Deep Learning

Machine Learning Basics

Gunhee Kim

Computer Science and Engineering

서울대학교

SEOUL NATIONAL UNIVERSITY

Outline

- A (Very) Short Intro to ML
- Capacity, Overfitting and Underfitting

Machine Learning (ML)

A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data

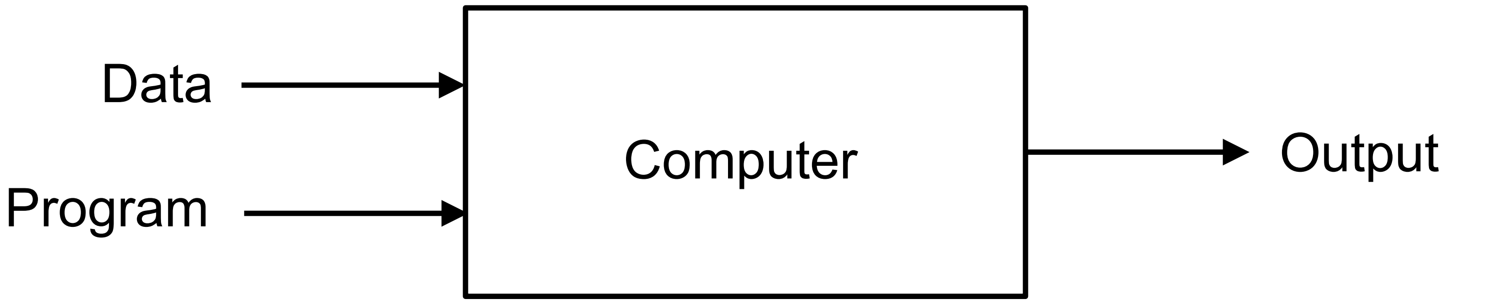
A ML algorithm is an algorithm that can ***learn*** from data

Tom Mitchell's definition

- *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*
- *T* : classification, regression, detection, …
- *P* : error rate, accuracy, likelihood, margin …
- *E*: data

Machine Learning (ML)

Traditional programming



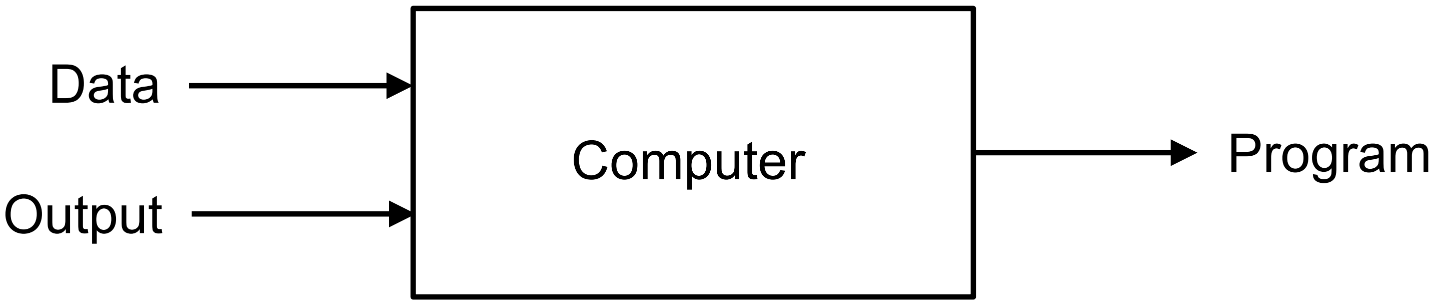
Data

Program

Computer

Output

Machine learning



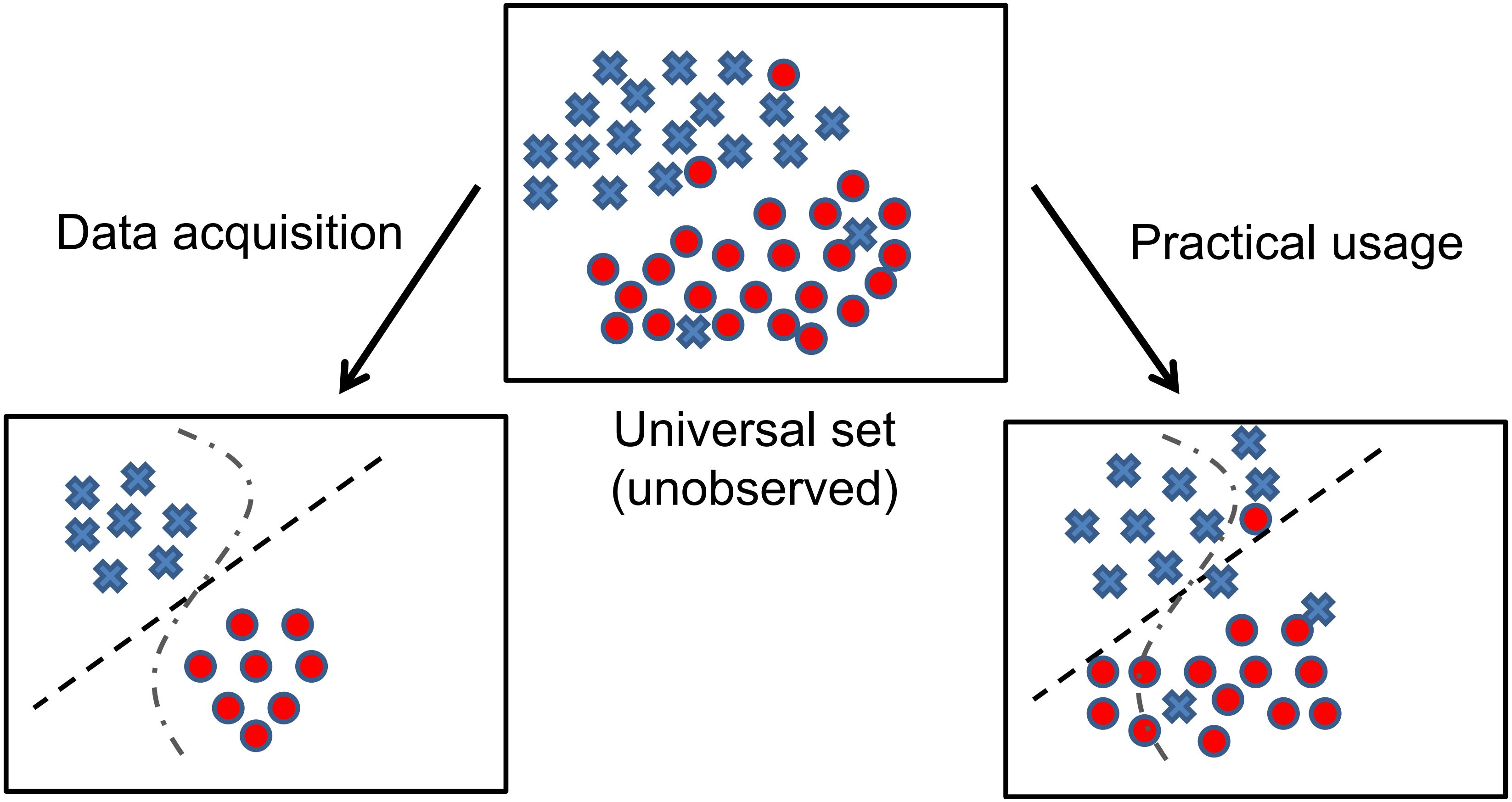
Data

Output

Computer

Program

Training Data Versus Test



Data acquisition

Practical usage

Universal set
(unobserved)

Training set
(observed)

Testing set
(unobserved)

Types of Learning

Supervised learning

- Training data includes desired outputs

Unsupervised learning

- Training data does not include desired outputs

Semi-supervised learning

- Some of training data includes desired outputs

Reinforcement learning

- Does not experience a fixed dataset
- A feedback loop between the learning system and its environment

Types of Learning

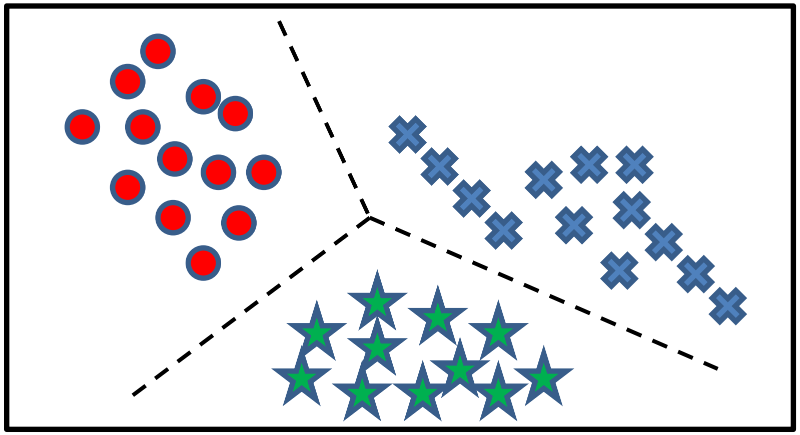
Supervised learning ($\{(\boldsymbol{x}_n \in \mathbb{R}^d, y_n \in \mathbb{R})\}_{n=1}^{N}$)

- Prediction
- Classification (discrete labels), Regression (real values)

