Lecture1. Learning from data

Outline

Introduction

Learning from Data
Problem Setup
A Simple Learning Model
Perceptron

Types of Learning

Summary

- Learning from data

  1) data

  2) pattern

  3) 수학적인 분석 불가능

- Learning model

  hypothesis set + learning algorithm

  - 이때, hypothesis set H를 functional form h(x)로 나타낸다 : parametric model

- Perceptron

  h(x) = sign (wTx)

- PLA : Perceptron learning algorithm

  iterative, converge in linearly separable data

  w(t+1) = w(t) = yx ,이 때 (x,y)는 misclassified point

- Learning의 종류

  1) Supervised learning : (input, correct output) :classification, regression

2) Reinforcement learning : (input, some output, grade) : game learning

3) Unsupervised learning : (input) : clustering, HMMs, feature extraction(PCA, ICA, SVD)

: finding patterns, precursor to supervised learning, to create high level representation

=> 우리는 supervised learning에 대해서 이야기한다.

Summary

- learning is used when
  - we have data, and a pattern exists
  - but we can hardly pin it down mathematically
- learning model: hypothesis set and learning algorithm
  - our first example: perceptron and PLA
- types of machine learning

  supervised: (input, correct output/label)

  unsupervised: (input, no label)

  reinforcement: (input, some output, grade for this output)
- supervised learning: our main theme for a while
  - unknown target function $y = f(\mathbf{x})$
  - known training data set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
  - learning algorithm picks $g \approx f$ from hypothesis set $\mathcal{H}$

Lecture2. Feasibility of learning

Outline

Learning Unknown Target Function
- Hoeffding Inequality
- Connection to Learning
- Generalizing the Hoeffding Bound

Feasibility of Learning

Error and Noise
- Error Measures
- Noisy Targets

Summary

- Hoeffding inequality

  $\mathbb{P}[|\hat{\mu} - \mu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$

  outside D 에 위치한 f 를 probabilistic way 로 learn

  u(out of sample error)=u^(in sample error) is PAC (probably approximately correct)

- Multiple hypothesis

  H (hypothesis set)에 여러 개의 h 가 있는 경우

  $\mathbb{P}[|\mathrm{E}_{\mathrm{in}}(g) - \mathrm{E}_{\mathrm{out}}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$: M 배 looser, M must be finite

- Feasibility of learning

  1. Eout(g) ~ Ein(g) 라고 할 수 있는가?

     Hoeffding ineq, VC analysis, bias-variance analysis

  2. Ein(g)를 충분히 작게 만들 수 있는가?

     various learning paradigms will help

  - complexity of H = M : M 이 크면, fits data well so smaller Ein(g), but Ein !~ Eout more

- complexity of f : f가 복잡하면 hard to fit, higher Ein(g)

- Error and Noise (다르다)

  Error: point-wise error 의 평균(Ein), 또는 Expectation(Eout)

  문제에 따라 error 다르게 정의 but

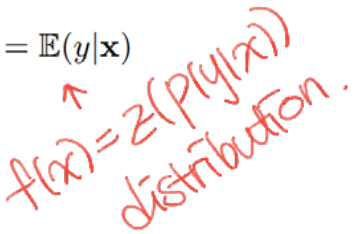
  1) plausible measures : squared error, Gaussian noise

  2) friendly measures : closed-form solution, convex optimization

  Noise : f is not uniquely determined by the input : f가 conditional probability

  regression (noise is added), rather than interpolation

  deterministic target as the average of noisy target : $f(\mathbf{x}) = \mathbb{E}(y|\mathbf{x})$



- P(y|x) and P(x)

  P(y|x) : target distribution

  P(x) : input distribution

  P(x,y) = P(y|x)P(x) : model (x,y)s are generated by P(x,y)

Summary

- learning *unknown* target f: only possible in a probabilistic sense
  - we need hypothesis $g \approx f$, which means $\mathrm{E}_{\mathrm{out}}(g) \approx 0$
- feasibility of learning is split into two questions
  1. can we make sure that $\mathrm{E}_{\mathrm{out}}(g) \approx \mathrm{E}_{\mathrm{in}}(g)$?
  2. can we make $\mathrm{E}_{\mathrm{in}}(g) \approx 0$?
- answering Q1: theoretical (first half of this course)
  - Hoeffding's inequality: $\mathbb{P}[|\mathrm{E}_{\mathrm{in}}(g) - \mathrm{E}_{\mathrm{out}}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$
  - VC analysis and bias-variance analysis
- answering Q2: practical (second half)
  - various learning paradigms will help us achieve that

- error measure quantifies the meaning of $\approx$ in expression $g \approx f$
  - ideally, should be specified by the user
  - practical alternatives: *plausible*/*friendly* measures

- averaging *pointwise* error gives
  - in-sample: $\mathrm{E}_{\mathrm{in}}(h) = \frac{1}{N}\sum_n \mathrm{e}(h(\mathbf{x}_n), f(\mathbf{x}_n))$
  - out-of-sample: $\mathrm{E}_{\mathrm{out}}(h) = \mathbb{E}_{\mathbf{x}}[\mathrm{e}(h(\mathbf{x}), f(\mathbf{x}))]$

- noisy targets
  - $y \sim P(y|\mathbf{x})$ instead of $y = f(\mathbf{x})$ [y is a random variable]
  - $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ generated by $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$
  - $\mathrm{E}_{\mathrm{out}}(h) = \mathbb{E}_{\mathbf{x},y}[\mathrm{e}(h(\mathbf{x}), y)]$

noisy

Lecture3. Linear classification and regression

Outline

- Linear model

  small VC dimension : generalize well from Ein to Eout

  use "signal" wTx

- 3 problems

  1) classification

  2) regression

  3) probability estimation (logistic regression)

  - linear classification

    $\mathcal{Y} = \{-1, +1\}$

    $h(\mathbf{x}) = \text{sign}(\mathbf{w}^{\text{T}}\mathbf{x})$

    $\text{e}(\hat{y}, y) = [\![\hat{y} \neq y]\!]$

    ▸ NP-hard in general

  - linear regression

    $\mathcal{Y} = \mathbb{R}$

    $h(\mathbf{x}) = \mathbf{w}^{\text{T}}\mathbf{x}$

    $\text{e}(\hat{y}, y) = (\hat{y} - y)^2$

    ▸ efficient closed-form solution

Linear classification

- Linear model : 1) Ein ~ Eout

  VC dimension, dvc : d+1

  VC generalization bound : (m: effective 한 hypothesis set 의 크기)

  $$E_{out} \leq E_{in} + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

  probability $\geq 1 - \delta$, $m_{\mathcal{H}}(N) \leq N^{d_{VC}} + 1$

  따라서, 높은 확률로 $E_{out}(g) = E_{in}(g) + O\left(\sqrt{\frac{d}{N} \ln N}\right)$ 이므로, large N 에서 Ein~Eout

- Linear model : 2) Ein ~ 0

  1) linearly separable data :

  언제나 Ein(W*)=0 을 만족하는 hypothesis w*가 존재한다. : PLA 등

  2) linearly in-separable data :

  -> line 으로 나눌 수 없거나, outlier(noise)가 존재하는 경우

  많은 경우 Ein 을 매우 작게 만들 수 있다. : pocket algorithm, SVM 등

- Linearly in-separable data 1. 전체적으로 linearly separable 하지만, outlier(noise)가 존재

  minimum Ein 을 가지도록 문제를 풀자.

  $$\min_{\mathbf{w} \in \mathbb{R}^{d+1}} \underbrace{\frac{1}{N} [\![\operatorname{sign}(\mathbf{w}^T \mathbf{x}_n) \neq y_n]\!]}_{E_{in}(\mathbf{w})}$$

  in general, 위의 식은 N P-hard 이므로 approximately minimize Ein

  ex1) Pocket algorithm → $w(t+1) = w(t) + \alpha y$

  run PLA for w(t+1), Ein(w(t+1))계산해서 개선되었으면 w^=w(t+1), 이를 반복

  // Ein(w(t+1)) 의해, much slower than PLA and good Ein 으로 빠르게 수렴한다는 보장 x

  ex2) digit recognition

  raw input space -> feature input space

  feature space : reduce dimension, better handle curse of dimensionality,

  but can cause loss of information

  이후, PLA 나 pocket algorithm 으로 해결

  ① 우리가 고려해야 하는것 지수적으로 증가해.

  ② 전체에서 우리가 아는 부분 비율 지수적으로 감소해.

Linear regression : y is continuous

- Regression : Target distribution rather than function
  statistical method
  assumption : homoscedasticity : (same finite variance for each value of x)

- Parameter estimation : 평가 방법
  1) least squares
  - OLS : ordinary least squares : closed-form solution exist
  - generalized least squares, iteratively reweighted least squares
  2) maximum likelihood
  - ridge , lasso regression
  - least absolute deviation regression

- OLS solution w* : minimizing Ein(w)
  OLS definition : $E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$, $\frac{1}{N}||X\mathbf{w} - \mathbf{y}||^2$ (X는 xnT가 각 행에 위치)
  linear regression 에서, h=wTx, (wTx called signal)

  Input matrix
  w is a vector.

  이때, Ein(w) is continuous, differentiable, convex, ∇Ein(w)=0
  $\nabla E_{in}(\mathbf{w}) = \frac{2}{N}(X^T X\mathbf{w} - X^T\mathbf{y})$,
  즉 normal equation(XTXw=XTy)을 만족하는 w 를 찾아야 한다.

  $X^TXw - X^Ty = 0$.

