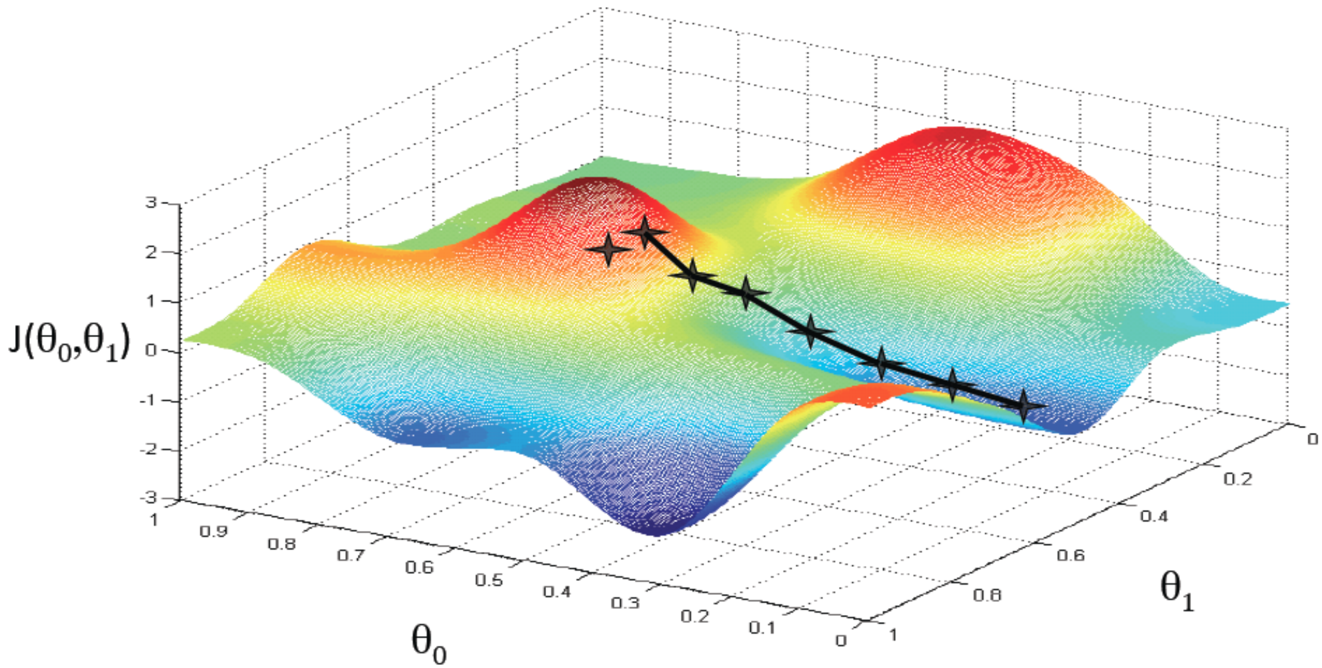
- Good learning stability (BGD)
- Good convergence rate (SGD)

- Easy to be parallelized
  - Parallelization within a mini-batch



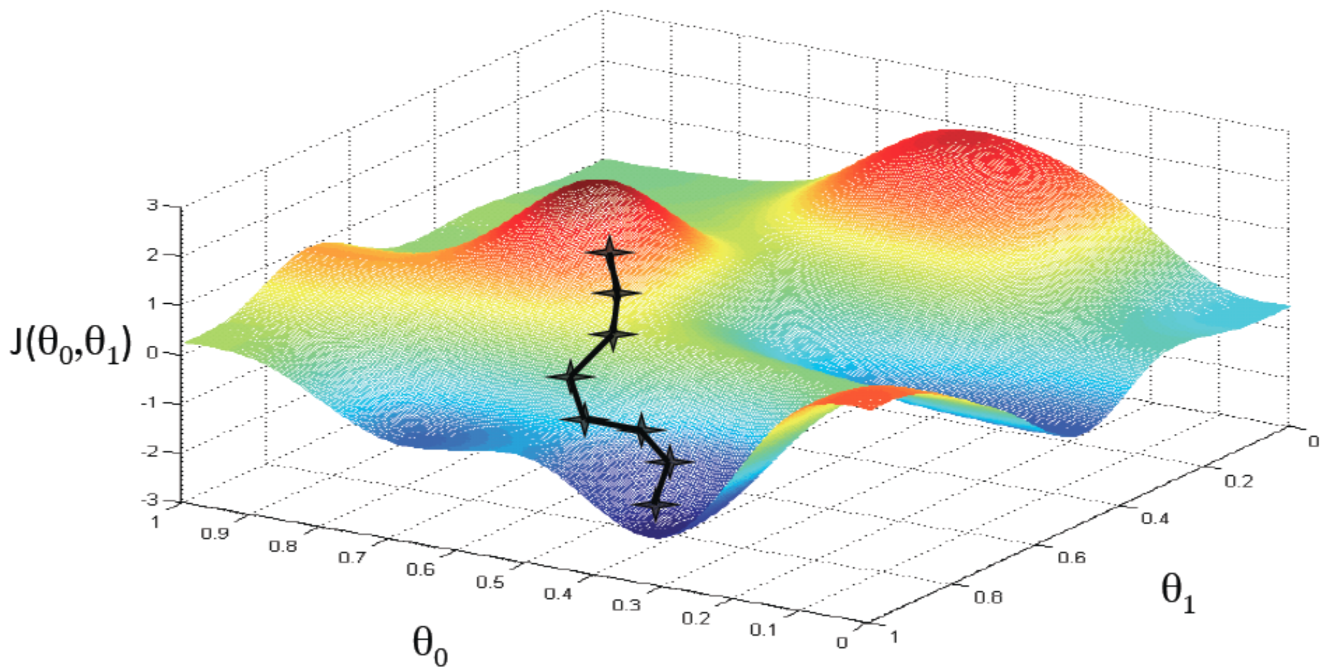
Basic Search Procedure

- Choose an initial value for θ
- Update θ iteratively with the data
- Until we research a minimum