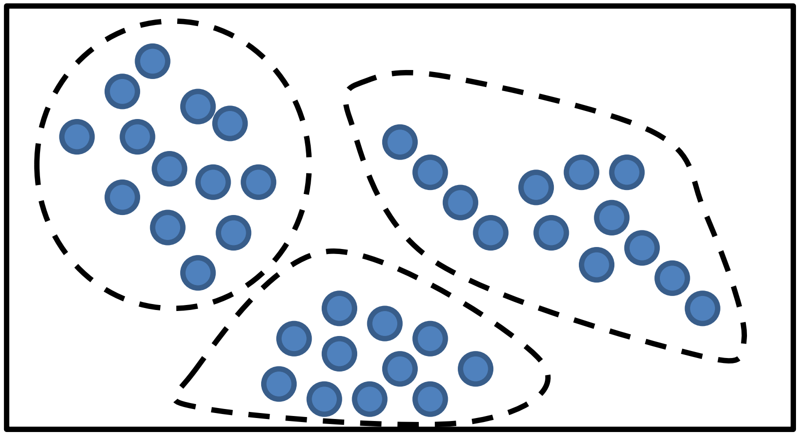
Unsupervised learning ($\{(\boldsymbol{x}_n \in \mathbb{R}^d)\}_{n=1}^{N}$)

- Clustering
- Probability distribution estimation
- Dimension reduction

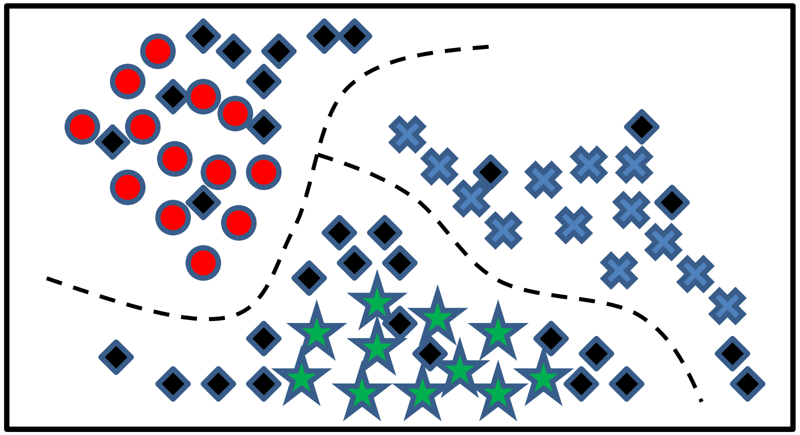
Types of Learning



Supervised learning



Unsupervised learning



Semi-supervised learning

Outline

- A (Very) Short Intro to ML
- Capacity, Overfitting and Underfitting

(Credits: Goodfellow et al’s Deep Learning)

Two Basic (Supervised) ML Problems

Classification $f(\boldsymbol{x}|\boldsymbol{w},\boldsymbol{b}) = sign(\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b})$

- Predict which class an example belongs to
- e.g., spam filtering example

Regression $f(\boldsymbol{x}|\boldsymbol{w},\boldsymbol{b}) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$

- Predict a real value or a probability
- e.g., probability of being spam

Both problems are highly inter-related

- Train on regression ➔ Use for classification

Formal Definitions

Training data $S = \{(\boldsymbol{x}_i, y_i)\}_{n=1}^{N}$ $\boldsymbol{x} \in \mathbb{R}^d, y \in \{-1,1\}$

Model class (a.k.a. hypothesis class)

$$h(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{b}) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$$ Linear Models

Goal: find $(\boldsymbol{w}, \boldsymbol{b})$ that predicts **well** on S

Loss Function

- Regression $L(a, b) = (a - b)^2$ Squared Loss
- Classification $L(a, b) = 1_{[a \neq b]}$ or $1_{[sign(a) \neq sign(b)]}$ 0/1 Loss

Learning objective $\text{argmin}_{\boldsymbol{w},\boldsymbol{b}} \sum_{i=1}^{N} L(y_i, h(\boldsymbol{x}_i|\boldsymbol{w}, \boldsymbol{b}))$

Optimization

Generalization Error

Objective of learning

- Not to learn an exact representation of the training data itself
- To build a statistical model that generates the data

True distribution: $P(\boldsymbol{x}, \boldsymbol{y})$

- All possible cases – unknown to us
- Train and test data are generated by $P(\boldsymbol{x}, \boldsymbol{y})$
- Assumption: iid (independent and identically distributed)

Train: Fit an hypothesis $h(\boldsymbol{x})$

- Using training data $S = \{(\boldsymbol{x}_i, y_i)\}_{n=1}^{N}$, sampled from $P(\boldsymbol{x}, \boldsymbol{y})$

Generalization Error

Generalization Error: $L_p(h) = E_{P(\boldsymbol{x},\boldsymbol{y})}\left[L\left(\boldsymbol{y}, h(\boldsymbol{x})\right)\right]$

- Prediction loss on all possible cases
- *Generalization*: ability to perform well on previously unseen input

Underfitting**:** Generalization Error < Training Error

- The training error is not sufficiently low

Overfitting**:** Generalization Error > Training Error

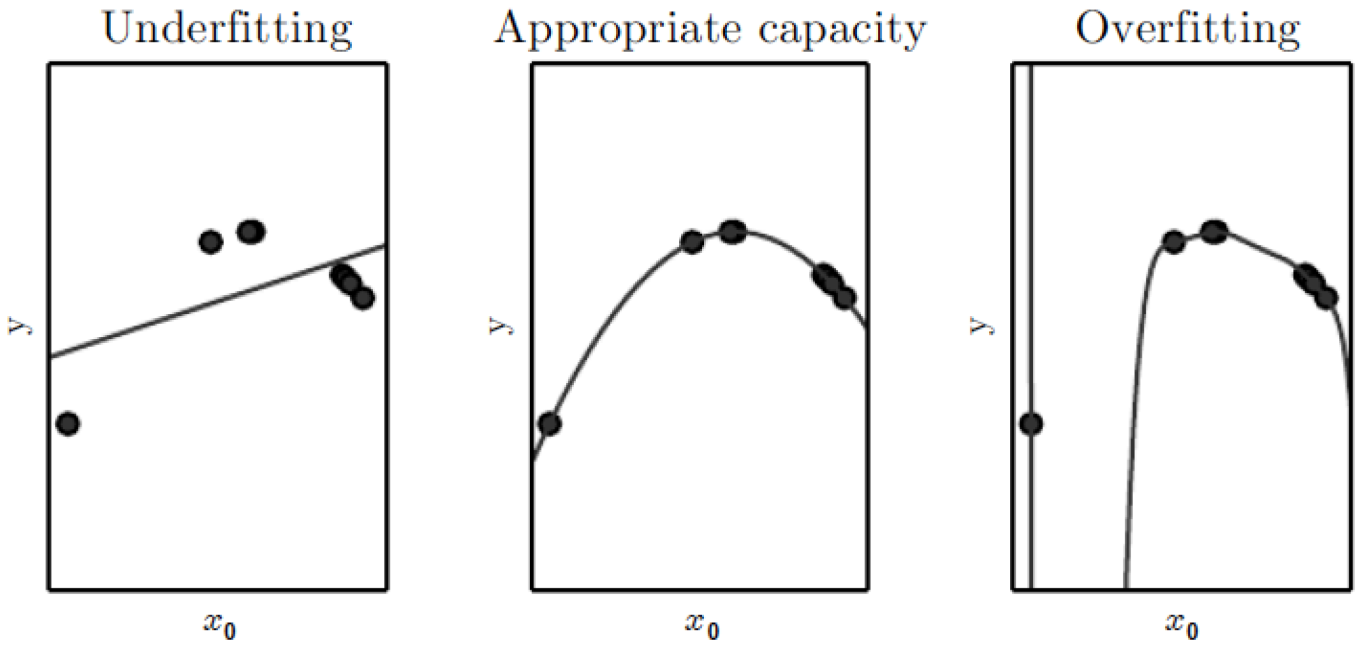
- The gap between the training and test error is too large

Training an ML algorithm well

- 1. Make the training error small
- 2. Make the gap between the training and test error small

Model's Capacity

We have 7 data, and fit them with three models



A linear function

- Suffer from underfitting
- Cannot capture the curvature in the data

A quadratic function

- Generalized well to unseen points

A polynomial of degree 9

- The model exactly passes through all training points
- A deep valley in between two datapoints?

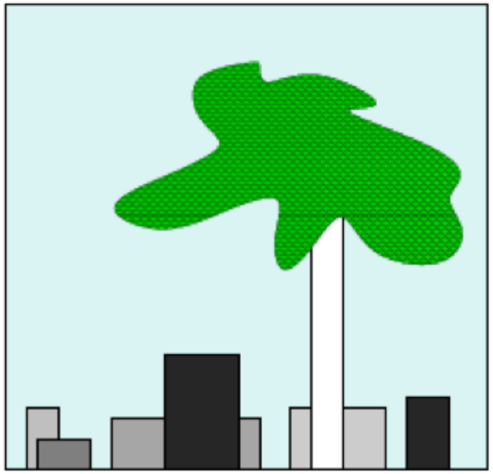
Occam’s Razor (A Principle of Parsimony)

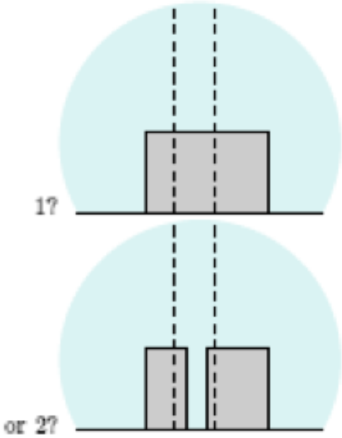
William of Ockham (1285-1349)

Pluralitas non est ponenda sine neccesitate
(entities should not be multiplied unnecessarily)

- All things being equal, the simplest solution tends to be the best one
- The simplest explanation tends to be the right one

How many boxes are behind the tree?