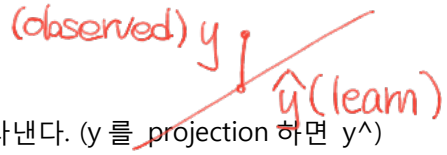
  1) XTX is invertible, w=X#y (X#=(XTX)``XT : pseudo-inverse of X), most cases
  2) XTX is not invertible, X# still defined, but no unique solution
  invertible 하지 않으면 SVD 등 다양한 방법이 존재

  $w = X^+y = (X^TX)^{-1}X^Ty$.
  X^+

- OLS solution (XTX is invertible)
  provides the best linear unbiased estimator (BLUE)
  one step learning, analytic formula

- 참고 SVD

  assume: the SVD of input matrix X is $X = U\Gamma V^{\mathrm{T}}$
  - $U \in \mathbb{R}^{N\times\rho}$ satisfies $U^{\mathrm{T}}U = I_\rho$,
  - $V \in \mathbb{R}^{(d+1)\times\rho}$ satisfies $V^{\mathrm{T}}V = I_\rho$,
  - $\Gamma \in \rho \times \rho$ is a positive diagonal matrix with $\rho = \mathrm{rank}(X)$
  - then $\mathbf{w}_{\mathrm{lin}} = V\Gamma^{-1}U^{\mathrm{T}}\mathbf{y}$ satisfies $X^{\mathrm{T}}X\mathbf{w}_{\mathrm{lin}} = X^{\mathrm{T}}\mathbf{y}$



- Hat matrix H (projection matrix)

  H 는 Ein 과 Eout 의 관계를 나타낸다. (y 를 projection 하면 y^)

  in-sample error 에 의해 y != y^ (y 는 관찰 값, y^은 learn 한 값)

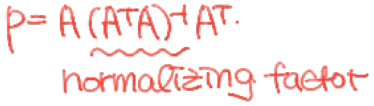
  $\hat{\mathbf{y}} = X\mathbf{w}_{\mathrm{lin}}$, $\hat{\mathbf{y}} = \underbrace{X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}}_{H \in \mathbb{R}^{N\times N}}\mathbf{y}$, H=X(XTX)``XT

  즉 y^ : orthogonal projection of y onto 속 column space of X

- Projection review

  - orthogonal projection in line, 결론 : P=uuT
  - orthonormal basis(u1,...uk), space U, A's column is u1,...uk, 결론 : P=AAT
  - non-orthonormal basis, 결론 : P=A(ATA)``AT

    (ATA)``is normalizing factor, Py=y^



- 참고 H

  HT=H, H2=H, 1-H 역시 projection 이고 H 와는 orthogonal space

  trace(H)=d+1, is sum for diagonal elements in H

- 정리

  input x 로 실수인 y 를 예측한다. 이 때, best x hyper plane 만들자. Error 는 OLS 로.

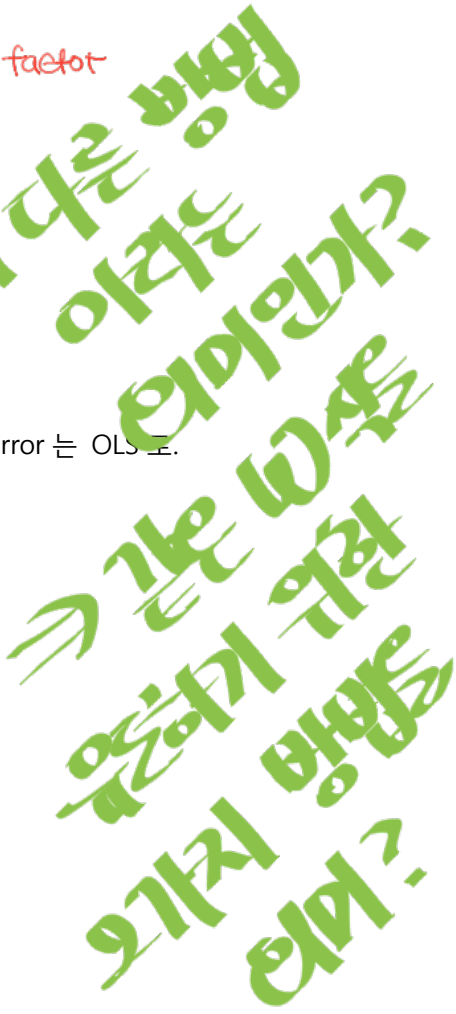
  1) 방법 1 : one-shot : w = X#y = (XTX)``XTy

  2) 방법 2 : hat matrix, 투영과 관찰을 통해 차이가 최소가 되도록

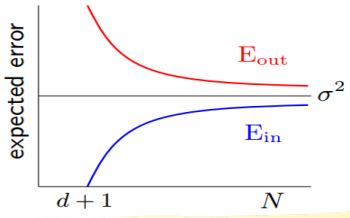
  generalization bound : Eout = Ein + O(d/N),

  if se assume y=f+e, (noise e has 0 mean and finite variance,

  we can also derive exact formulas for Ein and Eout



$$\mathrm{E}_{\text{in}} = \sigma^2\left(1 - \frac{d+1}{N}\right) < \sigma^2$$
$$\mathrm{E}_{\text{out}} = \sigma^2\left(1 + \frac{d+1}{N}\right) > \sigma^2$$



(Ein 은 in-sample noise 까지 같이 학습하기 때문에 수렴값보다 작다)

(Eout 은 w 가 fitted in in-sample error 이므로 수렴값보다 크다)

: 결론 : expected generalization error : O(2(d+1) /N) 이므로 learning is feasible

Summary

- linear models
  - used for classification, regression, probability estimation
  - have small d_{VC} and generalize well $\Rightarrow$ first practical choice
  - use the 'signal': $\sum_{i=0}^{d} w_i x_i = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

- linear classification

$$\mathcal{Y} = \{-1, +1\}$$
$$h(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$
$$\mathrm{e}(\hat{y}, y) = [\![\hat{y} \neq y]\!]$$

  - NP-hard in general

- linear regression

$$\mathcal{Y} = \mathbb{R}$$
$$h(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$
$$\mathrm{e}(\hat{y}, y) = (\hat{y} - y)^2$$

  - efficient closed-form solution

- linear regression algorithm (1-step learning): $\boxed{\mathbf{w} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\mathbf{y}}$

Lecture4. Logistic regression

Outline

- Logistic regression

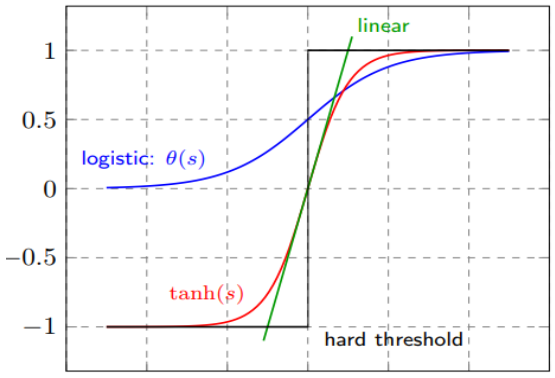
  input : h(x) = θ(wTx), θ is logistic function, sigmoid, soft threshold

  output : real but bounded

  θ 가 어떤 점에서 analytical and computational advantage 를 주는지 생각하면서 공부

  θ 의 특징 : 1- θ(s) = θ(-s)

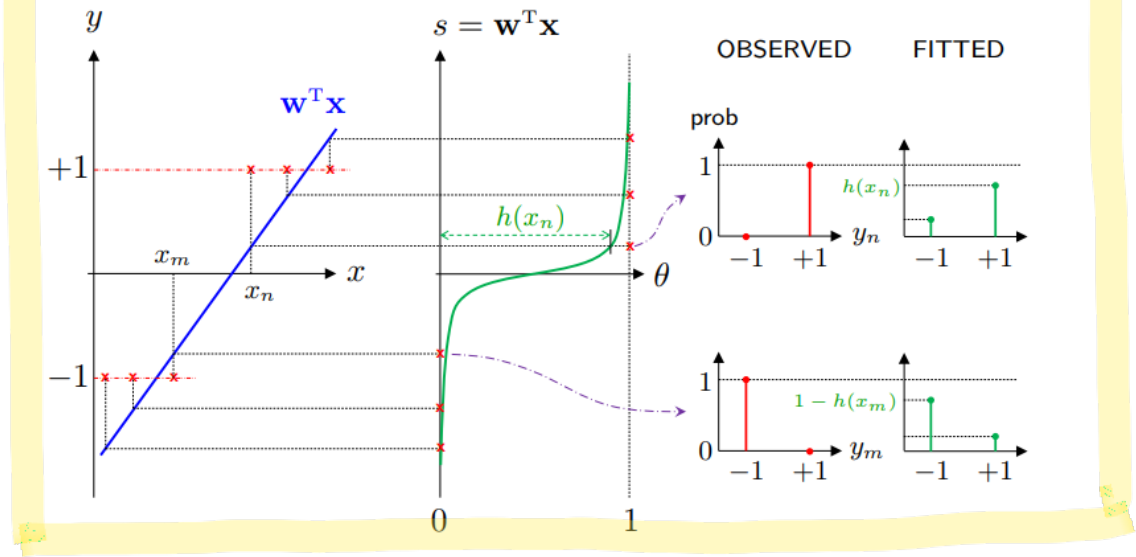
- Sigmoid function comparison



$$\theta(s) = \frac{1}{1+e^{-s}}$$

→ sigmoid function.

- Big picture



$$\mathbb{P}[y = +1|\mathbf{x}] = f(\mathbf{x}) \sim h(\mathbf{x}) = \theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

$$P(y|\mathbf{x}) = h(\mathbf{x})^{[\![y=+1]\!]}(1-h(\mathbf{x}))^{[\![y=-1]\!]} = \theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

$h(x) = \theta(w^Tx) = \frac{1}{1+e^{-w^Tx}}$

$h(-x) = \theta(-w^Tx) = 1-\theta(w^Tx) = 1-h(x)$.