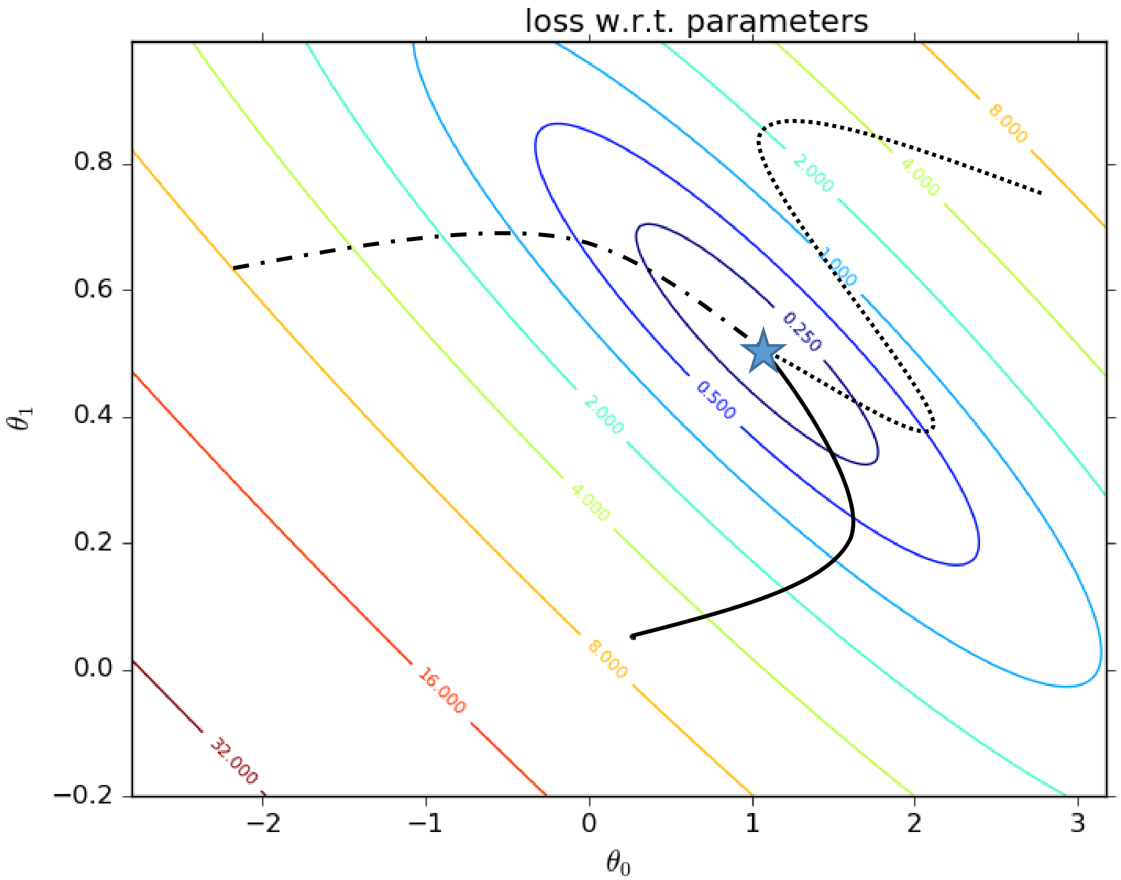


Basic Search Procedure

- Choose a new initial value for θ
- Update θ iteratively with the data
- Until we research a minimum



Unique Minimum for Convex Objective



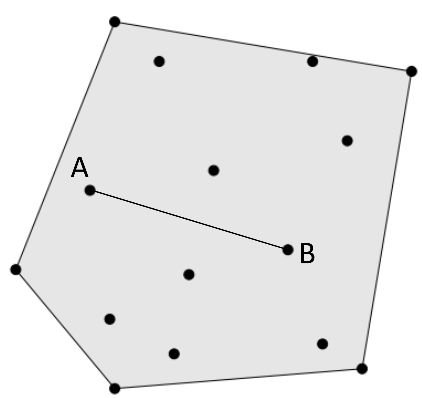
- Different initial parameters and different learning algorithm lead to the same optimum

Convex Set

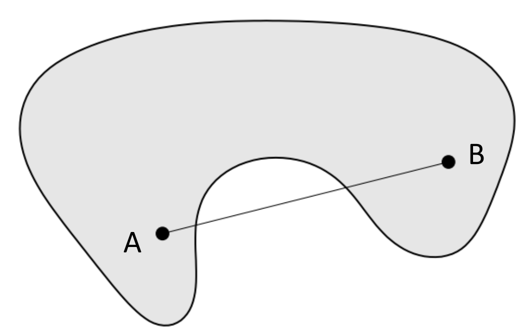
- A convex set *S* is a set of points such that, given any two points A, B in that set, the line AB joining them lies entirely within *S*.