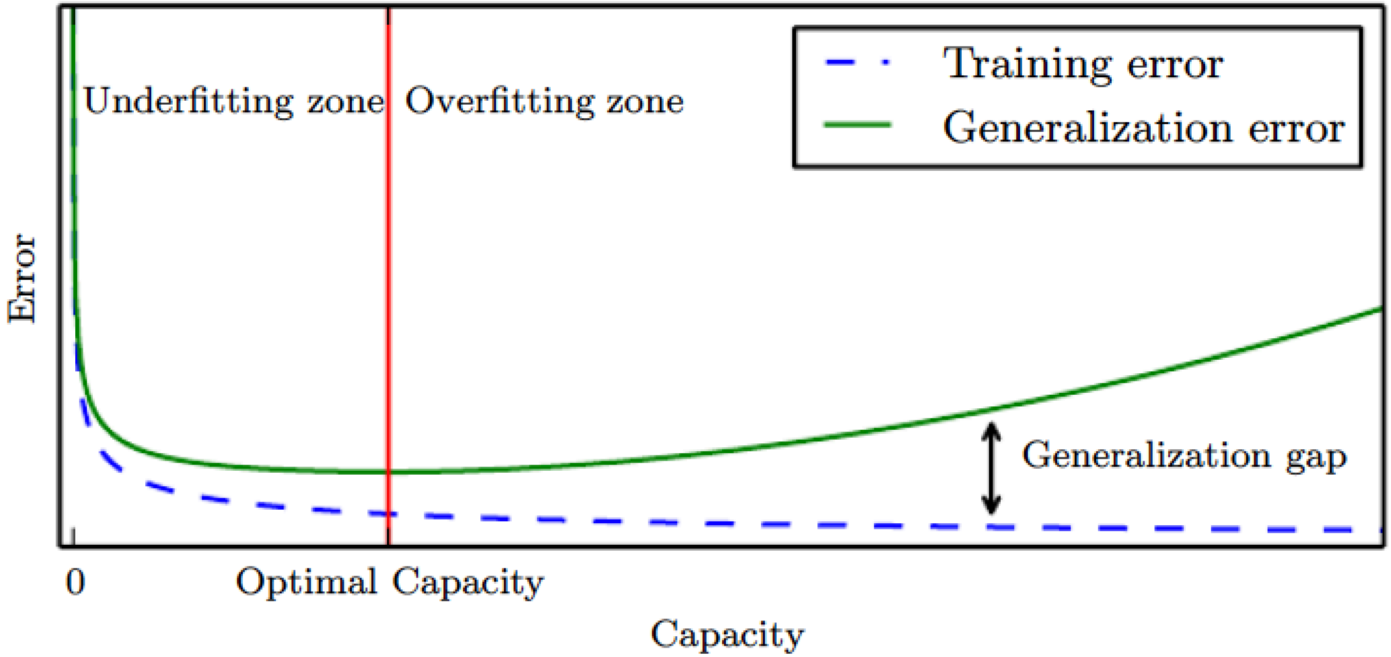




Typical Relation between Capacity and Error

Informally, a capacity is the function’s ability to fit a wide variety of functions

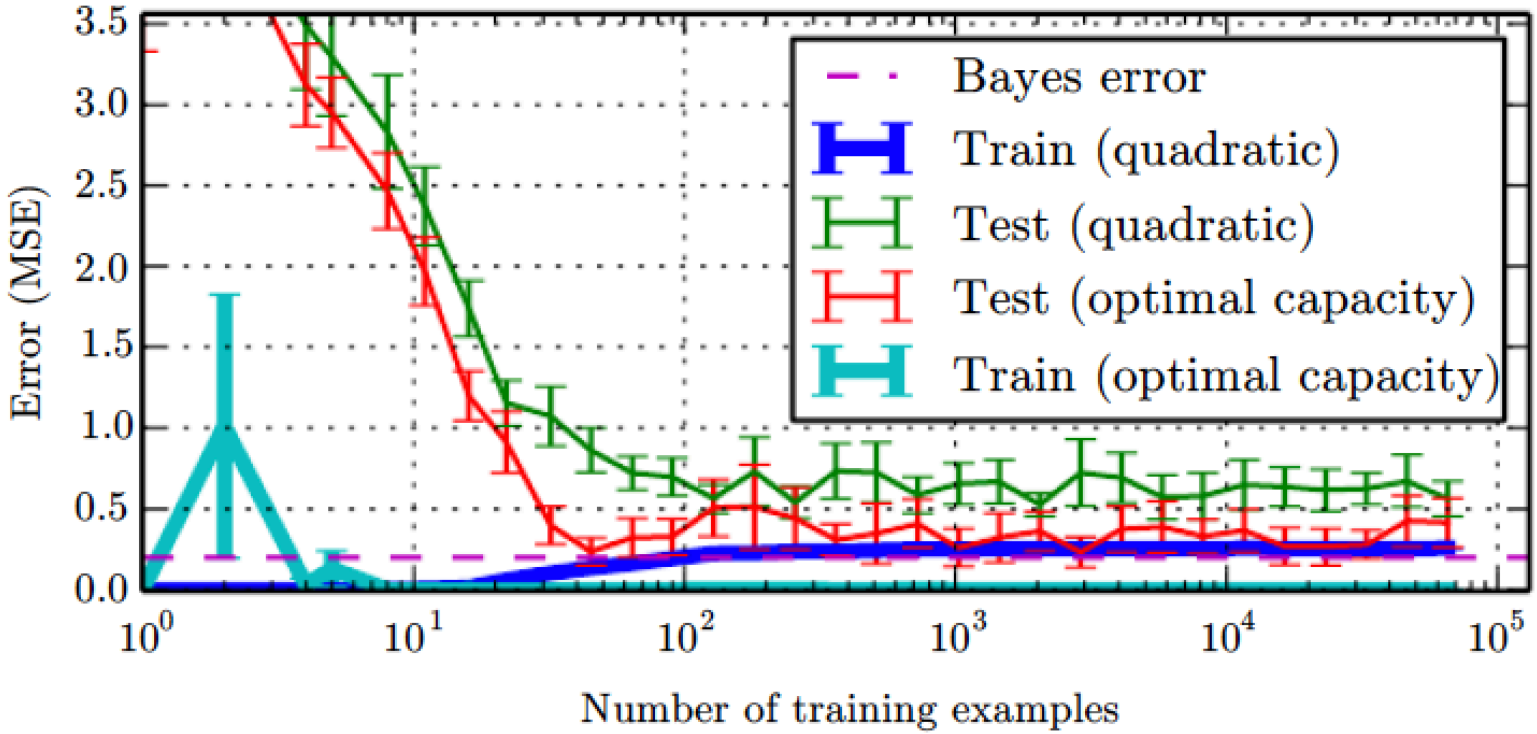
As capacity increases, training errors decreases but the gap increases



Training Data Size vs. Train/Test Errors

A quadratic model vs degree-5 model (optimal capacity)

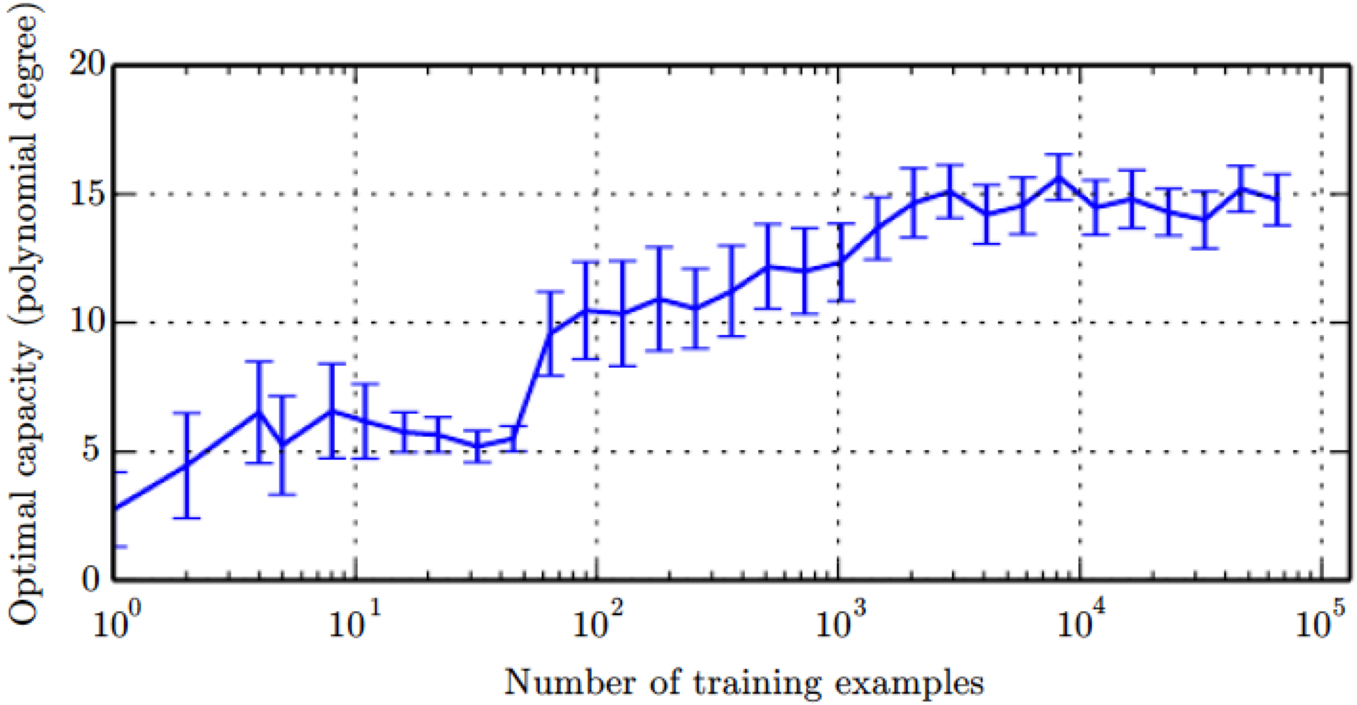
- Bayes error: the lowest possible error
- A high-capacity model is unstable with few training set
- The test error at optimal capacity asymptotes to the Bayes error



Training Data Size vs. Train/Test Errors

A quadratic model vs degree-5 model (optimal capacity)