<정리>

$h(x) = \theta(w^Tx)$.

$P(y|x) = \theta(yw^Tx)$

$= h(x)^{[y=1]} + (1-h(x))^{[y=-1]}$

이제 $P(y|x) = \begin{cases} h(x) & \text{if } y=1 \\ 1-h(x) & \text{if } y=-1 \end{cases}$

$\therefore P(y|x) = h(x)^{[y=1]} + (1-h(x))^{[y=-1]}$

$= h(yx) = \theta(yw^Tx)$

$= \frac{1}{1+e^{-yw^Tx}}$ ∴ 증명.

$y=1$: $\frac{1}{1+e^{-w^Tx}}$ $\quad$ $y=-1$: $\frac{1}{1+e^{w^Tx}}$

$= 1 - \frac{1}{1+e^{-w^Tx}} = \frac{e^{-w^Tx}}{1+e^{-w^Tx}}$

$= \frac{1}{1+e^{w^Tx}}$

Logistic regression 의 Error

- Learning target

  f(x) = P[y=+1 | x], y=+1 일 확률, noisy label (단지 1 또는 -1 뿐)

  P(y|x) = f(x) for y=+1

  = 1-f(x) for y=-1

- Error measure

  1) simple error measure $\mathrm{E}_{\mathrm{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}\left(h(\mathbf{x}_n) - \frac{1}{2}(1+y_n)\right)^2$ but very hard to minimize

  2) cross-entropy measure $\mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right)$, (error property and convex)

  - based do intuitive probabilistic interpretation
  - gradient-based optimization 에서 좋은 성질을 가진다.

- Error measure 유도 방법 1: likelyhood

  likelihood : P(y|x) = θ(ywTx)

  (이 때, 1-θ(x)= θ(-x) 임을 이용)

  likelihood of training data : $P(y_1|\mathbf{x}_1)P(y_2|\mathbf{x}_2)\cdots P(y_N|\mathbf{x}_N) = \prod_{n=1}^{N} P(y_n|\mathbf{x}_n)$

  maximum likelihood h 를 select

  수식을 변형 : minimizing $-\frac{1}{N}\ln\left(\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right) = \frac{1}{N}\sum_{n=1}^{N}\ln\frac{1}{P(y_n|\mathbf{x}_n)}$,wrt w

  결론 : in-sample error : $\boxed{\mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right)}$

  P(y|x) = θ(ywTx)를 대입하고, sigmoid 함수를 대입해 정리한 식

  implied point-wise error : ln(1+e^(-ywTx))

- Error measure 유도 방법 2: cross-entropy

  consider two pmf {p, 1-p} and {q, 1-q} with binary outcomes

  cross entropy : (p)log(1/q)+(1-p)log(1/1-q)

  : 'error' for 'observed' pmf {p, 1-p} by 'fitted' pmf {q, 1-q}

$$NLL(\mathbf{w}) \triangleq -\log\left\{\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right\}$$

$$= -\log\left\{\prod_{n=1}^{N} h(\mathbf{x}_n)^{[\![y_n=+1]\!]}(1-h(\mathbf{x}_n))^{[\![y_n=-1]\!]}\right\}$$

$$= \sum_{n=1}^{N}\left\{[\![y_n=+1]\!]\log\frac{1}{h(\mathbf{x}_n)} + [\![y_n=-1]\!]\log\frac{1}{1-h(\mathbf{x}_n)}\right\}$$

: cross entropy error function

nth term is cross entropy for two pmfs measured on (xn, y)

observed : { [y=+1], 1-[y=+1] }

fitted: { h(xn), 1-h(xn) }

이 때, p=[y=+1], q=h(xn)인 cross entropy

유도 과정에서 1-θ(x)= θ(-x)만 이용하여도 전개가 가능하다. Ein 은 다른 식으로 표현 돼.

Logistic regression 의 optimization : iterative (e.g. gradient descent)

- set ∇Ein=0 (in error surface)

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n \mathbf{x}_n}{1+e^{y_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n}}$$

$$= \frac{1}{N}\sum_{n=1}^{N} -y_n \mathbf{x}_n \theta(-y_n \mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$$

no analytic solution -> iterative solution

"gradient descent" : smooth 한 error measure 를 가지는 다양한 것을 train 가능

- Gradient descent : good at logistic regression

general for minimizing twice-differentiable function : logistic and neural network

initial location is crucial

- 1) 어디로? Negative gradient 방향, 2) 얼만큼? η learning rate 만큼

$$\begin{aligned}\Delta \mathrm{E}_{\mathrm{in}} &= \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t+1)) - \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t)) \\ &= \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t)+\eta\hat{\mathbf{v}}) - \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t)) \\ &\approx \eta \nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t))^{\mathrm{T}}\hat{\mathbf{v}} \\ &\geq -\eta ||\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t))||\end{aligned}$$

등호성립 조건.

$$\hat{\mathbf{v}} = -\frac{\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t))}{||\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t))||}$$

ΔEin 이 최소가 되길 바란다.

- Algorithm : gradient descent

for (interatively)

compute ∇Ein(w(t))

set v^ = negative gradient = −∇Ein(w(t)),

- v^is due to cancellation of |∇Ein|

update weights w(t+1) = w(t) - η∇Ein(w(t))

$\hat{v} = -\nabla E_{in}(w(t))$ 를 하고 이제 $w(t+1) = w(t) - \eta\hat{v}$ 이걸 여러번 반복.

- Gradient descent

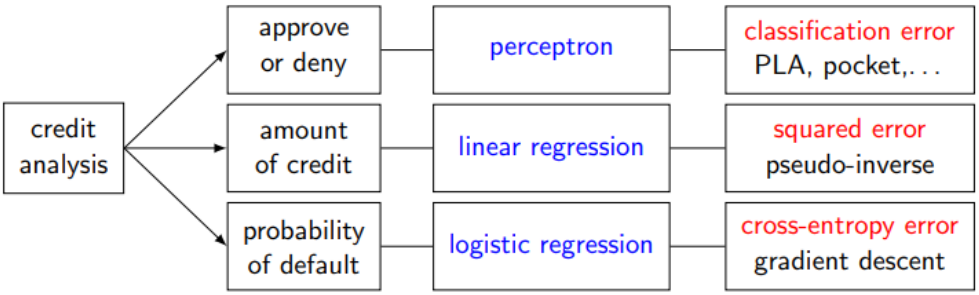
- approximately minimize Ein (convex 이면 정확하게)
- ∇Ein 의 계산이 O(N) time if batch-> 다 볼 필요 없는 mini batch, stochastic 사용하기도
- initialization 은 normal distribution 에서 choose
- when to terminate?

1) upper bound of large iterations

2) small size of gradient

3) error itself is small

Linear model review



(사실, 엄밀한 Perceptron 은 linear classification 에서 쓰이는 unit 을 이야기한다)

	linear classification	linear regression	logistic regression
$\mathcal{Y}$	$\{-1,+1\}$	$\mathbb{R}$	$\{-1,+1\}$
$\hat{y}=h(\mathbf{x})$	$\text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$	$\mathbf{w}^{\mathrm{T}}\mathbf{x}$	$\theta^{\star}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$
$\text{e}(\hat{y},y)$	0-1 loss $[\![\hat{y}\neq y]\!]$	squared error $(\hat{y}-y)^2$	cross-entropy error $[\![y=+1]\!]\ln\frac{1}{\hat{y}}$ $+[\![y=-1]\!]\ln\frac{1}{1-\hat{y}}$
$\text{E}_{\text{in}}(h)$	$\frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n)\neq y_n]\!]$	$\frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n)-y_n)^2$	$\frac{1}{N}\sum_{n=1}^{N}\ln\left(1+e^{-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}\right)$
opt.	combinatorial optimization (NP-hard)	set $\nabla\text{E}_{\text{in}}(\mathbf{w})=0$ (closed-form solution exists)	set $\nabla\text{E}_{\text{in}}(\mathbf{w})=0$ iterative optimization (*e.g.* gradient descent)

$$\frac{1}{N}\sum\left(y\frac{1}{P(y|x)}\right)$$
$$=\frac{1}{N}\sum\left([y=1]\frac{1}{h(x)}+[y=-1]\frac{1}{(1-h(x))}\right)$$

Summary

- logistic regression: estimates probabilities of binary events
  - approximates $\mathbb{P}[y=+1|\mathbf{x}]=f(\mathbf{x})$ by $h(\mathbf{x})=\theta(\mathbf{w}^{\mathrm{T}}\mathbf{x})$
  - likelihood: $P(y|\mathbf{x})=h(\mathbf{x})^{[\![y=+1]\!]}(1-h(\mathbf{x}))^{[\![y=-1]\!]}=\theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x})$
  - training: minimize cross-entropy error iteratively
- gradient descent: iteratively solves unconstrained optimization
  - general technique for minimizing twice-differentiable functions
  - batch gradient descent: $\mathbf{w}\leftarrow\mathbf{w}-\eta\nabla\text{E}_{\text{in}}(\mathbf{w})$ [over all data]
  - stochastic gradient descent: $\mathbf{w}\leftarrow\mathbf{w}-\eta\nabla\text{e}_n(\mathbf{w})$ [pointwise]
  - mini-batch gradient descent, batch normalization
- linear models should be a first try: simple/robust/work & generalize well

Lecture5. Artificial neural networks

Outline

시험: 복잡한 네트워크 정통방 아닌 간단한 방법으로 풀기

- Artificial neural network (ANN) = Multilayer perceptron (MLP)

  Relu : allows spares propagation of activations and gradients

  better for handling vanishing gradient problem

- 분포를 구현하기 위한 것이 함수

	K class	Binary
함수	Softmax function	Logistic function
1 번의 event	멀티눌리 distribution = categorical distribution (주사위 한번 던지는 것)	베르눌리 distribution (동전 한번 던지는 것)
N 번의 event	Multinomial distribution	Binomial distribution

softmax function $\sigma : \mathbb{R}^K \mapsto \mathbb{R}^K$

$$\sigma(\mathbf{h})_j = \frac{e^{h_j}}{\sum_{k=1}^{K} e^{h_k}}$$

- Error measures

Error measures

e_k: error signal on k-th output

$$\underbrace{e_k}_{\text{error}} \triangleq \underbrace{h_k}_{\text{NN output}} - \underbrace{y_k}_{\text{correct label}}$$

$e_k = h_k - y_k$.