$$tx_1 + (1-t)x_2 \in S$$

for all $x_1, x_2 \in S, 0 \leq t \leq 1$



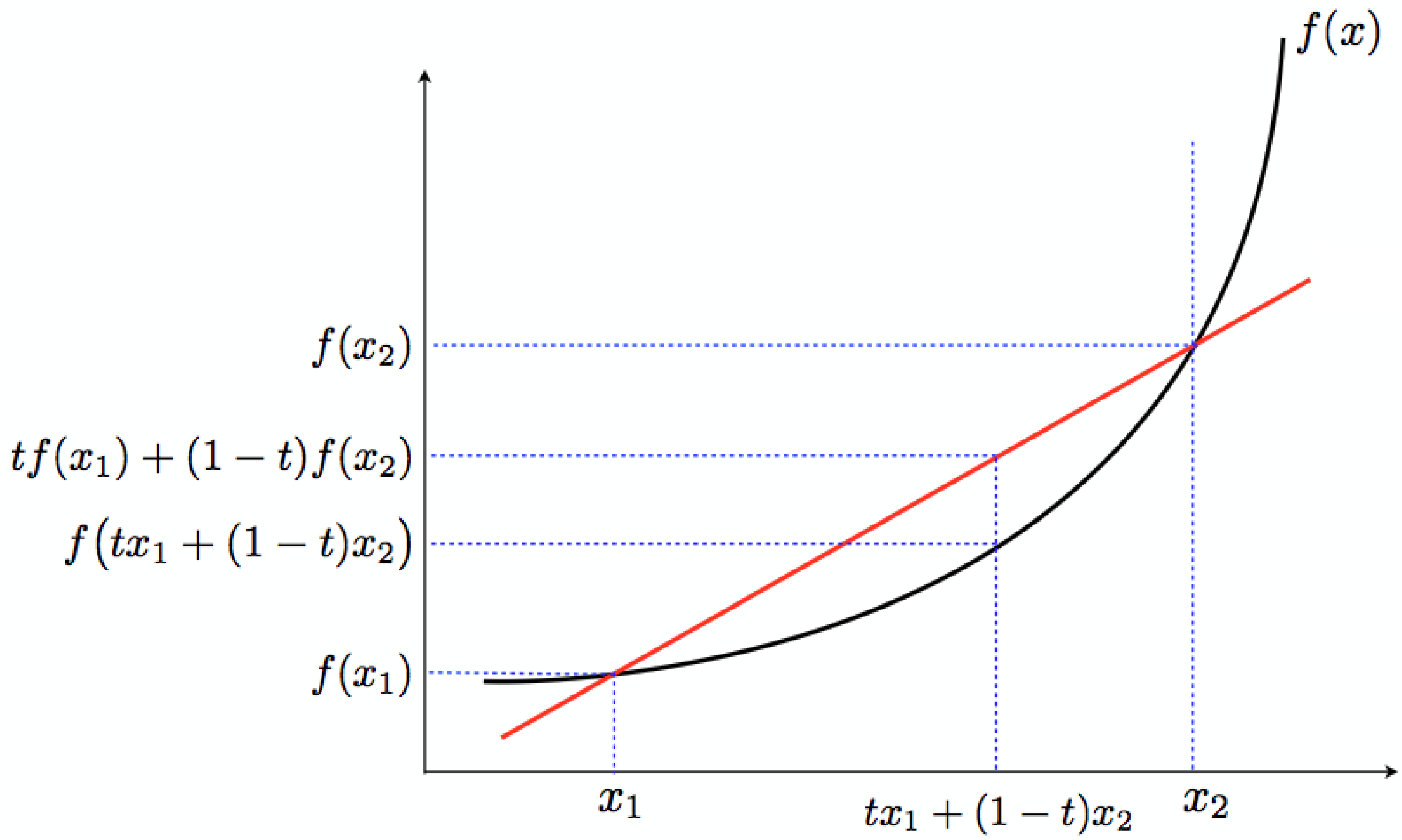
Convex set



Non-convex set

[Boyd, Stephen, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.]

Convex Function



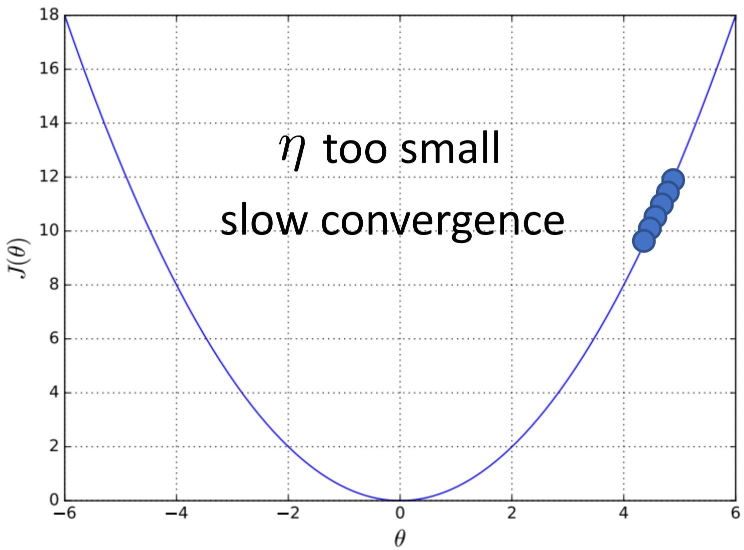
$f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathbf{dom}\ f$ is a convex set and

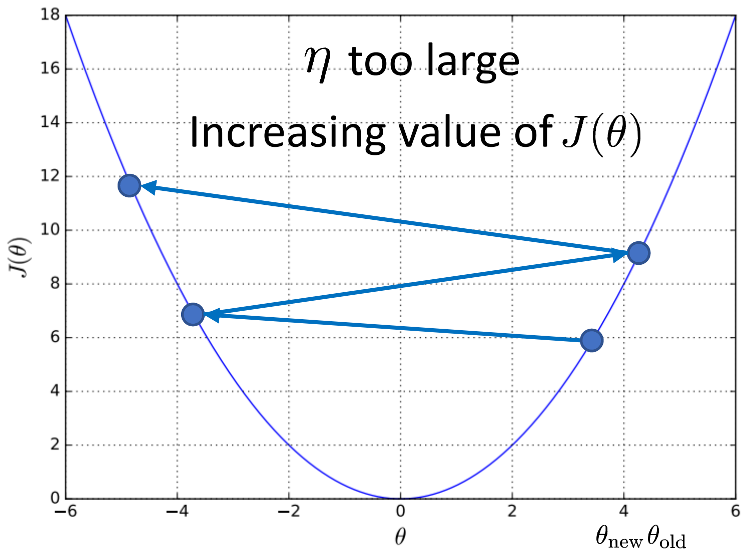
$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2)$$

for all $x_1, x_2 \in \mathbf{dom}\ f, 0 \le t \le 1$

Choosing Learning Rate

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta \frac{\partial J(\theta)}{\partial \theta}$$





- May overshoot the minimum
- May fail to converge
- May even diverge

- To see if gradient descent is working, print out $J(\theta)$ for each or every several iterations. If $J(\theta)$ does not drop properly, adjust η

Slide credit Eric Eaton

Algebra Perspective

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(n)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & x_3^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- Prediction $\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{x}^{(1)}\boldsymbol{\theta} \\ \boldsymbol{x}^{(2)}\boldsymbol{\theta} \\ \vdots \\ \boldsymbol{x}^{(n)}\boldsymbol{\theta} \end{bmatrix}$

- Objective $J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^\top(\boldsymbol{y} - \hat{\boldsymbol{y}}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})$

Matrix Form

- Objective

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \qquad \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Gradient