- As the training set size increases, the optimal capacity increases
- The optimal capacity plateaus after reaching sufficient complexity to solve the task



No Free Lunch Theorem for ML

No Machine learning algorithm is universally any better than any other

Do not try to seek a universal learning algorithm (No absolute best algorithm)

Regularization

Give an ML algorithm, a preference for one solution in its hypothesis space to another

- e.g. weight decay in a linear regression

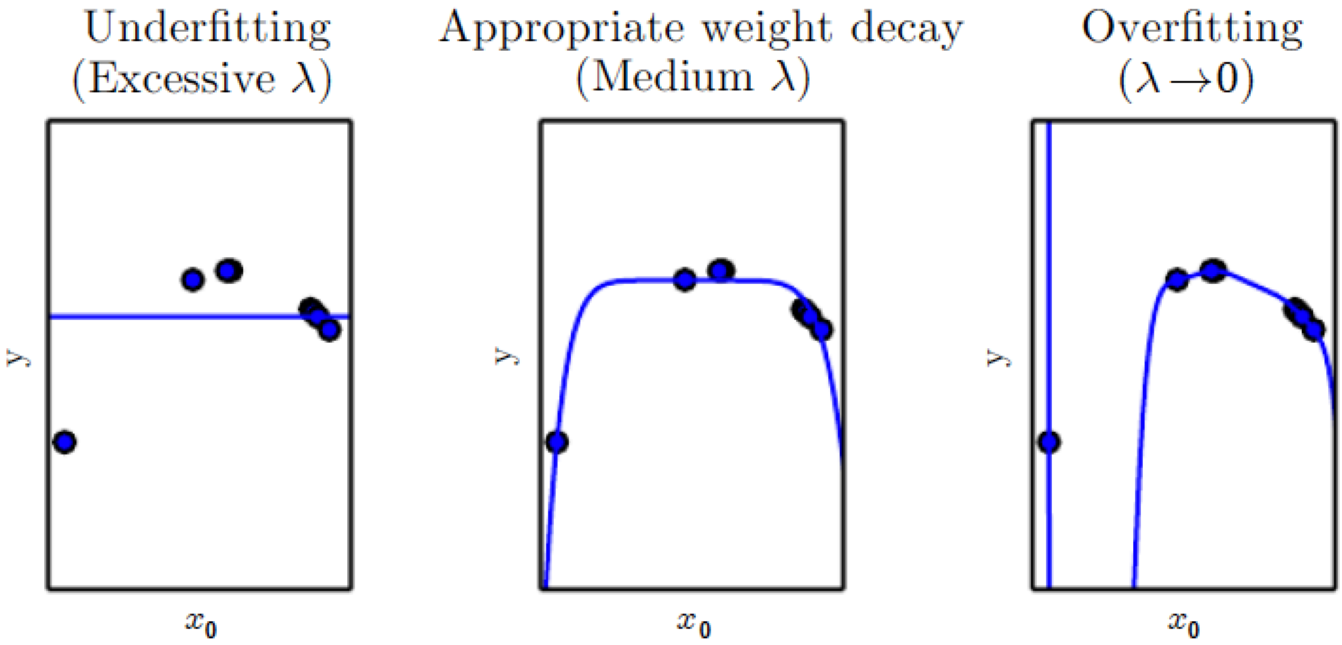
$$J(w) = (error) + \lambda w^T w$$

- λ controls the strength of a preference for smaller weight
- $\lambda = 0$: no preference, a large λ: a smaller weight
- The penalty term is called as a *regularizer*

The main objective of regularization is to reduce its generalization error but not its training error

Regularization

The same problem… but use only model with degree 9



Underfitting (Excessive λ)

Appropriate weight decay (Medium λ)

Overfitting ($\lambda \to 0$)

- No slope at all (i.e. constant function)
- Recover a cover in spite of using a model with degree 9
- Weight decay approaches zero (overfitting)

Hyperparameters

Parameters to control the behavior of the ML algorithm

- e.g. λ as a regularizer constant

How to choose hyperparameters: validation set

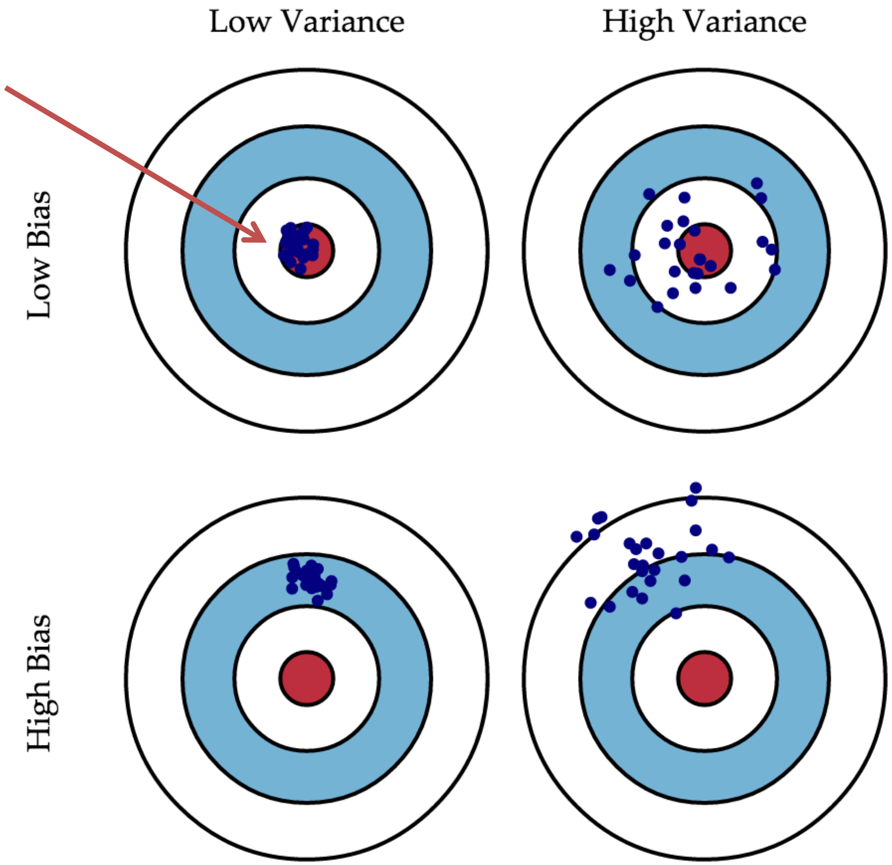
- It comes from the training data, but is not used for training
- However, it can be dangerous to divide the dataset into a fixed training/validation set

K-fold cross-validation

- Split the dataset into K disjoint subsets
- On i-th trial, the i-th subset is used for validation; the other for training
- Take the average test across K trials

Trade-off between Bias and Variance

Two sources of error in an estimator: bias and variance



Correct prediction

Low bias: predicted well

Low variance: Stable

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

Trade-off between Bias and Variance

Two sources of error in an estimator: bias and variance

(Test Error) = (Bias) + (Variance)

- Bias: Expected deviation from the true value of the function
- Variance: Deviation from the expected estimator values obtained from the different sampling of the data

Increasing capacity tends to increase variance and decrease bias

Deep Learning

Numerical Optimization

Gunhee Kim

Computer Science and Engineering

서울대학교
SEOUL NATIONAL UNIVERSITY

Overflow and Underflow

Fundamental issue: Representing real numbers with a finite bit patterns

- Rounding error is problematic, especially when it compounds across many operations

Underflow: numbers near zero are rounded to zero

- Division-by-zero (NaN value) or Logarithm of zero ($-\infty$)

Overflow: numbers with large magnitude are approximated as ∞ or $-\infty$

Overflow and Underflow

Example: softmax function

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

- Suppose that all of the $x_i = c$. → What we expect is all of the output should be equal to $1/n$
- If c is very negative, then $\exp(c)$ will underflow
- if c is very positive, then $\exp(c)$ will overflow

One simple remedy: use $\mathrm{softmax}(\boldsymbol{z})$, $\boldsymbol{z} = \boldsymbol{x} - \mathrm{max}_i x_i$

In practice, many low-level libraries implement stabilization functions

Poor Conditioning

Conditioning: how rapidly a function changes w.r.t. small changes in its input

- Functions that change rapidly when their inputs are perturbed slightly can be problematic for scientific computation

Eigenvalue decomposition

- Given a function $f(\boldsymbol{x}) = \boldsymbol{A}^{-1}\boldsymbol{x}$ where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ has an eigenvalue decomposition, and its ratio of the magnitude of the largest and the smallest eigenvalue

$$max_{i,j}|\lambda_i/\lambda_j|$$

- When it is large, matrix inversion is sensitive to error in the input

It is an intrinsic property of the matrix itself

- Poorly conditioned matrices amplify pre-existing errors

Gradient-based Optimization

Optimization: find $\boldsymbol{x}^*$ that minimizes or maximizes a function $f(\boldsymbol{x})$

- $f(\boldsymbol{x})$: *objective* (a.k.a *cost/loss/error function* for minimization)