$\mathcal{E}_n$: error energy on example $(\mathbf{x}_n, y_n)$ ← sum of squared errors

$$\mathcal{E}_n = \frac{1}{2}\sum_{k=1}^{K} e_{k,n}^2 \qquad \mathcal{E}_n = \frac{1}{2}\sum_{k=1}^{K} e_{k,n}^2$$

$\mathcal{E}_{\mathcal{D}}$: mean-squared error on data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

$$\mathcal{E}_{\mathcal{D}} = \frac{1}{N}\sum_{n=1}^{N} \mathcal{E}_n = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{K} e_{k,n}^2 \qquad \mathcal{E}_{\mathcal{D}} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{k=1}^{K} e_{k,n}^2$$

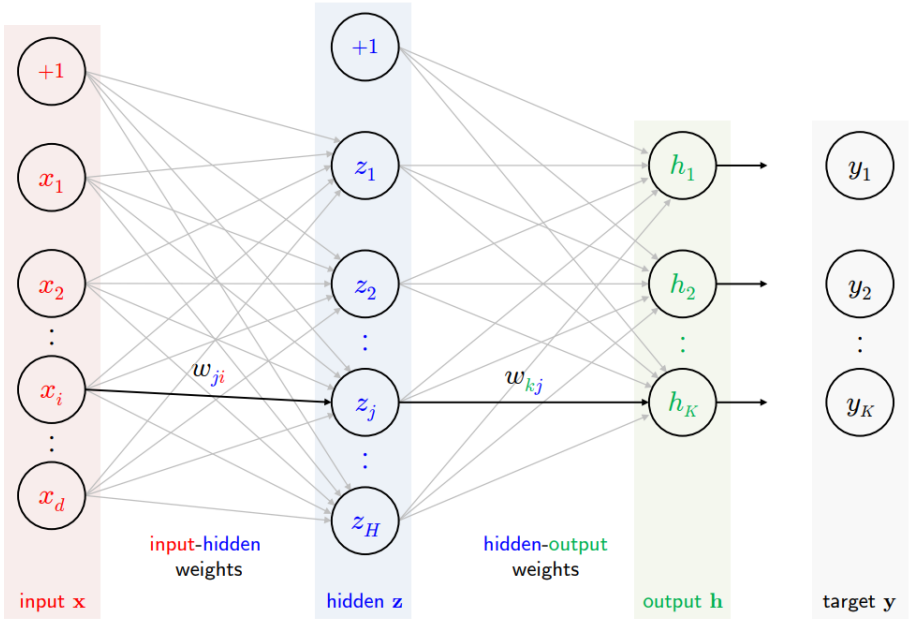
-> 우리는 ED (Ein)을 최소화하는 W 를 찾고 싶다.

- Optimization method

  1) second order method (Newton's method) but not attractive here

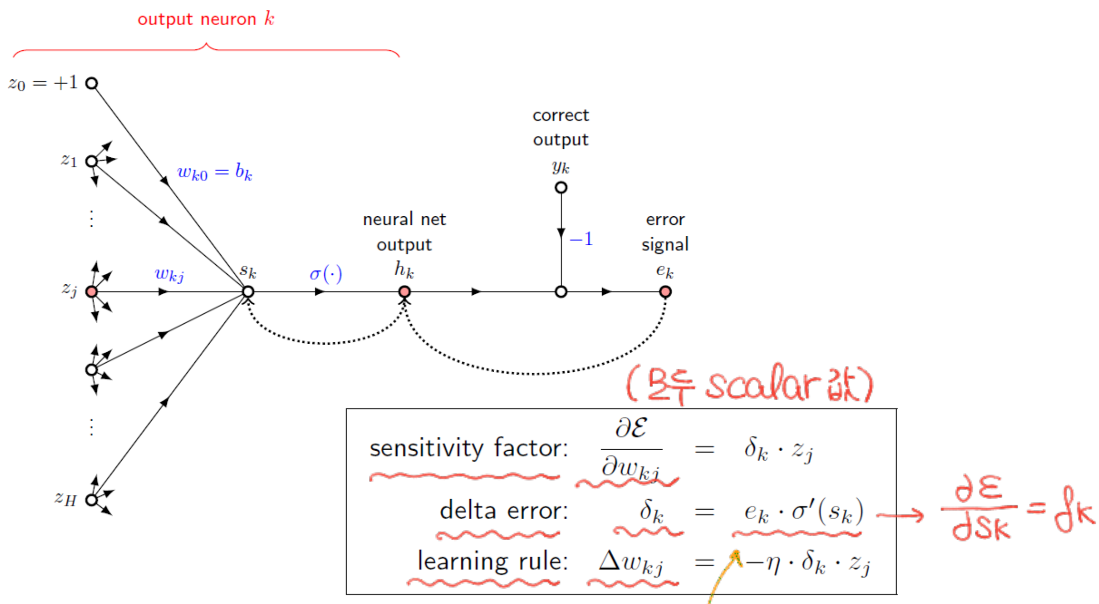
  2) first order method (gradient descent)

- Back propagation



- Back propagation in output layer

Summary: output layer

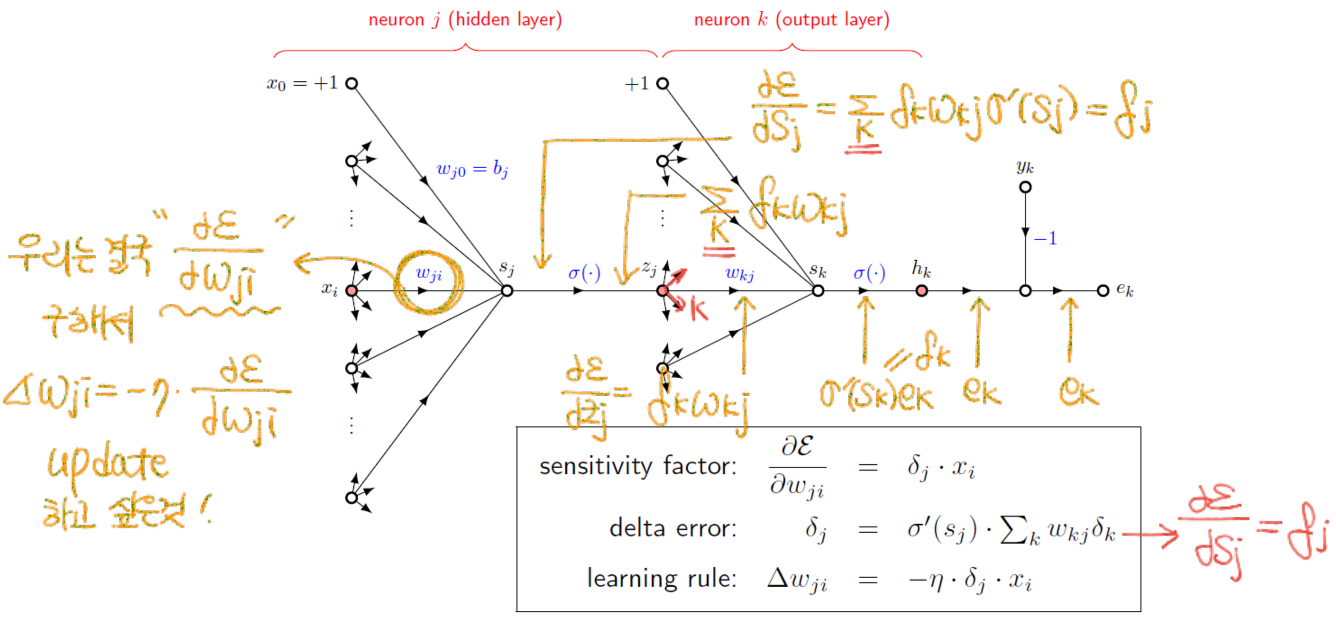


sensitivity factor:	$\frac{\partial \mathcal{E}}{\partial w_{kj}}$	$= \delta_k \cdot z_j$
delta error:	δ_k	$= e_k \cdot \sigma'(s_k)$
learning rule:	Δw_{kj}	$= -\eta \cdot \delta_k \cdot z_j$

(이때, error, input 이 클수록 w 를 많이 변화시킨다)

- Back propagation in input layer

Summary: hidden layer



sensitivity factor:	$\frac{\partial \mathcal{E}}{\partial w_{ji}}$	$= \delta_j \cdot x_i$
delta error:	δ_j	$= \sigma'(s_j) \cdot \sum_k w_{kj}\delta_k$
learning rule:	Δw_{ji}	$= -\eta \cdot \delta_j \cdot x_i$

- Back prop Algorithm

repeat

pick n (하나의 x 를 pick)

forward : compute all z, h

backward : compute all delta

update weights

until it is time to stop, update final weights

* Back Prop 알고리즘

하나의 input을 선택 ~> 여러개의 subset도 가능

forward → z,h 계산

backward → delta 계산

이를 바탕으로 전부 $\frac{\partial \mathcal{E}}{\partial w_{ij}}$ 를 계산 가능.