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})$$

- Solution

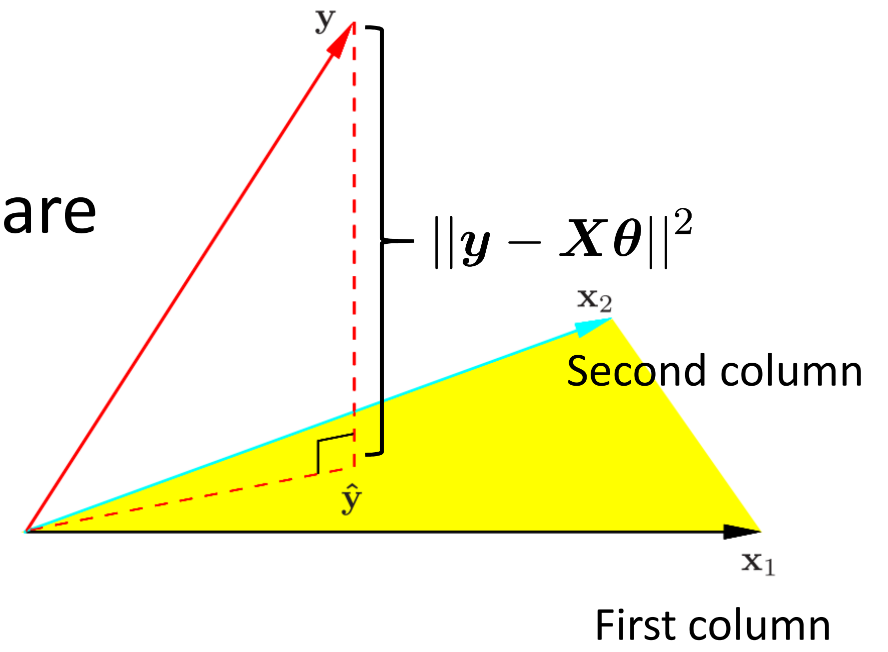
$$\begin{aligned}\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \boldsymbol{0} \;\Rightarrow\;& \boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) = \boldsymbol{0} \\ \Rightarrow\;& \boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta} \\ \Rightarrow\;& \hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}\end{aligned}$$

Matrix Form

- Then the predicted values are

$$\hat{\boldsymbol{y}} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$$
$$= \boldsymbol{H}\boldsymbol{y}$$

H: hat matrix



- Geometrical Explanation
  - The column vectors $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d]$ form a subspace of $\mathbb{R}^N$
  - *H* is a least square projection

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \ldots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \ldots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & x_3^{(n)} & \ldots & x_d^{(n)} \end{bmatrix} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_d] \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$\boldsymbol{X}^\top \boldsymbol{X}$ Might be Singular

- When some column vectors are not independent
  - For example, $\mathbf{x}_2 = 3\mathbf{x}_1$

then $\boldsymbol{X}^\top \boldsymbol{X}$ is singular, thus $\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$ cannot be directly calculated.

- Solution: regularization

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2$$

Matrix Form with Regularization

- Objective

$$J(\boldsymbol{\theta}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X\theta})^\top(\boldsymbol{y} - \boldsymbol{X\theta}) + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 \qquad \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

- Gradient

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X\theta}) + \lambda\boldsymbol{\theta}$$

- Solution

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbf{0} \;\rightarrow\; -\boldsymbol{X}^\top(\boldsymbol{y} - \boldsymbol{X\theta}) + \lambda\boldsymbol{\theta} = \mathbf{0}$$

$$\rightarrow\; \boldsymbol{X}^\top\boldsymbol{y} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{\theta}$$

$$\rightarrow\; \hat{\boldsymbol{\theta}} = (\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Linear Discriminative Models

- Discriminative model
  - modeling the dependence of unobserved variables on observed ones
  - also called conditional models.
  - Deterministic: $y = f_\theta(x)$
  - **Probabilistic**: $p_\theta(y|x)$

- Linear regression with Gaussian noise model

$$y = f_\theta(x) + \epsilon = \theta_0 + \sum_{j=1}^{d} \theta_j x_j + \epsilon = \theta^\top x + \epsilon$$

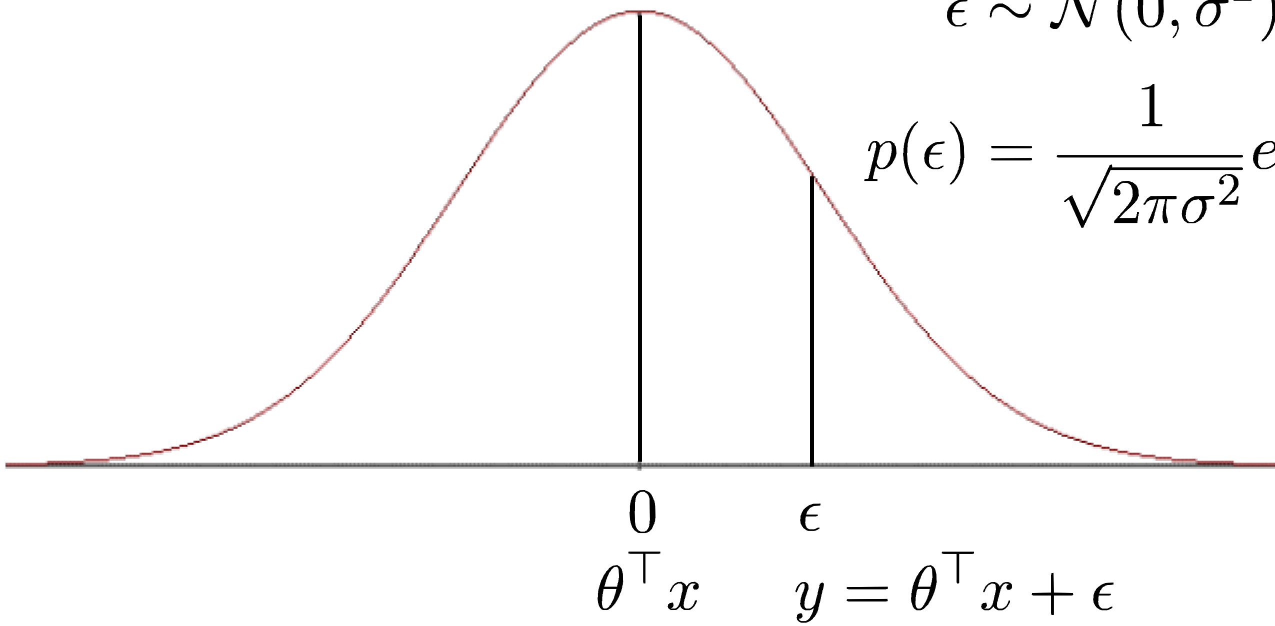
$$\epsilon \sim \mathcal{N}(0, \sigma^2)$$

$$x = (1, x_1, x_2, \ldots, x_d)$$

Objective: Likelihood

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

$$p(\epsilon) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\epsilon^2}{2\sigma}}$$



- Data likelihood

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\theta^\top x)^2}{2\sigma}}$$

Learning

- Maximize the data likelihood

$$\max_{\theta} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \theta^\top x_i)^2}{2\sigma}}$$