Consider univariate case $y = f(x)$

- Derivative of the function $f'(x)$ or dy/dx: the slope of $f(x)$ at x
- Specify how to scale the output changes w.r.t. a small change in an input

$$f(x + \epsilon) \approx f(x) + \epsilon f'(x)$$

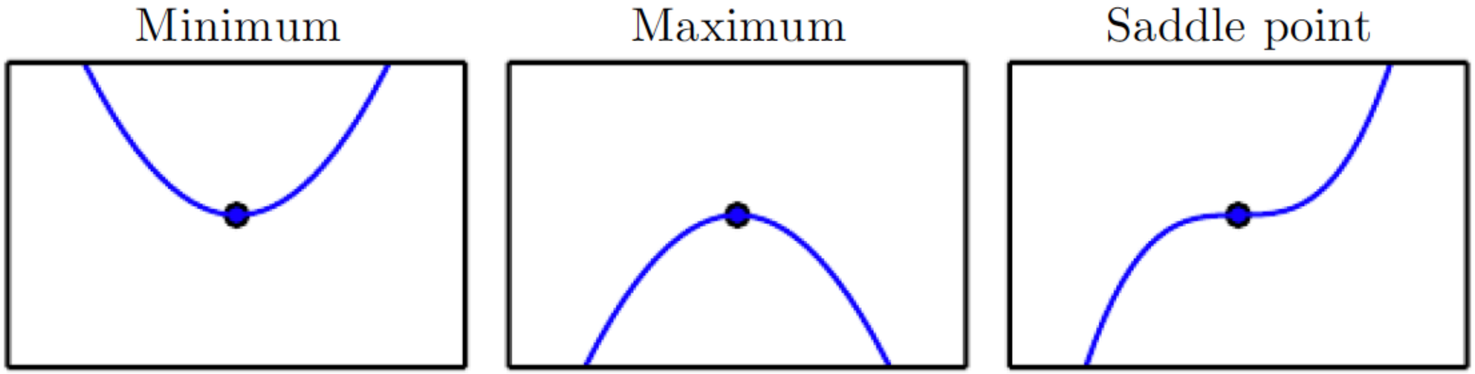
Derivative can tell how to change x to make a small improvement in y (*Gradient descent*)

- e.g. $f(x - \epsilon \mathrm{sign}(f'(x))) \leq f(x)$
- because $f(x - \epsilon \mathrm{sign}(f'(x))) = f(x) - \epsilon^2 \mathrm{sign}(f'(x))\, f'(x)$

Critical Points

The points where $f'(x) = 0$ (i.e. zero slopes)

- Local minimum: a point where $f(\boldsymbol{x})$ is lower than at all neighbors
- Local maximum: a point where $f(\boldsymbol{x})$ is higher than at all neighbors
- Saddle points: neither maxima nor minima



- Global minimum: the absolute lowest value of $f(\boldsymbol{x})$
- In many cases (i.e. non-convex functions), we can only achieves local minimum

Derivative and Gradient

Partial derivative $\partial f(\boldsymbol{x})/\partial x_i$

- The case where input $\boldsymbol{x}$ has multiple dimension
- How f changes as only the variable x_i increases at point $\boldsymbol{x}$

Gradient: generalize the derivative with respect to a vector

- $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = [\partial f(\boldsymbol{x})/\partial x_1, \dots, \partial f(\boldsymbol{x})/\partial x_n]$
- Critical point where $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{0}$ (every element of the gradient = 0)

Directional derivation (in the direction of a unit vector $\boldsymbol{u}$)

- Projection of $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ into $\boldsymbol{u}$: $\boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$

Derivation of Gradient Descent

Goal: find the direction where f decreases the fastest

- We find $\boldsymbol{u}$ such that

$$min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1}\boldsymbol{u}^T\nabla_{\boldsymbol{x}}f(\boldsymbol{x})$$

$$= min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1}||\boldsymbol{u}||_2||\nabla_{\boldsymbol{x}}f(\boldsymbol{x})||_2 \cos\theta$$

$$= min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1} \cos\theta$$

- Since $||\boldsymbol{u}||_2 = 1$ and ignore the terms that do not depend on $\boldsymbol{u}$
- Therefore, $\theta = -\pi$ (i.e. $\boldsymbol{u}$ should be opposite direction to $\nabla_{\boldsymbol{x}}f(\boldsymbol{x})$)

Gradient descent (or *steepest descent*): decrease f by moving in the negative gradient direction

- Iterate $\boldsymbol{x}' = \boldsymbol{x} - \epsilon\nabla_{\boldsymbol{x}}f(\boldsymbol{x})$ until reach a critical point where $\nabla_{\boldsymbol{x}}f(\boldsymbol{x}) = \boldsymbol{0}$ (every element of the gradient = 0)

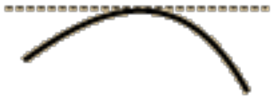
Second Derivative

Second derivative: $\frac{\partial^2 f}{\partial x_i \partial x_j}$

- Derivative w.r.t x_i of the derivative of $\boldsymbol{f}$ w.r.t. x_j
- How the first derivative changes as varying the input

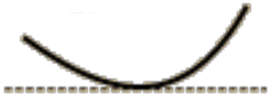
Can determine whether a critical point ($f'(x) = 0$) is local max, min, or saddle point (called *second derivative test*)

Local maxima
if $f''(x) < 0$



$f'(x - \epsilon) > 0$ $f'(x + \epsilon) < 0$

Local minima
if $f''(x) > 0$

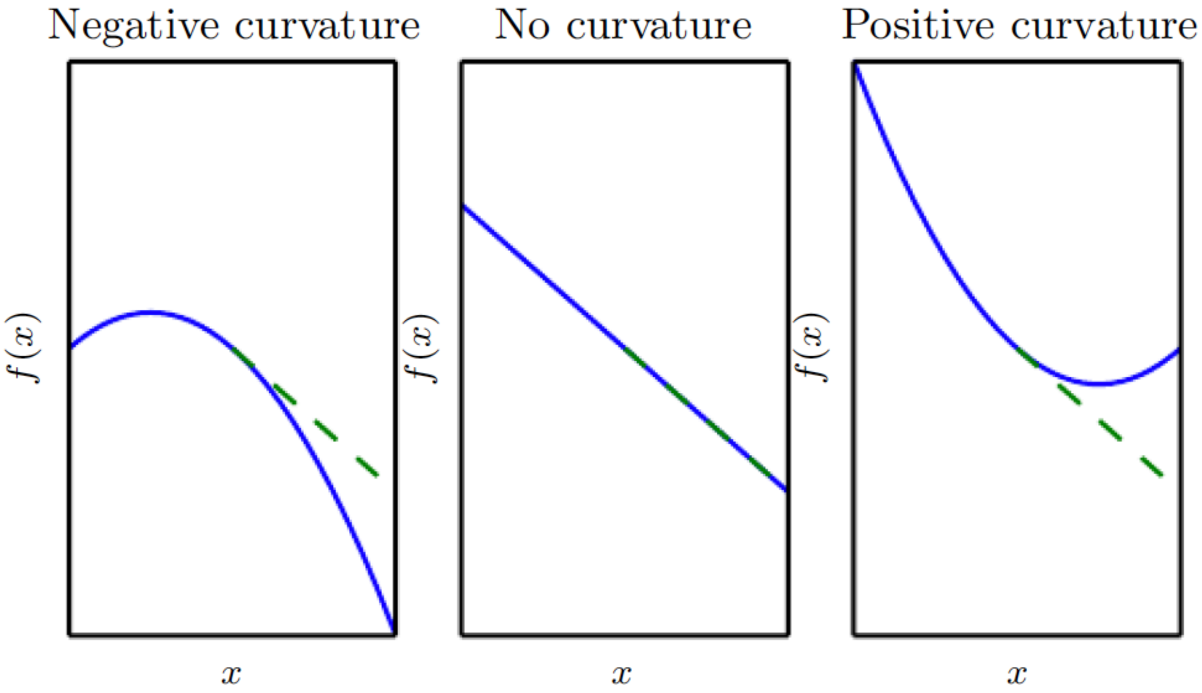


$f'(x - \epsilon) < 0$ $f'(x + \epsilon) > 0$

- If $f''(x) = 0$, the test is inconclusive
  (It can be a saddle point or a part of a flat region)

Second Derivative

Second derivative measures *curvature*



- No curvature: the gradient predicts the cost function correctly
- Negative: the cost decreases faster than the gradient predicts
- Positive: the cost decreases slower (too large step sizes can increase the cost inadvertently)

Beyond the Gradient: Jacobian

Jacobian

- All of the partial derivatives of a function where input and output are both vectors
- For a function $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $\boldsymbol{f}$

$$= \left[\frac{\partial f(\boldsymbol{x})_i}{\partial x_j}\right]_{i=1:m,j=1:n} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})_1}{\partial x_1}, & \dots & , \frac{\partial f(\boldsymbol{x})_1}{\partial x_n} \\ & \dots & \\ \frac{\partial f(\boldsymbol{x})_m}{\partial x_1}, & \dots & , \frac{\partial f(\boldsymbol{x})_m}{\partial x_n} \end{bmatrix}$$

Beyond the Gradient: Hessian

Hessian: Jacobian of the gradient

- A square matrix of second-order partial derivatives of a scalar-valued function
- For a function $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ of $\boldsymbol{f}$

$$= \left[\frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}\right]_{i,j=1:n} = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2}, & \cdots & , \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ & \cdots & \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_i}, & \cdots & , \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n^2} \end{bmatrix}$$

Hessian is symmetric

- The differential operators are commutative ($\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$)

Hessian for Second Derivate Test

Second derivative test in high dimension

- e.g. in one-dimension, at $f'(x) = 0$, if $f''(x) > 0$, it's local minimum
- For a critical point $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{0}$, if Hessian is positive-definite (i.e. all eigenvalues are positive), it's local minimum
- If Hessian is negative-definite, it's local maximum
- If Hessian has differently signed eigenvalues, it's a saddle point
- If at least one of eigenvalues are zeros and all the others have the same sign, it's inconclusive

Directional second-derivative

- For a direction $\boldsymbol{u}$, the projection of second derivative is $\boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u}$
- If $\boldsymbol{u}$ is an eigenvector $\boldsymbol{d}$, then $\boldsymbol{d}^T \boldsymbol{H} \boldsymbol{d}$ is its corresponding eigenvalue
- The max/min eigenvalue determines the max/min second derivative

Hessian and Gradient Descent

A second-order Taylor approximation to a function $f(\boldsymbol{x})$ around the point $\boldsymbol{x}^{(0)}$

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^{(0)}\right) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{g} + \frac{1}{2}\,(\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{H} (\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