그래서 weight update → 평균을 구해서 update 값 결정.

이를 여러번 반복

: "전체 N/subset수" 만큼 반복해서 w를 update.

→ 이 전체의 과정을 여러번 반복하는것도 가능.

Back prop 간단한 계산

- 간단한 계산 공식

  1) out = f(in) : f `(in)을 곱한다.

  2) out = in1 * in2 : in1 에 대한 식을 구하는 경우 in2 를 곱한다.

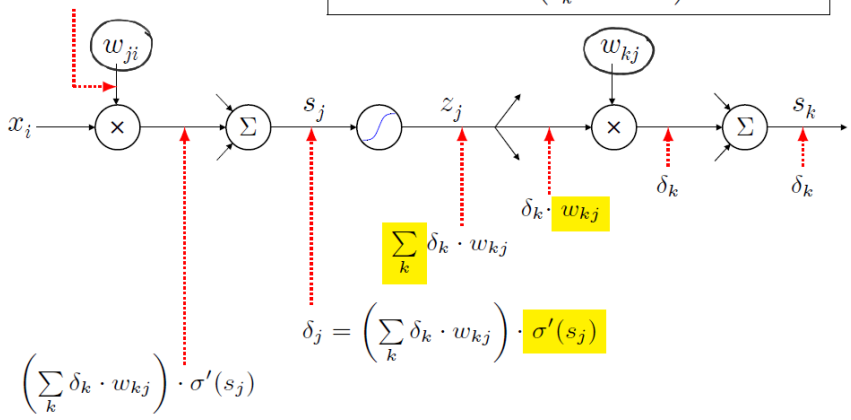
  3) out = sum(in) : 그대로 fanout 한다.

  4) out = max(in) : ini 가 max 이면 fanout, 아니면 0 이다.

  5) out = in ,(fanout) : sum 으로 모두 더한다.

- computing $\frac{\partial \mathcal{E}}{\partial w_{ji}}$

$$\left(\sum_k \delta_k \cdot w_{kj}\right) \cdot \sigma'(s_j) \cdot x_i \implies \boxed{\frac{\partial \mathcal{E}}{\partial w_{ji}} = \delta_j \cdot x_i = \left(\sum_k \delta_k \cdot w_{kj}\right) \cdot \sigma'(s_j) \cdot x_i}$$

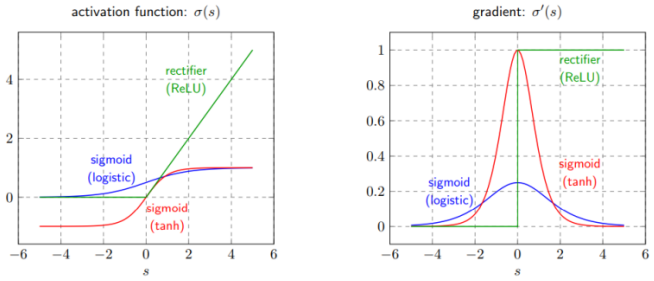


Deep learning 에 대한 간단한 overview

- DL: deep 하게 쌓을 경우 exponential 한 무엇인가를 polynomial 에 나타내는 것이 가능해

- Vanishing gradient problem

  (non-linearity 를 위해서는 양 끝단에서 작동해야 하지만 이때는 gradient 가 죽는다)



Summary

- artificial neural network: universal approximator of functions
  - neuron model: synapse (weights), adder, activation functions
  - conventional: feed-forward multilayer perceptrons (w/ backprop)
  - recent: RBM, AE, CNN, RNN, GAN and many other variants
- hidden layers act as feature detectors
  - # hidden layers ↑ ⟹ intelligence ↑ but training difficulty ↑
  - practical solutions exist for effective training of deep neural networks
- two types of signals for training feed-forward multilayer perceptrons
  1. function signals: forward propagation
  2. error signals: backward propagation

그림에서 x가 2등 layer에 fan out 되어!

$\frac{\partial \mathcal{E}}{\partial w_{ji}} = \delta_j x_i$

$\frac{\partial \mathcal{E}}{\partial w_{kj}} = z_j e_k \theta'(s_k) \delta_k$

w_{ji} $\Delta w_{ji} = -\eta \cdot \delta_j x_i$

w_{kj} $\Delta w_{kj} = -\eta \cdot z_j e_k \theta'(s_k) = -\eta z_j \delta_k$

y_k

x_i Σ s_j z_j Σ s_k h_k e_k

δ_j $\sum_k e_k \theta'(s_k) w_{kj}$ $e_k \theta'(s_k)$ $e_k \theta'(s_k) w_{kj}$ $e_k \theta'(s_k) = \delta_k$ e_k e_k e_k

$\delta_j = \sum_k e_k \theta'(s_k) w_{kj} \theta'(s_j)$

$= \theta'(s_j) \sum_k e_k \theta'(s_k) w_{kj}$

$\mathcal{E} = \frac{1}{2} e_k^2$

$\frac{\partial \mathcal{E}}{\partial e_k} = e_k$에서 시작.

Lecture6. Support vector machines : SVM

Outline

- Discriminant function : g

  - representing classifiers

  - divide the feature space into K decision regions (이 때, reject region 을 두기도)

  g1(vector x) > g2(vector x) : class1 로 분류

  ex) gi(x) = p(Ci|x) : Bayes error 를 minimize

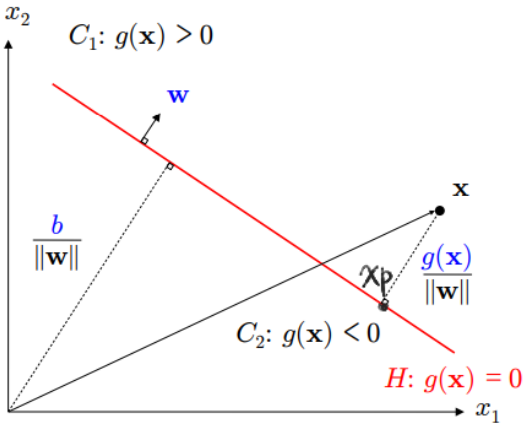
- Linear discriminant function : g(x) = wTx + b

  지금까지는 b 를 x 에 포함하여 같이 보았지만 이제는 따로 생각한다.

  2 category case : g(x)=0 defines decision boundary

  for linear g(x), this decision surface is a hyperplane

- Geometric interpretation
  - big picture:



g(x) gives distance from x to H

$$
\begin{aligned}
g(\mathbf{x}) &= \mathbf{w}^\top \mathbf{x} + b \\
&= \mathbf{w}^\top \left(\mathbf{x}_p + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \right) + b \\
&= (\mathbf{w}^\top \mathbf{x}_p + b) + r\|\mathbf{w}\| \\
&= r\|\mathbf{w}\|
\end{aligned}
$$

proof:

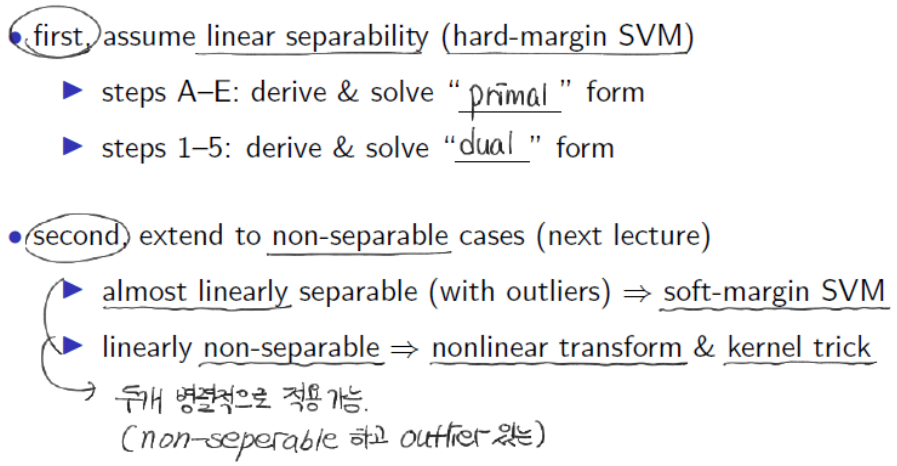
$$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|} = \frac{\mathbf{w}^\top \mathbf{x} + b}{\|\mathbf{w}\|}$$

(이때, w 는 surface 의 방향을 지정, b 는 위치를 지정)

- SVM : maximum margin (hyperplane)classifier

  maximum margin : best generalization performance 을 위해

  앞으로의 overview



- Optimization

  http://www.csc.kth.se/utbildning/kth/kurser/DD3364/Lectures/KKT.pdf

  constraint 가 존재하는 min/max 문제

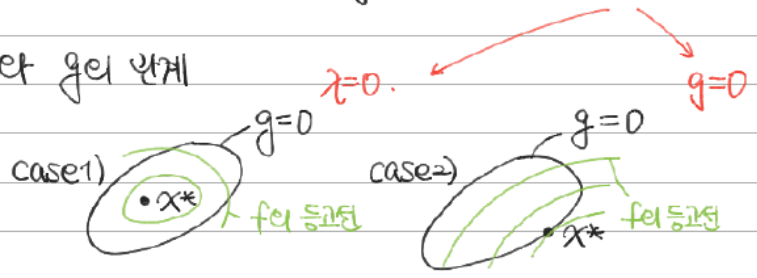
  equality : 라그랑지안, inequality : KKT

최적화에 대한 내용

1) 우리가 풀고자 하는 문제 : max / min f ,
문제의 제약조건 : $g \geq 0$ 또는 $g \leq 0$.
∴ $\mathcal{L} = f \pm \lambda g$ 를 이용해 풀어나간다.