- Maximize the data log-likelihood

$$\log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \theta^\top x_i)^2}{2\sigma}} = \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - \theta^\top x_i)^2}{2\sigma}}$$

$$= -\sum_{i=1}^{N} \frac{(y_i - \theta^\top x_i)^2}{2\sigma} + \text{const}$$

$$\min_{\theta} \sum_{i=1}^{N} (y_i - \theta^\top x_i)^2$$

Equivalent to least square error learning

Linear Classification

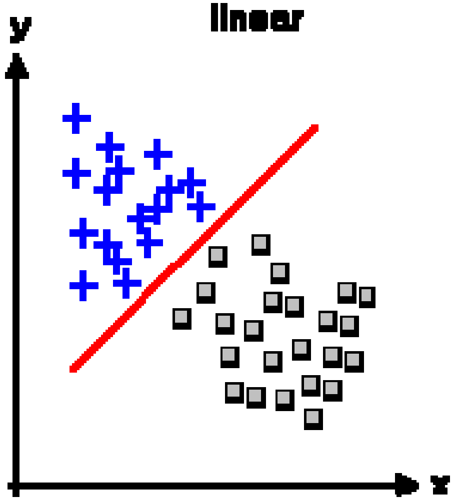
Linear Models for Supervised Learning

Classification Problem

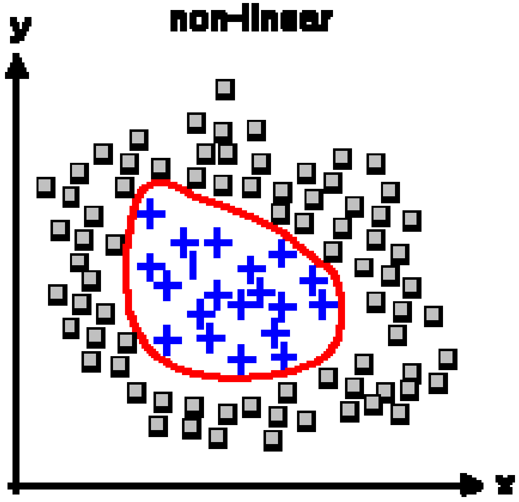
- Given:
  - A description of an instance, $x \in \mathbb{X}$, where $\mathbb{X}$ is the instance space.
  - A fixed set of categories: $C = \{c_1, c_2, \ldots, c_m\}$

- Determine:
  - The category of x : $f(x) \in C$, where $f(x)$ is a categorization function whose domain is $\mathbb{X}$ and whose range is C
  - If the category set binary, i.e. $C = \{0, 1\}$ ({false, true}, {negative, positive}) then it is called binary classification.

Binary Classification



Linearly inseparable



Non-linearly inseparable

Linear Discriminative Models

- Discriminative model
  - modeling the dependence of unobserved variables on observed ones
  - also called conditional models.
  - Deterministic: $y = f_\theta(x)$
    - Non-differentiable
  - **Probabilistic**: $p_\theta(y|x)$
    - Differentiable
- For binary classification

$$p_\theta(y = 1|x)$$

$$p_\theta(y = 0|x) = 1 - p_\theta(y = 1|x)$$

Loss Function

- Cross entropy loss

$$H(p,q) = -\sum_x p(x) \log q(x)$$

$$H(p,q) = -\int_x p(x) \log q(x) dx$$

- For classification problem

Ground Truth	0	1	0	0	0
Prediction	0.1	0.6	0.05	0.05	0.2

$$\mathcal{L}(y, x, p_\theta) = \sum_k \delta(y = c_k) \log p_\theta(y = c_k | x)$$

$$\delta(z) = \begin{cases} 1, & z \text{ is true} \\ 0, & \text{otherwise} \end{cases}$$

Cross Entropy for Binary Classification

	Class 1	Class 2
Ground Truth	0	1
Prediction	0.3	0.7

- Loss function

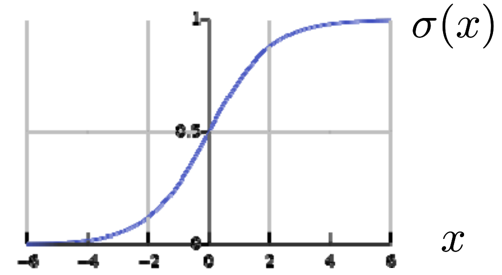
$$
\begin{aligned}
\mathcal{L}(y, x, p_\theta) &= -\delta(y=1)\log p_\theta(y=1|x) - \delta(y=0)\log p_\theta(y=0|x) \\
&= -y\log p_\theta(y=1|x) - (1-y)\log(1-p_\theta(y=1|x))
\end{aligned}
$$

Logistic Regression

- Logistic regression is a binary classification model

$$p_\theta(y=1|x) = \sigma(\theta^\top x) = \frac{1}{1+e^{-\theta^\top x}}$$

$$p_\theta(y=0|x) = \frac{e^{-\theta^\top x}}{1+e^{-\theta^\top x}}$$



- Cross entropy loss function

$$\mathcal{L}(y, x, p_\theta) = -y \log \sigma(\theta^\top x) - (1-y)\log(1-\sigma(\theta^\top x))$$

- Gradient

$$\frac{\partial \mathcal{L}(y, x, p_\theta)}{\partial \theta} = -y \frac{1}{\sigma(\theta^\top x)} \sigma(z)(1-\sigma(z))x - (1-y)\frac{-1}{1-\sigma(\theta^\top x)}\sigma(z)(1-\sigma(z))x$$

$$= (\sigma(\theta^\top x) - y)x$$

$$\theta \leftarrow \theta + \eta(y - \sigma(\theta^\top x))x$$

$$\boxed{\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))}$$

Label Decision

- Logistic regression provides the probability

$$p_\theta(y = 1|x) = \sigma(\theta^\top x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$p_\theta(y = 0|x) = \frac{e^{-\theta^\top x}}{1 + e^{-\theta^\top x}}$$

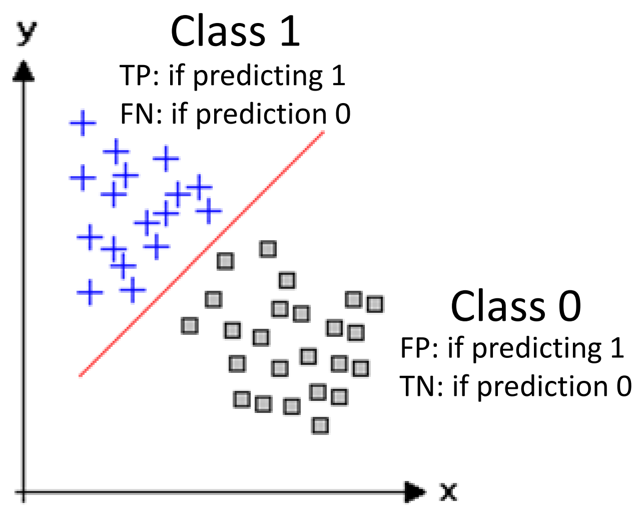
- The final label of an instance is decided by setting a threshold h

$$\hat{y} = \begin{cases} 1, & p_\theta(y = 1|x) > h \\ 0, & \text{otherwise} \end{cases}$$