- By GD, the new parameter is $\boldsymbol{x} = \boldsymbol{x}_0 - \epsilon \boldsymbol{g}$. Therefore,

$$f\left(\boldsymbol{x}^{(0)} - \epsilon \boldsymbol{g}\right) \approx f\left(\boldsymbol{x}^{(0)}\right) - \epsilon \boldsymbol{g}^T \boldsymbol{g} + \frac{1}{2} \epsilon^2 \boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}$$

- If the last term is too large, the GD step can move uphill
- If it is zero or negative, the GD will decrease the function forever

If $\boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g} > 0$, the optimal step size is $\epsilon^* = \dfrac{\boldsymbol{g}^T \boldsymbol{g}}{\boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g}}$

- The worst case is $\boldsymbol{g}^T \boldsymbol{H} \boldsymbol{g} = c\lambda_{max}$, thus it becomes $1/\lambda_{max}$

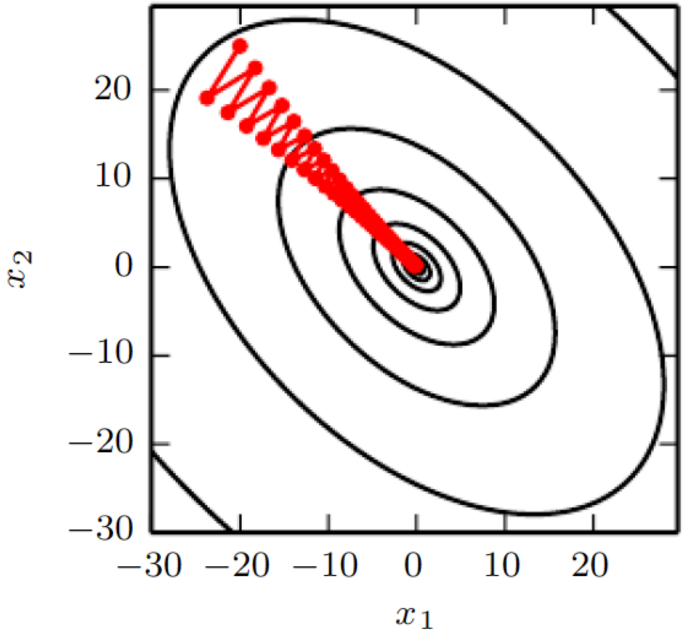
Poorly Conditioned Hessian

If Hessian has a poor condition number ($max_{i,j}|\lambda_i/\lambda_j|$)

- The derivative increases rapidly in one direction, while very slowly in another direction

Difficult to choose a good step size for GD

- Waste too much time
- Overshot for one direction (i.e. eigenvalue with large λ)
- May be too small for another (i.e. eigenvalue with small λ)
- Use GDs with adaptive learning rate (e.g. AdaGrad, Adam)



Second-Order Method

Newton’s method

- Using the second-derivative information for faster convergence
- Idea: (1) Approximate the function using a second-order Taylor approximation, (2) directly jump to the critical point, and (3) Iterate it

Derivation

- A second-order Taylor approximation

$$f(\boldsymbol{x}) \approx f\left(\boldsymbol{x}^{(0)}\right) + (\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{g} + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}^{(0)})^T \boldsymbol{H} (\boldsymbol{x} - \boldsymbol{x}^{(0)})$$

- Find a critical point

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = 0 \quad \Rightarrow \boldsymbol{g} + \boldsymbol{H}\left(\boldsymbol{x} - \boldsymbol{x}^{(0)}\right) = 0 \quad \Rightarrow \boldsymbol{x}^* = \boldsymbol{x}^{(0)} - \boldsymbol{H}^{-1}\boldsymbol{g}$$

- Useful near a local minimum, but harmful near a saddle point

Deep Learning

Linear/Logistic/Softmax Regression

Gunhee Kim

Computer Science and Engineering

서울대학교

SEOUL NATIONAL UNIVERSITY

Outline

- Linear Regression
  - Least Mean Square algorithm
  - Normal equations
  - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

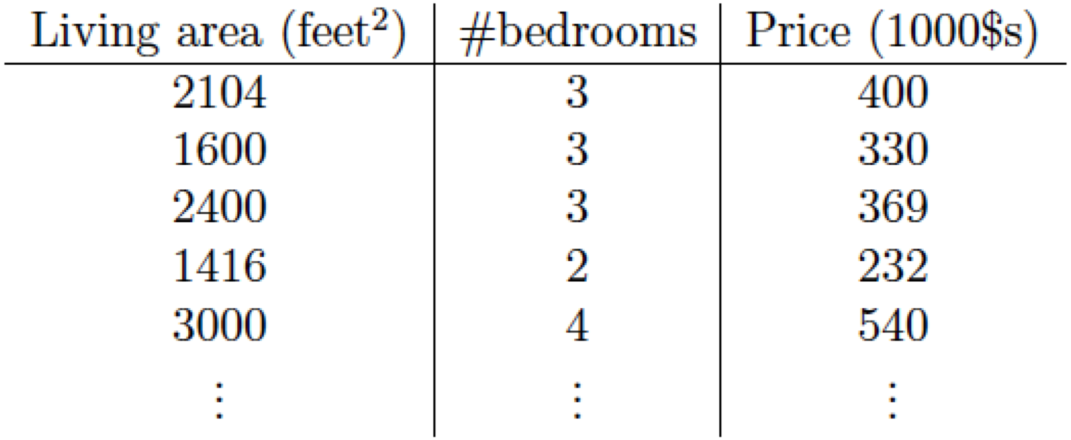
(Reference: Andrew Ng's CS229 Lecture Notes)

Simple Example of Linear Regression

Can we predict a house price from living area and # bedrooms?

Features $\boldsymbol{x}$

Target variables y



Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
$\vdots$	$\vdots$	$\vdots$

Let’s design a predictor (or hypothesis) with parameter $\boldsymbol{\theta}$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = \sum_{i=0}^{n} \theta_i x_i = \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}$$

Least Mean Square

Supervised learning: Given a training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, predict a parameter $\boldsymbol{\theta}$

LSM (Least Mean Square) algorithm

- Define a cost function and find $\boldsymbol{\theta}$ that minimizes it

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

- Gradient decent: repeatedly update the parameter as follows

$$\theta_j := \theta_j - \varepsilon \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \quad \text{where } \varepsilon\text{: learning rate}$$

- Derive a partial derivative for a single training sample

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{2}(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2$$

Least Mean Square

LSM (Least Mean Square) algorithm

- Derive a partial derivative for a single training sample

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2$$

$$= 2 \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \frac{\partial}{\partial \theta_j} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)$$

$$= \ (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \frac{\partial}{\partial \theta_j} (\sum_{i=0}^{n} \theta_i x_i - y)$$

$$= \ (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) x_j$$

- Then the gradient update is

$$\theta_j := \theta_j - \varepsilon (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) x_j = \theta_j + \varepsilon (y - h_{\boldsymbol{\theta}}(\boldsymbol{x})) x_j$$

Least Mean Square

Batch gradient descent

- Consider all training examples for a single parameter update

```
Repeat until convergence {
    $\theta_j := \theta_j + \varepsilon \sum_{i=1}^{m} (y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))x_j^{(i)}$
}
```

Stochastic gradient descent

- Gradient update w.r.t a single training example

```
Repeat until convergence {
    for $i = 1: m$ {
        $\theta_j := \theta_j + \varepsilon(y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))x_j^{(i)}$
    }
}
```

Outline

- Linear Regression
  - Least Mean Square algorithm
  - Normal equations
  - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

(Reference: Andrew Ng’s CS229 Lecture Notes)

Another Approach to a Solution

Minimize the cost function by explicitly taking its derivatives w.r.t $\boldsymbol{\theta}$ and setting them to zero

- No iterative algorithm

Some basics of linear algebra are required…

- Matrix derivatives
- Trace

Matrix Derivatives

Suppose a function $f\colon \mathbb{R}^{m\times n} \to \mathbb{R}$

- Takes a matrix $A \in \mathbb{R}^{m\times n}$ and returns a real number

Derivative of f w.r.t A

$$\nabla_A f(A) = \begin{pmatrix} \partial f/\partial A_{11} & \cdots & \partial f/\partial A_{1n} \\ \vdots & \ddots & \vdots \\ \partial f/\partial A_{m1} & \cdots & \partial f/\partial A_{mn} \end{pmatrix}$$

An example

- $A = \begin{bmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{bmatrix}$ and $f\colon \mathbb{R}^{2\times 2} \to \mathbb{R}$ is $f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$
- Then $\nabla_A f(A) = \begin{bmatrix} 3/2 & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$

Trace

The sum of the diagonal entries of a square matrix A

$$\mathrm{tr}\,A = \sum_{i=1}^{n} A_{ii}$$

Basic properties

- $\mathrm{tr}\,ABC = \mathrm{tr}\,CAB = \mathrm{tr}\,BCA$ (commutative)
- $\mathrm{tr}\,A$ = $\mathrm{tr}\,A^{\mathrm{T}}$, $\mathrm{tr}\,(A+B)$ = $\mathrm{tr}\,A + \mathrm{tr}\,B$, $\mathrm{tr}\,aA = a\,\mathrm{tr}\,A$

Properties of matrix derivatives

- $\nabla_A\,\mathrm{tr}AB = B^T$, $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$
- $\nabla_A\,\mathrm{tr}ABA^TC = CAB + C^TAB^T$, $\nabla_A|A| = |A|(A^{-1})^T$
- $\nabla_{A^T}\mathrm{tr}ABA^TC = B^TA^TC^T + BA^TC$ from the 2nd and the 3rd