2) $\mathcal{L} = f \pm \lambda g$.
① $\lambda \geq 0$.
② $g \leq 0$ 또는 $g \geq 0$. 문제의 조건
③ $\nabla_x \mathcal{L} = \nabla_x f \pm \lambda \nabla_x g = 0$.
④ $\lambda g = 0$ (이때, 둘다 0은 아니다)

3) f와 g의 관계

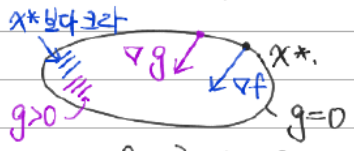


$\lambda = 0$. g는 필요X
g : inactive constraint

$\lambda > 0$. $g = 0$.
g : active constraint

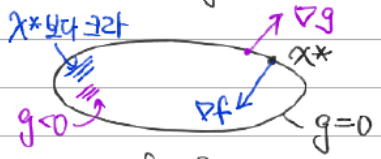
4) 4가지 case.
(f가 max문제, min 문제
g가 $g \geq 0$, $g \leq 0$.

case A) min f. $g \geq 0$.



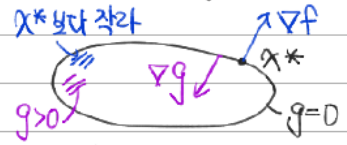
∴ $\nabla f - \lambda \nabla g = 0$

case B) min f. $g \leq 0$



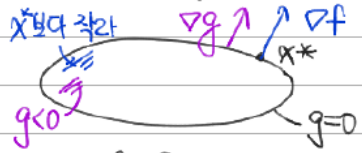
∴ $\nabla f + \lambda \nabla g = 0$

case C) max f. $g \geq 0$.



∴ $\nabla f + \lambda \nabla g = 0$.

case D) max f. $g \leq 0$



∴ $\nabla f - \lambda \nabla g = 0$

Primal form : 이해하기 좋다. But 의미를 파악하기 힘들다.

- Step A

  g(x+) = +1, g(-x) = -1 로 scale 조정

  -> called support vectors

- Step B : setting constraints

  $y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq +1$ (y=+1, -1 의 경우를 모두 포함)

- Step C : determining objective function

  margin : 2/|w|

  therefore, want to minimize |w| -> 편의상 제곱한 것을 minimize

- Step D : SVM optimization problem

  SVM (primal) optimization problem

  $$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq 1, \ \forall t \in [1, N] \end{aligned}$$

  w 에게 convex

  w 에게 linear.

  구하고 싶은것 : w, b ∴ d+1개

  N개 (x_n, y_n)

  - constrained optimization problem

    convex quadratic problem with d+1 parameters, N constraints

  - 특징

    cost function sis convex function of w

    constraints are linear in w

- Step E : QP (Quadratic programming)

  % QP

  $$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{z}^\top Q\mathbf{z} + \mathbf{c}^\top \mathbf{z} \\ \text{subject to} \quad & A\mathbf{z} \geq \mathbf{a} \end{aligned}$$, solution : $\mathbf{z}^* \leftarrow \text{QP}(Q, \mathbf{c}, A, \mathbf{a})$

  solving primal by QP

  $$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq 1, \ \forall t \in [1, N] \end{aligned}$$

$$\mathbf{z} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} \in \mathbb{R}^{d+1}$$

$$\mathbf{w}^\top \mathbf{w} = \begin{bmatrix} b & \mathbf{w}^\top \end{bmatrix} \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} = \mathbf{z}^\top \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \mathbf{z} \Rightarrow Q = \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix}, \mathbf{c} = \mathbf{0}$$

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \equiv \begin{bmatrix} y_n & y_n \mathbf{x}_n^\top \end{bmatrix} \mathbf{z} \geq 1 \Rightarrow \underbrace{\begin{bmatrix} y_1 & y_1 \mathbf{x}_1^\top \\ \vdots & \vdots \\ y_N & y_N \mathbf{x}_N^T \end{bmatrix}}_{A} \mathbf{z} \geq \underbrace{\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}}_{\mathbf{a}}$$

이제, N개의 조건을 한번에 고려.

여기서 QP solver 에 넣을 값을 지정해 줄 수 있어야 한다.

$$A = \begin{bmatrix} y_1 & y_1 x_{1,1} \cdots y_1 x_{1,d} \\ \vdots & \vdots \\ y_N & y_N x_{N,1} \cdots y_N x_{N,d} \end{bmatrix} = \begin{bmatrix} y_1 & y_1 x_1^T \\ \vdots & \\ y_N & y_N x_N^T \end{bmatrix} N$$

$d+1$

- Issues on primal problem

  단점 1)

  d 가 무한으로 가는 경우, 문제를 푸는 것이 불가능하다.

  |w|의 제곱 계산량이 너무 많다.

  단점 2)

  constraint 가 너무 많다. (N 개)

  단점 3)

  insight 를 주지 못한다. Support vector 의 의미를 찾을 수 없다.

* QP에 SVM 대입.

QP 식: $\frac{1}{2} z^T Q z + C^T z$ 를 최적화.

조건: $Az \geq a$.

SVM primal 식: $\frac{1}{2} w^T w$를 최적화.

조건: $y_t(w^T x_t + b) \geq 1, \forall t \in [1..N]$.

만약 조를 다르게 설정한다면 Q.C.A.a가 어떻게 변화하는지 보자.

* $z = \begin{bmatrix} w \\ b \end{bmatrix}$ 라 하자.

1) $\begin{bmatrix} w & b \end{bmatrix} \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = w^T w$

$\begin{bmatrix} w \\ 0 \end{bmatrix}$

$\therefore Q = \begin{bmatrix} I_d & 0 \\ 0 & 0 \end{bmatrix}, c = \mathbb{0}$

2) $N \begin{bmatrix} y_1 x_{1,1} \cdots y_1 x_{d,1} y_1 \\ \vdots \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ b \end{bmatrix} \geq \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$ ($d+1$)

$y_t(w^T x_t + b) \geq 1$.

$\therefore A = \begin{bmatrix} y_1 x_1^T & y_1 \\ \vdots & \\ y_N x_N^T & y_N \end{bmatrix}, a = \mathbb{1}$

Dual form : 이해하기 힘들다. But support vector, kernel trick 에 대한 이해 가능

- Dual formulation

  optimization theory

  if cost, constraint functions are strictly convex, dual form is exist

  and solving dual form is equivalent to solving primal

- Step 1 : compute Lagrangian

  주어진 primal form :

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq 1, \ \forall t \in [1, N] \end{aligned}$$

  위의 primal form 을 constraint 가 없는 Lagrangian optimization 문제로 변환 :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha_t [y_t(\mathbf{w}^\top \mathbf{x}_t + b) - 1]$$

  a(알파) : Lagrangian multipliers = dual variables, 0 이상의 값을 가진다.

  -> solution 은 saddle point for Lagrangian

  L 은 w,b 에 대해서는 min, a 에 대해서는 max : saddle point

- Step 2 : KKT conditions

  KKT condition 을 이용한다.

  (nonlinear programming 이 optimal 하기 위한 필요 조건이다)

  - w,b 를 a 로 대체하기 위해 사용한다.
  - Step1 의 식을 dual form 으로 변환할 수 있다.

> **Lagrangian duality**
>
> Constrained **minimum** of convex programming problem can also be obtained as **maximization** task applied on Lagrangian.

  KKT conditions

condition 1: $\frac{\partial \mathcal{L}(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)}{\partial \mathbf{w}} = \mathbf{0}$

condition 2: $\frac{\partial \mathcal{L}(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)}{\partial b} = 0$

condition 3: $\alpha_t^* \left[y_t(\mathbf{w}^{*\top}\mathbf{x}_t + b^*) - 1 \right] = 0, \ \forall t$

미분해서 0 Stationarity $\nabla f - \lambda \nabla g = 0$

$\lambda g = 0$

→ 이를 만족하는 w, b가 optimal

$\alpha_t^* = 0$ 이거나 $= 0$ 이다. 즉 boundary에 존재해. → support vector

KKT condition 을 Lagrangian 식에 대입

(condition 1,2)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{t=1}^{N} \alpha_t y_t = 0 \quad \Rightarrow \quad \sum_{t=1}^{N} \alpha_t y_t = 0$$

(insight 1 : w 를 a 의 식으로 표현 가능, w 는 x 와 y 의 weighted sum)

(insight 2 : ay 의 비율 합은 0)

(condition3)

→ 이러한 조건이 N개

condition3 에서, 둘 중 하나만 0 이다. (둘 다 0 이 될 수 없다)

$$\alpha_t > 0 \implies y_t(\mathbf{w}^\top \mathbf{x}_t + b) = 1$$

$$y_t(\mathbf{w}^\top \mathbf{x}_t + b) > 1 \implies \alpha_t = 0$$

→ $\alpha_t > 0$. 즉 S.V 이면 active constraint

(만일, a 가 0 이 아니면, active(=) constraint $y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq 1, \ \forall t \in [1, N]$ 을 의미,

즉 constraint 에서 =등호가 성립한다는 것을 의미. 경계에 존재)

-> Lagrangian multipliers a 는 0 또는 양수

$$\mathbf{w} = \sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t = \sum_{t=1}^{N_s} \alpha_t y_t \mathbf{x}_t$$

N_s : 서포트 벡터 수. $N_s \ll N$

weight 정할때 S.V만 쓰면 된다!