Evaluation Measures

		Prediction	
		1	0
Label	1	True Positive	False Negative
	0	False Positive	True Negative

- True / False
  - True: prediction = label
  - False: prediction ≠ label
- Positive / Negative
  - Positive: predict y = 1
  - Negative: predict y = 0



Evaluation Measures

		Prediction	
		1	0
Label	1	True Positive	False Negative
	0	False Positive	True Negative

- Accuracy: the ratio of cases when prediction = label

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Evaluation Measures

Prediction

Label

	1	0
1	True Positive	False Negative
0	False Positive	True Negative

- **Precision**: the ratio of true class 1 cases in those with prediction 1

$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Prediction

Label

	1	0
1	True Positive	False Negative
0	False Positive	True Negative

- **Recall**: the ratio of cases with prediction 1 in all true class 1 cases

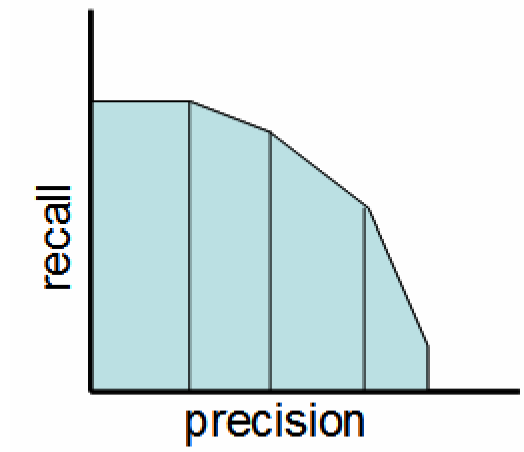
$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Evaluation Measures

- Precision-recall tradeoff

$$\hat{y} = \begin{cases} 1, & p_\theta(y = 1|x) > h \\ 0, & \text{otherwise} \end{cases}$$

  - Higher threshold, higher precision, lower recall
    - Extreme case: threshold = 0
  - Lower threshold, lower precision, higher recall
    - Extreme case: threshold = 0

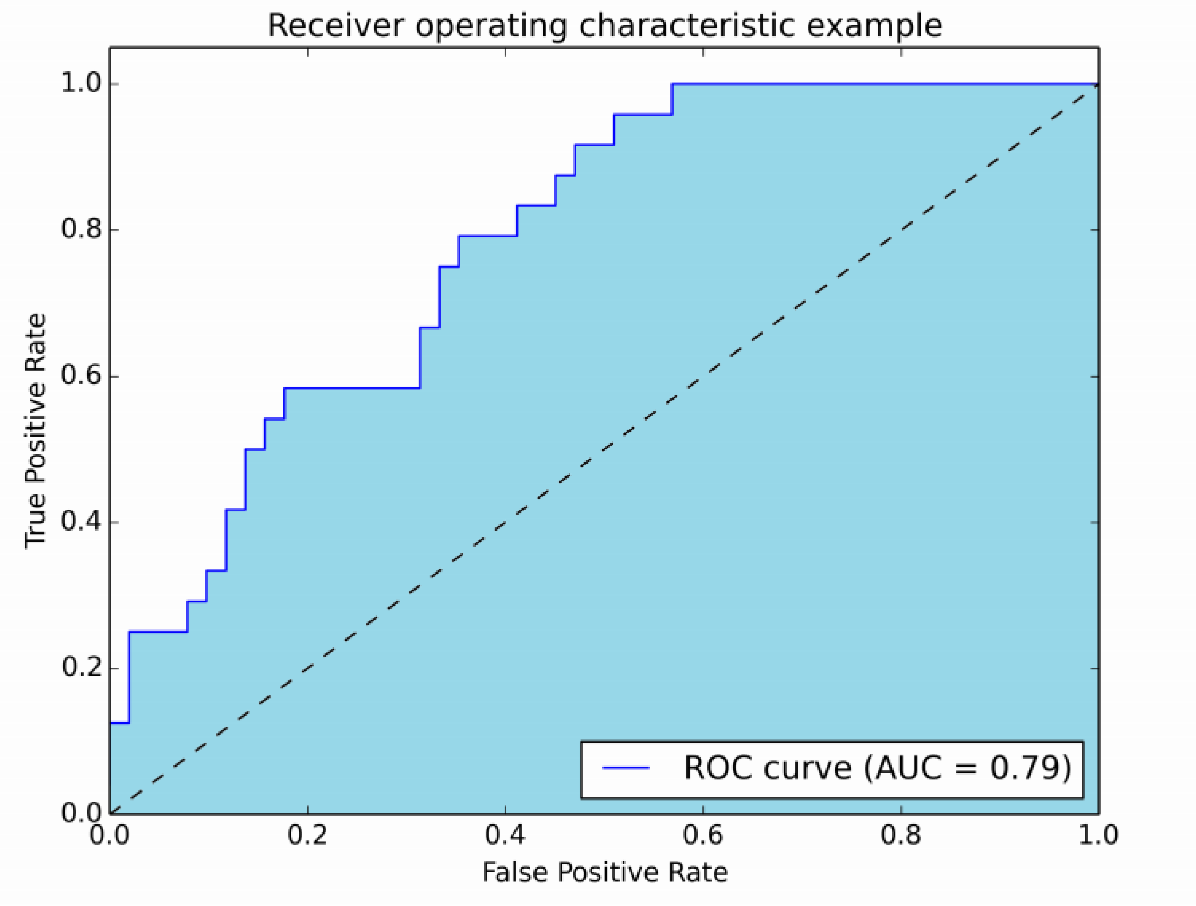


- F1 Measure

$$\text{F1} = \frac{2 \times \text{Prec} \times \text{Recall}}{\text{Prec} + \text{Rec}}$$