Least Squares Revisited

Represent a training set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ in a matrix form

$$\boldsymbol{X} = \begin{bmatrix} (\boldsymbol{x}^{(1)})^T \\ (\boldsymbol{x}^{(2)})^T \\ \vdots \\ (\boldsymbol{x}^{(m)})^T \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \qquad \boldsymbol{X\theta} - \boldsymbol{y} = \begin{bmatrix} (x^{(1)})^T\boldsymbol{\theta} - y^{(1)} \\ (x^{(2)})^T\boldsymbol{\theta} - y^{(2)} \\ \vdots \\ (x^{(m)})^T\boldsymbol{\theta} - y^{(m)} \end{bmatrix}$$

Cost function

$$J(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{m}(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2 = \frac{1}{2}(\boldsymbol{X\theta} - \boldsymbol{y})^T(\boldsymbol{X\theta} - \boldsymbol{y})$$

Minimize the cost by explicitly taking its derivatives w.r.t $\boldsymbol{\theta}$ and setting them to zero

Least Squares Revisited

Find $\boldsymbol{\theta}$ that sets the derivatives to zero

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \frac{1}{2} (\boldsymbol{X\theta} - \boldsymbol{y})^T (\boldsymbol{X\theta} - \boldsymbol{y}) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} (\boldsymbol{\theta}^T \boldsymbol{X}^T \boldsymbol{X\theta} - \boldsymbol{\theta}^T \boldsymbol{X}^T \boldsymbol{y} - \boldsymbol{y}^T \boldsymbol{X\theta} + \boldsymbol{y}^T \boldsymbol{y}) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \mathrm{tr}(\boldsymbol{\theta}^T \boldsymbol{X}^T \boldsymbol{X\theta} - \boldsymbol{\theta}^T \boldsymbol{X}^T \boldsymbol{y} - \boldsymbol{y}^T \boldsymbol{X\theta} + \boldsymbol{y}^T \boldsymbol{y}) \quad (\because a = \mathrm{tr}\, a) \\
&= \frac{1}{2} \nabla_{\boldsymbol{\theta}} \, (\mathrm{tr}\, \boldsymbol{\theta}^T \boldsymbol{X}^T \boldsymbol{X\theta} - 2\, \mathrm{tr}\, \boldsymbol{y}^T \boldsymbol{X\theta}) \quad (\because \mathrm{tr}\, A = \mathrm{tr}\, A^T, \nabla_{\boldsymbol{\theta}} \boldsymbol{y}^T \boldsymbol{y} = 0) \\
&= \frac{1}{2} (\boldsymbol{X}^T \boldsymbol{X\theta} + \boldsymbol{X}^T \boldsymbol{X\theta} - 2\, \boldsymbol{X}^T \boldsymbol{y}) \quad (\because \nabla_{A^T} \mathrm{tr} ABA^T C = B^T A^T C^T + BA^T C) \\
&= \boldsymbol{X}^T \boldsymbol{X\theta} - \boldsymbol{X}^T \boldsymbol{y} = 0
\end{aligned}
$$

The solution is $\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

Outline

- Linear Regression
  - Least Mean Square algorithm
  - Normal equations
  - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

(Reference: Andrew Ng’s CS229 Lecture Notes)

Probabilistic Interpretation

Assume that the target variables and inputs are related as

$$y^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} + \epsilon^{(i)}$$

- $\epsilon^{(i)}$: an error term of random noise
- $\epsilon^{(i)} \sim N(0, \sigma^2)$ is often assumed to be distributed i.i.d accord to a Gaussian distribution

$$P\left(\epsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(\epsilon^{(i)})^2}{2\sigma^2})$$

- This implies that

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

Likelihood

Given a training set $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, What is its probability for a fixed value of $\boldsymbol{\theta}$?

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{m} P\left(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right)$$

$$= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)})^2}{2\sigma^2})$$

Maximum likelihood: find $\boldsymbol{\theta}$ that maximizes the likelihood

- Instead, maximize log-likelihood $\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta})$
- $\log_e()$ is a strictly increasing function

Likelihood

Maximize log likelihood $\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta})$

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)})^2}{2\sigma^2})$$

$$= \sum_{i=1}^{m} \log(\frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)})^2}{2\sigma^2}))$$

$$= m \log \frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)})^2$$

Maximizing log likelihood corresponds to minimize

$$\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)})^2$$

Finding MLE solution of $\boldsymbol{\theta}$ = solving original least square regression

Outline

- Linear Regression
  - Least Mean Square algorithm
  - Normal equations
  - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

(Reference: Andrew Ng's CS229 Lecture Notes)

Logistic Regression

Now turn our attention to a binary classification problem

- In our dataset $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$, $y^{(i)} = \{0,1\}$ has a binary value

Based on linear regression, only a single modification is to apply a logistic (aka sigmoid) function g

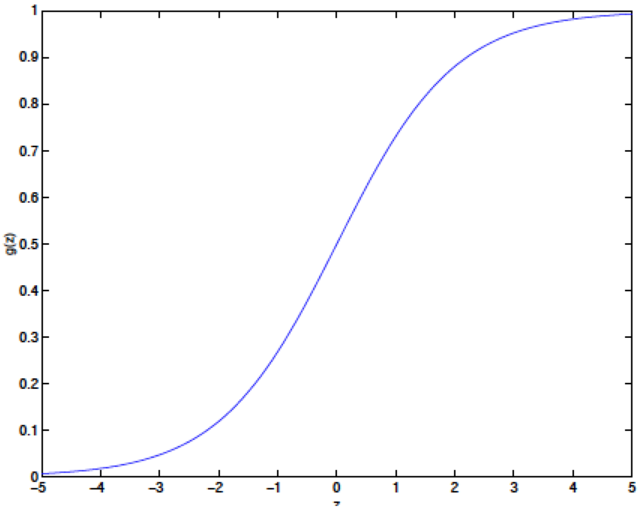
- Change a hypothesis of LR $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$ to

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \boldsymbol{x})}$$

- Logistic (sigmoid) function $g(z) = \frac{1}{1+\exp(-z)}$

Logistic (Sigmoid) Function

$g(z) \to 0$ as $z \to -\infty$ and $g(z) \to 1$ as $z \to \infty$



Derivative of sigmoid function

$$g'(z) = \frac{d}{dz}\frac{1}{1+\exp(-z)} \quad = \frac{1}{(1+\exp(-z))^2}\exp(-z)$$

$$= \frac{1}{1+\exp(-z)}(1 - \frac{1}{1+\exp(-z)}) \ = g(z)(1 - g(z))$$

A Solution to Logistic Regression

Use maximum likelihood formulation

$$P(y = 1|\boldsymbol{x}; \boldsymbol{\theta}) = h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

$$P(y = 0|\boldsymbol{x}; \boldsymbol{\theta}) = 1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

- A single formula

$$P(y|\boldsymbol{x}; \boldsymbol{\theta}) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}))^{y}(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}))^{1-y}$$

- Likelihood

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{m} P\left(y^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) = \prod_{i=1}^{m} (h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right))^{y^{(i)}}(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right))^{1-y^{(i)}}$$

- Log-likelihood

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{m} y^{(i)} \log h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right))$$

A Solution to Logistic Regression

Take derivatives of log-likelihood (for a single sample)

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} y \log g(\boldsymbol{\theta}^T \boldsymbol{x}) + (1-y) \log(1 - g(\boldsymbol{\theta}^T \boldsymbol{x}))$$

$$= (y \frac{1}{g(\boldsymbol{\theta}^T \boldsymbol{x})} - (1-y) \frac{1}{1-g(\boldsymbol{\theta}^T \boldsymbol{x})}) \frac{\partial}{\partial \theta_j} g(\boldsymbol{\theta}^T \boldsymbol{x})$$

$$= (y \frac{1}{g(\boldsymbol{\theta}^T \boldsymbol{x})} - (1-y) \frac{1}{1-g(\boldsymbol{\theta}^T \boldsymbol{x})}) \, g(\boldsymbol{\theta}^T \boldsymbol{x})(1 - g(\boldsymbol{\theta}^T \boldsymbol{x})) \frac{\partial}{\partial \theta_j} \boldsymbol{\theta}^T \boldsymbol{x}$$

$$= (y(1 - g(\boldsymbol{\theta}^T \boldsymbol{x})) - (1-y) g(\boldsymbol{\theta}^T \boldsymbol{x})) x_j$$

$$= (y - g(\boldsymbol{\theta}^T \boldsymbol{x})) x_j$$

A Solution to Logistic Regression

As a result, the stochastic gradient rule is

$$\theta_j \coloneqq \theta_j + \alpha(y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))x_j^{(i)}$$

- cf) the stochastic gradient rule of linear regression

$$\theta_j \coloneqq \theta_j + \alpha(y^{(i)} - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}))x_j^{(i)}$$

- Only difference is $g(\boldsymbol{\theta}^T\boldsymbol{x})$ for logistic regression and $\boldsymbol{\theta}^T\boldsymbol{x}$ for linear regression

- Can be generalized to GLMs (Generalized Linear Models)

Outline

- Linear Regression
  - Least Mean Square algorithm
  - Normal equations
  - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

(Reference: Andrew Ng's CS229 Lecture Notes and UFLDL Tutorial)

Softmax Regression

Generalize the binary classification of logistic regression to a multiple classification problem

- Replace $y = \{0,1\}$ to $y = \{1,2, \dots, k\}$

Review two discrete distributions

- Binomial distribution: the probability of getting exactly k heads in n coin flips with p of head probability

$$f(k; n, p) = \frac{n!}{k!\,(n-k)!} p^k (1-p)^{n-k}$$

- Multinomial distribution: Given that we extract n balls of k different colors with the probability of $\{p_1, p_2, \dots, p_k\}$, the probability of getting $\{x_1, x_2, \dots, x_k\}$ number of balls

$$f(x_1, \dots, x_k; n, p_1, \dots, p_k) = \frac{n!}{x_1! \dots x_k!} {p_1}^{x_1} \dots {p_k}^{x_k}$$