유한한 적은 수의 (x_t, y_t, α_t)로 w를 구할 수 있다.

-> dual 을 사용하니 s.v.의 intuition 이 살아난다.

-> primal 에서는 w 가 d 차원이므로 d 가 무한대이면 문제를 풀 수 없었다.

- Step 3 : formulate dual problem

기존의 primal form 의 Lagrangian :

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha_t [y_t(\mathbf{w}^\top \mathbf{x}_t + b) - 1]$$

식을 조금 변형하면

$$\mathcal{L} = \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha_t y_t \mathbf{w}^\top \mathbf{x}_t - b\sum_{t=1}^{N} \alpha_t y_t + \sum_{t=1}^{N} \alpha_t$$

primal form 을 위의 KKT condition 에서 얻은 3 가지 condition 을 이용해 변형한다.

$$\begin{aligned}
\mathcal{L} &= \frac{1}{2}||\mathbf{w}||^2 - \sum_{t=1}^{N} \alpha_t y_t \mathbf{w}^\top \mathbf{x}_t + \sum_{t=1}^{N} \alpha_t \\
&= \frac{1}{2}\left|\left|\sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t\right|\right|^2 - \sum_{t=1}^{N} \alpha_t y_t \left(\sum_{s=1}^{N} \alpha_s y_s \mathbf{x}_s\right)^\top \mathbf{x}_t + \sum_{t=1}^{N} \alpha_t \\
&= -\frac{1}{2}\sum_{s=1}^{N}\sum_{t=1}^{N} \alpha_s \alpha_t y_s y_t \mathbf{x}_s^\top \mathbf{x}_t + \sum_{t=1}^{N} \alpha_t
\end{aligned}$$

SVM (dual) optimization problem

이제 max 하는 문제로 !!!

내적.

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \sum_{t=1}^{N} \alpha_t - \frac{1}{2}\sum_{s=1}^{N}\sum_{t=1}^{N} \alpha_s \alpha_t y_s y_t \mathbf{x}_s^\top \mathbf{x}_t = \mathcal{L}$$

$$\text{subject to} \quad \sum_{t=0}^{N} \alpha_t y_t = 0$$

← 나머지는 문제에 이미 포함되어 있다.

$$\mathbf{w} = \sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t$$

$$\alpha_t \geq 0, \ \forall t \in [1, N]$$

dual variable 에 대한 조건이 추가됨. $\alpha_t \geq 0$. $\forall t \in [1, N]$

SVM (primal) optimization problem

$$\underset{\mathbf{w}}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$\text{subject to} \quad y_t(\mathbf{w}^\top \mathbf{x}_t + b) \geq 1, \ \forall t \in [1, N]$$

(to compare)

primal 이 w 에 대한 식이었다면 dual 은 a 에 대한 식

(첫 번째 식에서, XTX 를 제외하면 d 에 대한 식이 아닌, N 에 대한 식으로,

d 가 무한으로 가더라도 풀 수 있다.

뒤에서 XTX 가 d 차원과 관련이 있지만, 내적의 형태이므로 kernel trick 이용 가능)

이제, QP solver 를 이용해 문제를 풀 수 있다.

$$\underset{\mathbf{z}}{\text{minimize}} \quad \frac{1}{2}\mathbf{z}^\top Q \mathbf{z} + \mathbf{c}^\top \mathbf{z}$$

$$\text{subject to} \quad A\mathbf{z} \geq \mathbf{a}$$

(d 에 scale 하지 않고 N 에 scale. but 여전히 large N 에 대한 문제가 있다)

dual 의 중요한 성질

1) 오직 terms of training data

2) function depends only on input ad inner products XTX -> can use kernel trick

따라서, nonlinearly separable case 도 고려할 수 있다.

3) SVM 에서, primal and dual is equivalent

dual 에서 a 는 w 로 directly 변환될 수 있다. $\mathbf{w} = \sum_{t=1}^{N} \alpha_t y_t \mathbf{x}_t$

- Step 4 : solve dual problem as QP

- QP solver ~> 등호조건이 1개 추가된 QP solver를 이용

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{c}^\top \mathbf{x} \quad (\alpha\text{가 } x) \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & E\mathbf{x} = \mathbf{d} \end{aligned}$$

- Dual form

maximize 내가 -1배를 통해 minimize 문제로!

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{t=1}^{N} \alpha_t - \frac{1}{2}\sum_{s=1}^{N}\sum_{t=1}^{N} \alpha_s \alpha_t y_s y_t \langle \mathbf{x}_s, \mathbf{x}_t \rangle \\ \text{subject to} \quad & \alpha_t \ge 0, \ \forall t \in [1, N] \\ & \sum_{t=1}^{N} \alpha_t y_t = 0 \end{aligned}$$

$x = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} = \alpha$ 라 하면

$\therefore c = -\mathbb{1}$

$\therefore Q = \left[+y_i y_j \langle x_i, x_j \rangle \right]$ ($x_i^T x_j$ 와 같음)

$\therefore A = -I_N$, $b = 0_N$.

$\therefore E = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}^T = [y_1 .. y_N]$, $d = 0$

이때 E는 vector, d는 scalar

- Dual form 을 QP 에 적용하면

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \mathbf{1}^\top \boldsymbol{\alpha} \\ \text{subject to} \quad & -\boldsymbol{\alpha} \le \mathbf{0} \\ & \mathbf{y}^\top \boldsymbol{\alpha} = \mathbf{0} \end{aligned}$$

$$Q = \begin{bmatrix} y_1 y_1 \mathbf{x}_1^\top \mathbf{x}_1 & y_1 y_2 \mathbf{x}_1^\top \mathbf{x}_2 & \cdots & y_1 y_N \mathbf{x}_1^\top \mathbf{x}_N \\ y_2 y_1 \mathbf{x}_2^\top \mathbf{x}_1 & y_2 y_2 \mathbf{x}_2^\top \mathbf{x}_2 & \cdots & y_2 y_N \mathbf{x}_2^\top \mathbf{x}_N \\ \cdots & \cdots & \cdots & \cdots \\ y_N y_1 \mathbf{x}_N^\top \mathbf{x}_1 & y_N y_2 \mathbf{x}_N^\top \mathbf{x}_2 & \cdots & y_N y_N \mathbf{x}_N^\top \mathbf{x}_N \end{bmatrix}$$

- final form

  - final form

$$\underset{\alpha}{\text{minimize}}\ \frac{1}{2}\alpha^\top \underbrace{\begin{bmatrix} y_1y_1\mathbf{x}_1^\top\mathbf{x}_1 & y_1y_2\mathbf{x}_1^\top\mathbf{x}_2 & \cdots & y_1y_N\mathbf{x}_1^\top\mathbf{x}_N \\ y_2y_1\mathbf{x}_2^\top\mathbf{x}_1 & y_2y_2\mathbf{x}_2^\top\mathbf{x}_2 & \cdots & y_2y_N\mathbf{x}_2^\top\mathbf{x}_N \\ \cdots & \cdots & \cdots & \cdots \\ y_Ny_1\mathbf{x}_N^\top\mathbf{x}_1 & y_Ny_2\mathbf{x}_N^\top\mathbf{x}_2 & \cdots & y_Ny_N\mathbf{x}_N^\top\mathbf{x}_N \end{bmatrix}}_{\text{quadratic coefficients } Q}\alpha + \underbrace{(-\mathbf{1}^\top)}_{\text{linear coefficients}}\alpha$$

$$\text{subject to}\ \underbrace{\mathbf{y}^\top\alpha = 0}_{\text{linear constraint}}$$

$$\underbrace{\mathbf{0}}_{\text{lower bounds}} \le \alpha \le \underbrace{\infty}_{\text{upper bounds}}$$

나중에 soft margin SVM 할때 비교하려고.

- practical considerations

  Q 는 (N,N)의 dense 한 matrix, large N 에서는 solution 을 찾기 힘들다.

  worst case 의 경우, O(N3) time, O(N2) space

- QP solver 는 return solution vector alpha

  대부분 (s.v.가 아닌 경우) a=0 이므로 매우 sparse 한 vector alpha

- Step 5 : complete

final solution

a* = QP 에서 나온 solution

w* = a*와 training data 로 계산한 결과

b* = for any s.v. x+ and x-,

$$b^* = -\frac{1}{2}\left(\mathbf{w}^{*\top}\mathbf{x}_+ + \mathbf{w}^{*\top}\mathbf{x}_-\right) \qquad \begin{aligned}\mathbf{w}^{*\top}\mathbf{x}_+ + b &= +1 \\ \mathbf{w}^{*\top}\mathbf{x}_- + b &= -1\end{aligned}$$

SVM classifier g(x) : linearly separable case

$$g(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{N}\alpha_t^* y_t \mathbf{x}_t^\top\mathbf{x} + b^*\right)$$

$\text{sign}(w^{*T}x+b^*)$

$w^* = \sum_{t=1}^{N} \alpha_t^* y_t x_t$

genalization bound

generalization bound of SVM (Vapnik, 1995):

$$\mathbb{E}[E_{out}] \le \frac{\mathbb{E}[\#\text{ of support vectors}]}{N-1}$$ (매우크다)

Eout 매우 좋다.

- Issues on dual problem

  장점 1)

  d 가 아니라 N 에 대해서 scale, x 의 inner product

  -> 따라서 high d (심지어 무한 dimension)도 handle 가능 → non-linears

  장점 2)

  SVM 의 key properties 의미를 충분히 전달한다.