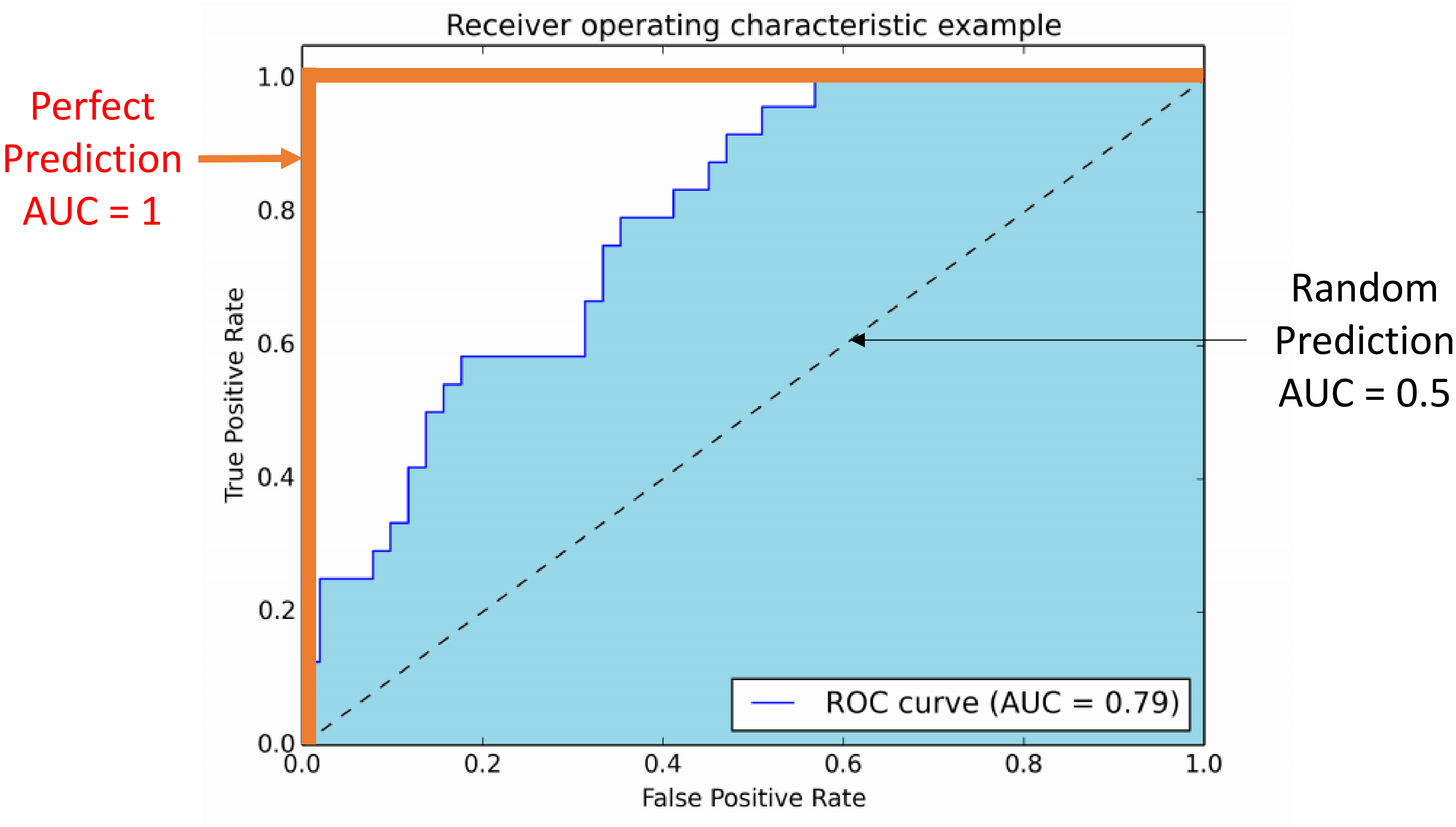
Evaluation Measures

- Ranking-based measure: Area Under ROC Curve (AUC)



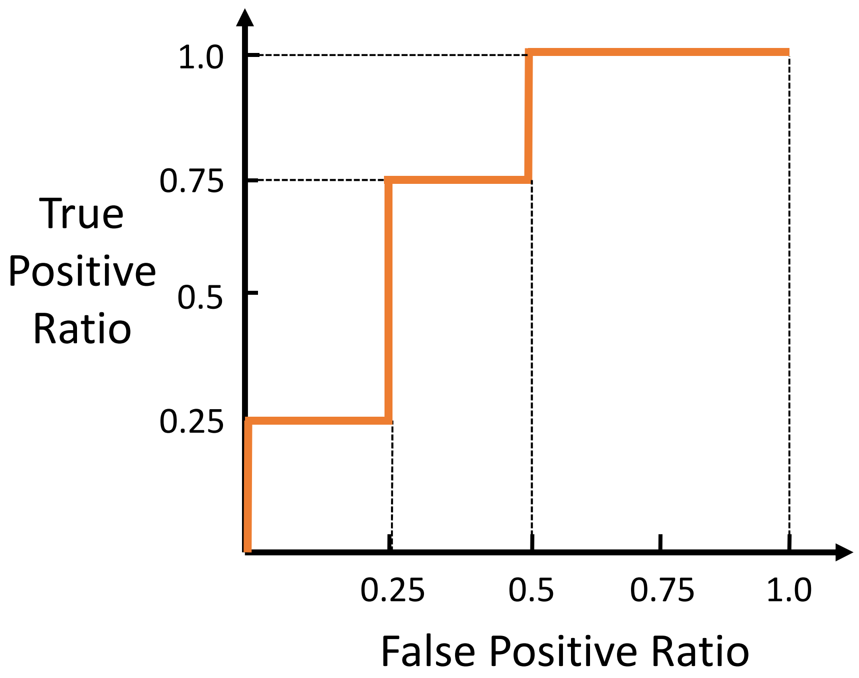
Evaluation Measures

- Ranking-based measure: Area Under ROC Curve (AUC)