Softmax Function

cf) Logistic function as hypothesis of logistic regression

$$P(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x})} \qquad P(y=0|\boldsymbol{x};\boldsymbol{\theta}) = \frac{\exp(-\boldsymbol{\theta}^T\boldsymbol{x})}{1+\exp(-\boldsymbol{\theta}^T\boldsymbol{x})}$$

Use softmax function as a hypothesis of softmax regression

- A generalization of the logistic function

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} P(y=1|\boldsymbol{x};\boldsymbol{\theta}) \\ \vdots \\ P(y=k|\boldsymbol{x};\boldsymbol{\theta}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k}\exp(\boldsymbol{\theta}_j^T\boldsymbol{x})} \begin{bmatrix} \exp(\boldsymbol{\theta}_1^T\boldsymbol{x}) \\ \vdots \\ \exp\left(\boldsymbol{\theta}_k^T\boldsymbol{x}\right) \end{bmatrix}$$

- A set of parameters $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k\}$
- Softmax is over-parameterized (one parameter is redundant)
- i.e. subtracting ψ from every $\boldsymbol{\theta}_j$ does not affect our prediction
- e.g. a logistic function: $\{\boldsymbol{0}, \boldsymbol{\theta}\}$

Relation to Logistic Regression

A softmax with $k = 2$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{\exp(\boldsymbol{\theta}_1^T\boldsymbol{x}) + \exp(\boldsymbol{\theta}_2^T\boldsymbol{x})}\begin{bmatrix}\exp(\boldsymbol{\theta}_1^T\boldsymbol{x})\\ \\ \exp(\boldsymbol{\theta}_2^T\boldsymbol{x})\end{bmatrix}$$

- Let $\psi = \boldsymbol{\theta}_1$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{\exp(\mathbf{0}^T\boldsymbol{x}) + \exp((\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^T\boldsymbol{x})}\begin{bmatrix}\exp(\mathbf{0}^T\boldsymbol{x})\\ \\ \exp((\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1)^T\boldsymbol{x})\end{bmatrix}$$

$$= \frac{1}{1 + \exp(\boldsymbol{\theta}^T\boldsymbol{x})}\begin{bmatrix}1\\ \vdots\\ \exp(\boldsymbol{\theta}^T\boldsymbol{x})\end{bmatrix} = \begin{bmatrix}\dfrac{1}{1 + \exp(\boldsymbol{\theta}^T\boldsymbol{x})}\\ \\ 1 - \dfrac{1}{1 + \exp(\boldsymbol{\theta}^T\boldsymbol{x})}\end{bmatrix}$$

A Solution to Softmax Regression

Derive the log-likelihood

- Start from the hypothesis function

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} P(y=1|\boldsymbol{x};\boldsymbol{\theta}) \\ \vdots \\ P(y=k|\boldsymbol{x};\boldsymbol{\theta}) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} \exp(\boldsymbol{\theta}_j^T \boldsymbol{x})} \begin{bmatrix} \exp(\boldsymbol{\theta}_1^T \boldsymbol{x}) \\ \vdots \\ \exp(\boldsymbol{\theta}_k^T \boldsymbol{x}) \end{bmatrix}$$

- The definition of log-likelihood

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{m} \log(P(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta}))$$

$$= \sum_{i=1}^{m} \log \prod_{j=1}^{k} \left(\frac{\exp(-\boldsymbol{\theta}_j^T \boldsymbol{x}^{(i)})}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})} \right)^{1\{y^{(i)}=j\}}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log \frac{\exp(-\boldsymbol{\theta}_j^T \boldsymbol{x}^{(i)})}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})}$$

A Solution to Softmax Regression

Now take the derivative of the log-likelihood

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} \log \frac{\exp(-\boldsymbol{\theta}_j^T \boldsymbol{x}^{(i)})}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})}$$

$$= -\sum_{i=1}^{m} \sum_{j=1}^{k} 1\big\{y^{(i)} = j\big\} (\boldsymbol{\theta}_j^T \boldsymbol{x}^{(i)} - \log(\sum_{l=1}^{k} \exp\big(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)}\big)))$$

- Take derivative

$$\nabla_{\boldsymbol{\theta}_n} \ell(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \sum_{j=1}^{k} 1\big\{y^{(i)} = j\big\} \; (\boldsymbol{x}^{(i)} 1\{j = n\} - \frac{\boldsymbol{x}^{(i)} \exp\big(-\boldsymbol{\theta}_n^T \boldsymbol{x}^{(i)}\big)}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})})$$

$$= -\sum_{i=1}^{m} \boldsymbol{x}^{(i)} (\sum_{j=1}^{k} 1\big\{y^{(i)} = j\big\} 1\{j = n\} - \sum_{j=1}^{k} 1\big\{y^{(i)} = j\big\} \frac{\exp\big(-\boldsymbol{\theta}_n^T \boldsymbol{x}^{(i)}\big)}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})})$$

$$= -\sum_{i=1}^{m} \boldsymbol{x}^{(i)} (1\big\{y^{(i)} = n\big\} - \frac{\exp\big(-\boldsymbol{\theta}_n^T \boldsymbol{x}^{(i)}\big)}{\sum_{l=1}^{k} \exp(-\boldsymbol{\theta}_l^T \boldsymbol{x}^{(i)})})$$

A Solution to Softmax Regression

Finally we have

$$\nabla_{\boldsymbol{\theta}_n}\ell(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \boldsymbol{x}^{(i)}(1\{y^{(i)} = n\} - \frac{\exp\left(-\boldsymbol{\theta}_n^T\boldsymbol{x}^{(i)}\right)}{\sum_{l=1}^{k}\exp(-\boldsymbol{\theta}_l^T\boldsymbol{x}^{(i)})})$$

- Note that $\nabla_{\boldsymbol{\theta}_n}\ell(\boldsymbol{\theta})$ is a vector

The gradient descent is to update

$$\boldsymbol{\theta}_n \coloneqq \boldsymbol{\theta}_n - \alpha\nabla_{\boldsymbol{\theta}_n}\ell(\boldsymbol{\theta}) \quad \text{for } n = 1, \dots, k$$

Outline

- Linear Regression
    - Least Mean Square algorithm
    - Normal equations
    - Probabilistic Interpretation
- Logistic Regression
- Softmax Regression
- Generalized Linear Models

(Reference: Andrew Ng's CS229 Lecture Notes and UFLDL Tutorial)

Generalized Linear Models

Regression and classification models have similar forms

- $p(y|\boldsymbol{x}, \boldsymbol{\theta}) \sim N(\mu, \sigma^2)$: linear regression
- $p(y|\boldsymbol{x}, \boldsymbol{\theta}) \sim Bernoulli(\phi)$: logistic regression
- $p(y|\boldsymbol{x}, \boldsymbol{\theta}) \sim Multinoulli(\boldsymbol{\phi})$: softmax regression

A generalized class of regression and classification models

Exponential Family

A class of distribution is in the exponential family if it can be written in the form

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

- η: the natural (or canonical) parameter
- $T(y)$: sufficient statistics
- $a(\eta)$: partition function ($\exp a(\eta)$ works as a normalizer to make the integration of $p(y;\eta)$ to 1)

Bernoulli Distribution is in Exponential Family

The Bernoulli distribution is

$$
\begin{aligned}
p(y;\phi) &= \phi^{y}(1-\phi)^{1-y} \\
&= \exp(y\log(\phi) + (1-y)\log(1-\phi))) \\
&= \exp(\log\left(\frac{\phi}{1-\phi}\right)y + \log(1-\phi))
\end{aligned}
$$

In an exponential family form

$$p(y;\eta) = b(y)\exp(\eta^{T}T(y) - a(\eta))$$

- $\eta = \log(\frac{\phi}{1-\phi})$
- $T(y) = y$
- $a(\eta) = -\log(1-\phi\ ) = \log(1+\exp(\eta))$
- $b(y) = 1$

Gaussian Distribution is in Exponential Family

The Gaussian distribution is

$$p(y;\phi) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}(y-\mu)^2)$$

$$= \frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}y^2)\ \exp(\mu y - \frac{1}{2}\mu^2)$$

In an exponential family form

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta))$$

- $\eta = \mu$
- $T(y) = y$
- $a(\eta) = \mu^2/2 = \eta^2/2$
- $b(y) = \left(\frac{1}{\sqrt{2\pi}}\right)\exp(-y^2/2)$

How to Construct GLMs

Want to predict some random variable y as a function of $\boldsymbol{x}$

Three steps to construct GLMs

- 1. $y|\boldsymbol{x};\boldsymbol{\theta} \sim$ *ExponentialFamily*(η): given $\boldsymbol{x}$ and $\boldsymbol{\theta}$, we assume the response y follows an exponential family distribution with parameter η
- 2. Our goal is to predict the expected value of $T(y)$ given $\boldsymbol{x}$ (In many cases, $T(y) = y$)
- 3. Natural parameter η and input $\boldsymbol{x}$ are linearly related: $\eta = \boldsymbol{\theta}^T \boldsymbol{x}$

Examples

Ordinary linear regression

- 1. $y|\boldsymbol{x};\boldsymbol{\theta}\sim N(\mu,\sigma^2)$ where $\eta=\mu$
- 2. $T(y)=y$ and our goal is to predict $E[y|\boldsymbol{x}]=\mu$
- 3. $\eta=\boldsymbol{\theta}^T\boldsymbol{x}$

Logistic Regression

- 1. $y|\boldsymbol{x};\boldsymbol{\theta}\sim Bernoulli(\phi)$ where $\eta=\log(\frac{\phi}{1-\phi})$ or $\phi=1/(1+\exp(-\eta))$
- 2. $T(y)=y$ and our goal is to predict $E[y|\boldsymbol{x}]=\phi$
- 3. $\eta=\boldsymbol{\theta}^T\boldsymbol{x}$

Softmax Regression (Do it by yourself!)