Summary

- support vector machines: max-margin linear discriminant
  - larger margin $\Rightarrow$ lower d_{VC} $\Rightarrow$ better generalization
  - # support vectors $\approx$ effective # parameters $\approx d_{\text{VC}}$
  - typically much fewer support vectors than data points
  - one of the best off-the-shelf classification algorithms

  → 나중에 VC dimension 다 배우고 나서 다시보기.

- derivation of support vector machines
  - formulated in mathematical optimization framework
  - primal/dual forms can be solved by quadratic programming
  - dual form: allows more efficient solution (also more insightful)

Lecture7. Support vector machines (linearly non-separable)

Outline

* kernel 예제) $K(x,x') = (1+x^Tx')^2$, d=2.

$K(x,x') = (1+x_1x_1' + x_2x_2')^2$

$= 1 + x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1' + 2x_2x_2' + 2x_1x_2x_1'x_2'$

$\therefore z = (1, x_1, x_2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2)$

* kernel 예제 2) $K(x,x') = (x^Tx')^2$, step d.

$K(x,x') = (x_1x_1' + \cdots + x_dx_d')^2$

$= \left(\sum_{i=1}^{d} x_ix_i'\right)^2 = \sum_{i=1}^{d}\sum_{j=1}^{d} x_ix_i'x_jx_j'$

$= \sum_{i,j=1}^{d} (x_ix_j)(x_i'x_j')$

ft) 만일 d=2이면 $z = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_2x_1 \\ x_2x_2 \end{bmatrix}$

$z^Tz' = x_1x_1x_1'x_1' + x_1x_2x_1'x_2' + x_2x_1x_2'x_1' + x_2x_2x_2'x_2'$

Linearly non-separable case1) kernel trick 으로 해결

- Non-linear transform

  feature space Z, feature vector z

  <z,z`>을 <x,x`>으로 나타낼 수 있다. Z 의 차원이 무한이어도 계산 가능하다.

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{t=1}^{N} \alpha_t - \frac{1}{2}\sum_{s=1}^{N}\sum_{t=1}^{N} \alpha_s\alpha_t y_s y_t \mathbf{x}_s^\top \mathbf{x}_t$$

$$\downarrow$$

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{t=1}^{N} \alpha_t - \frac{1}{2}\sum_{s=1}^{N}\sum_{t=1}^{N} \alpha_s\alpha_t y_s y_t \mathbf{z}_s^\top \mathbf{z}_t$$

$z_s^Tz_t = K(x_s, x_t)$

  1) 문제를 푸는 cost 는 x,z 에서 동일하다.

  2) generalization bound 는 d 가 아니라 #s.v.이므로 d 가 커져도 변하지 않는다.

- Kernel trick (or kernel substitution)

  -> inner product 만을 이용하는 어떤 algorithm 에서도 적용 가능하다.

  Kernel : inner product in feature space, zTz`=K(x,x`),

  Kernel trick : feature space 로 가지 않고 input space 에서 kernel 을 계산할 수 있다.

  K(x,x`) : x,x`또는 z,z`이 얼마나 similar 한가, that is nonlinear similarity measure

- polynomials (of degree $q > 0$):

$$K(\mathbf{x}, \mathbf{x}') = (a\mathbf{x}^\top \mathbf{x}' + b)^q$$

- radial-basis functions ($\gamma > 0$):

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$

- Gaussian RBF:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

- hyperbolic tangent (aka sigmoid, multilayer perceptron kernel):

$$K(\mathbf{x}, \mathbf{x}') = \tanh(\beta \mathbf{x}^\top \mathbf{x}' + \gamma)$$

- multiquadric:

$$K(\mathbf{x}, \mathbf{x}') = \sqrt{||\mathbf{x} - \mathbf{x}'||^2 + c^2}$$

- wave:

$$K(\mathbf{x}, \mathbf{x}') = \frac{\theta}{||\mathbf{x} - \mathbf{x}'||} \sin \frac{||\mathbf{x} - \mathbf{x}'||}{\theta}$$

- equivalent kernel : polynomial 에서 a=b=0
- linear kernel : polynomial 에서 q=1
- RBF kernel 은 infinite feature dimension, r 이 너무 크면 over fitting 일어난다.

- Valid kernel 인가? : Mercer's theorem

Let function $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ be given. For K to be a valid (Mercer) kernel, it is necessary and sufficient that, for any finite set of instances, the corresponding *kernel matrix* is

: 1) symmetric 2) positive semi-definite

$K_{ij} = K(x_i, x_j)$ 가 성질을 만족.
예) 어떤 finite set $\{x_1 \dots x_m\}$ 에 대해서도 성립해야 해.

kernel matrix

m finite points {x1, ..., xm}

K = (m,m) matrix, Kij = K(xi,xj)

positive semi-definite : PSD

≥ 0 ≥ 0

all eigen values, determinant is non-negative

real symmetric K 가 tTKt 는 0 이상 for all vector t

$t^T K t \geq 0$ for $\forall t$

Linearly non-separable case2) soft-margin SVM

- Reformulating SVM problem : primal form

primal form (C: positive constant, C 가 클수록 hard margin 에 가까워)

원래 primal form 에서 $C \geq \xi_t$ 더해.
또한 조건식 변형, 추가

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{t=1}^{N}\xi_t \\ \text{subject to} \quad & y_t(\mathbf{w}^T\mathbf{x}_t + b) \geq 1 - \xi_t, \quad \forall t \\ & \xi_t \geq 0, \quad \forall t \end{aligned}$$

primal variable: $(\mathbf{w}, b, \xi_t)$

- Reformulating SVM problem : dual form 으로 변환하는 과정
  - Lagrangian $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$

$$= \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{t=1}^{N}\xi_t - \sum_{t=1}^{N}\alpha_t(y_t(\mathbf{w}^\top\mathbf{x}_t + b) - 1 + \xi_t) - \sum_{t=1}^{N}\beta_t\xi_t$$

    - minimize wrt $\mathbf{w}$, b, and $\boldsymbol{\xi}$
    - maximize wrt each $\alpha_t \geq 0$ and $\beta_t \geq 0$

  (primal variable 에 대해서는 minimum, dual variable 에 대해서는 maximum)

  - KKT conditions are obtained as follows:

$$\nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum_{t=1}^{N}\alpha_t y_t \mathbf{x}_t = \mathbf{0} \quad \Rightarrow \quad \mathbf{w} = \sum_{t=1}^{N}\alpha_t y_t \mathbf{x}_t$$

$$\frac{\partial\mathcal{L}}{\partial b} = -\sum_{t=1}^{N}\alpha_t y_t = 0 \quad \Rightarrow \quad \sum_{t=1}^{N}\alpha_t y_t = 0$$

$$\frac{\partial\mathcal{L}}{\partial \xi_t} = C - \alpha_t - \beta_t = 0 \quad \Rightarrow \quad \alpha_t + \beta_t = C$$

$$\alpha_t[y_t(\mathbf{w}^\top\mathbf{x}_t + b) - 1 + \xi_t] = 0 \quad \Rightarrow \quad \alpha_t = 0 \vee y_t(\mathbf{w}^\top\mathbf{x}_t + b) = 1 - \xi_t$$

$$\beta_t\xi_t = 0 \quad \Rightarrow \quad \beta_t = 0 \vee \xi_t = 0$$

Final form, 아래의 식을 QP 에 적용

$$\begin{aligned} \text{maximize} \quad & \sum_{t=1}^{N}\alpha_t - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top\mathbf{x}_j \\ \text{subject to} \quad & \sum_{t=1}^{N}\alpha_t y_t = 0 \\ & 0 \leq \alpha_t \leq C \quad \forall t \end{aligned}$$

only change

final solution

Soft-margin SVM classifier

$$g(\mathbf{x}) = \text{sign}\left(\mathbf{w}^\top\mathbf{x} + b\right)$$

where

$$\mathbf{w} = \sum_{0<\alpha_t\leq C}\alpha_t y_t \mathbf{x}_t$$

$$b = y_{\text{sv}} - \sum_{0<\alpha_t\leq C}\alpha_t y_t \mathbf{x}_t^\top\mathbf{x}_{\text{sv}}$$

- solution a 를 이용해 optimal w 를 구한다. 이후 from any s.v.에서 b 를 구한다.
- ξ_t 는 식에 나타나지 않지만 C 의 형태로 존재한다.
- ξ_t >0 인 vectors (linearly separable 에서는 없었던) :

outliers.
(margin 내에 있거나, 잘못 분류된)

w = sum (ayx)에서 a=C 이므로 w 에서 비중이 크다.

- 모두 정리

	모범생	원래 S.V	군사분계선	간첩
$y_t(\mathbf{w}^\top \mathbf{x}_t + b)$	y(wTx+b)>1	y(wTx+b)=1	0≤y(wTx+b)<1	y(wTx+b)<0
ξ_t	E=0	E=0	0<E≤1	1<E
a (alpha)	a=0	0<a≤C	a=C	a=C
b (beta)	b≠0	b≠0	b=0	b=0
Vector x lies	Out of margin	On margin	In margin	The other area
분류	right	right	right	wrong

여기서 outlier의 a값이 C인것 어떻게 알수있어?

Summary

- SVM advantages:
  - systematic implementation through quadratic programming ⇒ very efficient implementations exist (매우 잘 개발된 QP solver 사용可.)
  - excellent data-dependent generalization bounds exist ???
  - regularization built into cost function
  - statistical performance independent of dim of feature space (d에 independent.)
- drawbacks:
  - treatment of non-separable case somewhat heuristic → 왜?
  - number of support vectors may depend strongly on
    - kernel type and hyper-parameters ※S.V는 어떤 커널, 어떤 hyper 파라미터에 따라 달라진다.
  - systematic choice of kernels is difficult (need prior information)
  - optimization may require clever heuristics for large problems