Evaluation Measures

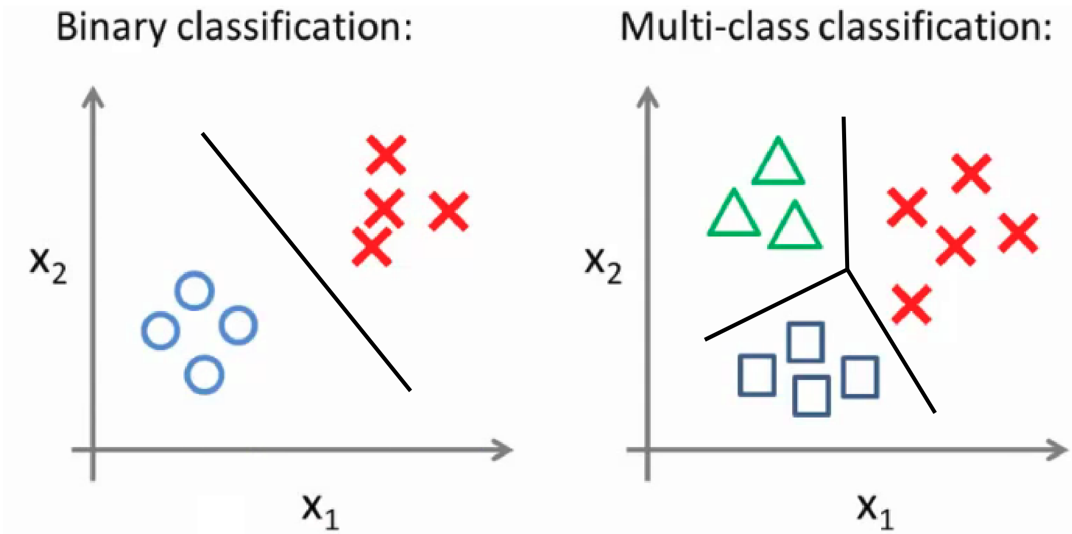
- A simple example of Area Under ROC Curve (AUC)



AUC = 0.75

Prediction	Label
0.91	1
0.85	0
0.77	1
0.72	1
0.61	0
0.48	1
0.42	0
0.33	0

Multi-Class Classification



- Still cross entropy loss

Ground Truth	0	1	0
Prediction	0.1	0.7	0.2

$$\mathcal{L}(y, x, p_\theta) = \sum_k \delta(y = c_k) \log p_\theta(y = c_k | x) \qquad \delta(z) = \begin{cases} 1, & z \text{ is true} \\ 0, & \text{otherwise} \end{cases}$$

Multi-Class Logistic Regression

- Class set $C = \{c_1, c_2, \dots, c_m\}$

- Predicting the probability of $p_\theta(y = c_j|x)$

$$p_\theta(y = c_j|x) = \frac{e^{\theta_j^\top x}}{\sum_{k=1}^m e^{\theta_j^\top x}} \quad \text{for } j = 1, \dots, m$$

- Softmax
  - Parameters $\theta = \{\theta_1, \theta_2, \dots, \theta_m\}$
  - Can be normalized with m-1 groups of parameters

Multi-Class Logistic Regression

- Learning on one instance $(x, y = c_j)$
  - Maximize log-likelihood

$$\max_{\theta} \log p_{\theta}(y = c_j | x)$$

  - Gradient

$$\begin{aligned}
\frac{\partial p_{\theta}(y = c_j | x)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} \log \frac{e^{\theta_j^{\top} x}}{\sum_{k=1}^{m} e^{\theta_k^{\top} x}} \\
&= x - \frac{\partial}{\partial \theta_j} \log \sum_{k=1}^{m} e^{\theta_k^{\top} x} \\
&= x - \frac{e^{\theta_j^{\top} x} x}{\sum_{k=1}^{m} e^{\theta_k^{\top} x}}
\end{aligned}$$

Application Case Study: Click-Through Rate (CTR) Estimation in Online Advertising

Linear Models for Supervised Learning

Ad Click-Through Rate Estimation

[http://news.ifeng.com]

User response estimation problem

- Problem definition

One instance data

- Date: 20160320
- Hour: 14
- Weekday: 7
- IP: 119.163.222.*
- Region: England
- City: London
- Country: UK
- Ad Exchange: Google
- Domain: yahoo.co.uk
- URL: http://www.yahoo.co.uk/abc/xyz.html
- OS: Windows
- Browser: Chrome
- Ad size: 300*250
- Ad ID: a1890
- User occupation: Student
- User tags: Sports, Electronics

Corresponding label

Click (1) or not (0)?

Predicted CTR (0.15)

One-Hot Binary Encoding

- A standard feature engineering paradigm

x=[Weekday=Friday, Gender=Male, City=Shanghai]

x=[0,0,0,0,1,0,0 0,1 0,0,1,0…0]

Sparse representation: x=[5:1 9:1 12:1]

- High dimensional sparse binary feature vector
  - Usually higher than 1M dimensions, even 1B dimensions
  - Extremely sparse

Training/Validation/Test Data

- Examples (in LibSVM format)

  1 5:1 9:1 12:1 45:1 154:1 509:1 4089:1 45314:1 988576:1
  0 2:1 7:1 18:1 34:1 176:1 510:1 3879:1 71310:1 818034:1
  …

- Training/Validation/Test data split
  - Sort data by time
  - Train:validation:test = 8:1:1
  - Shuffle training data

Training Logistic Regression

- Logistic regression is a binary classification model

$$p_\theta(y = 1|x) = \sigma(\theta^\top x) = \frac{1}{1 + e^{-\theta^\top x}}$$

- Cross entropy loss function with L2 regularization

$$\mathcal{L}(y, x, p_\theta) = -y \log \sigma(\theta^\top x) - (1 - y) \log(1 - \sigma(\theta^\top x)) + \frac{\lambda}{2}||\theta||_2^2$$

- Parameter learning

$$\theta \leftarrow (1 - \lambda\eta)\theta + \eta(y - \sigma(\theta^\top x))x$$

  - Only update non-zero entries

Experimental Results

- Datasets
  - Criteo Terabyte Dataset
    - 13 numerical fields, 26 categorical fields
    - 7 consecutive days out of 24 days in total (about 300 GB) during 2014
    - 79.4M impressions, 1.6M clicks after negative down sampling
  - iPinYou Dataset
    - 65 categorical fields
    - 10 consecutive days during 2013
    - 19.5M impressions, 937.7K clicks without negative down sampling

Performance

Model	Linearity	AUC		Log Loss	
		Criteo	iPinYou	Criteo	iPinYou
Logistic Regression	Linear	71.48%	73.43%	0.1334	5.581e-3
Factorization Machine	Bi-linear	72.20%	75.52%	0.1324	5.504e-3
Deep Neural Networks	Non-linear	75.66%	76.19%	0.1283	5.443e-3

- Compared with non-linear models, linear models
  - Pros: standardized, easily understood and implemented, efficient and scalable
  - Cons: modeling limit (feature independent assumption), cannot explore feature interactions

[Yanru Qu et al. Product-based Neural Networks for User Response Prediction. ICDM 2016.]

For more machine learning materials, you can check out my machine learning course at Zhiyuan College

CS420 Machine Learning

Weinan Zhang

Course webpage:

http://wnzhang.net/teaching/cs420/index.html