CS229: Machine Learning

Junchuan Zhao

May 24, 2022

Abstract

This document includes my notes of CS229: Machine Learning.

1 Lecture 2

1.1 Linear Regression

Hypothesis: $h(x) = \sum\limits_{j=0}^{n} \theta_j x_j$ (n: number of features)
Linear Regression: the hypothesis is the linear combination of the training dataset.
Loss function (goal): $\min\limits_{\theta} \frac{1}{2} \sum\limits_{i=1}^{m} (h_\theta(x^i) - y^i)^2$
Parameters: θ, m, n, x, y

1.2 Gradient Descent

1.2.1 Batch Gradient Descent

Basic Ideas: start with some θ, keep changing θ to reduce $J(\theta)$, repeat until convergent
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (α: learning rate)
$\frac{\partial}{\partial \theta_j} J(\theta) = (h_\theta(x) - y) \cdot x_j$
$\theta_j := \theta_j - \alpha \sum\limits_{i=1}^{m} (h_\theta(x)^i - y^i) \cdot x_j^i$ (α: learning rate)
Batch Gradient Descent: all the training data as a batch when optimizing the loss function. (-): not good for large scale dataset, very expensive.

1.2.2 Stochastic Gradient Descent

Algorithm 1 Stochastic Gradient Descent

```
Require: the parameter of $j^{th}$ feature
Ensure: the updated parameter of $j^{th}$ feature
1: for $i = 1$ to $m$ do
2:     $\theta_j := \theta_j - \alpha \cdot (h_\theta(x)^i - y^i) \cdot x_j^i$;
3: end for
```

stochastic gradient descent heads over to the global optimum.

1.2.3 Total Equations

A Total Equation for Stochastic Gradient Descent: $\nabla_\theta J(\theta) = \mathbf{0}$
If A is square ($A \in \mathbb{R}_{n \times n}$)
Trace of A: $tr(A) = \sum\limits_i A_{ii}$
Features of Trace:

- $tr(A) = tr(A^T)$
- $f(A) = tr(AB)$, $\nabla_A f(A) = B^T$
- $tr(AB) = tr(BA)$
- $tr(ABC) = tr(CAB)$
- $\nabla_A tr(AA^TC) = CA + C^TA$

Loss Function: $\nabla_\theta J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
$= \frac{1}{2}(\theta^TX^T - y^T)(X\theta - y)$
$= \frac{1}{2}(\theta^TX^TX\theta - \theta^TX^Ty - y^TX\theta)$
$= X^TX\theta - X^Ty$
Loss Function: $X^TX\theta - X^Ty = \mathbf{0} \iff X^TX\theta = X^Ty$
$\theta = (X^TX)^{-1}X^Ty$

2 Lecture 3

2.1 Locally Weighted Regression

"Parametric" learning algorithm: fit fixed set of parameters (θ_i) to data.
"Non-parametric learning algorithm": amount of data/parameters you need to keep grows (linearly) with the size of data.

Linear Regression: to evaluate h at certain x
Fit θ to minimize
$\frac{1}{2}\sum\limits_i(y^i - \theta^Tx^i)^2$,
return θ^Tx

Linear Regression: to evaluate h at local region of x
Fit θ to minimize
$\sum\limits_i w^i(y^i - \theta^Tx^i)^2$, where w^i is a "weight function".
common choice for w^i is $w^i = e^{(-\frac{(x^i - x)^2}{2\tau^2})}$
If $|x^i - x|$ is small, then $w^i \approx 1$.
If $|x^i - x|$ is large, then $w^i \approx 0$.
τ: bandwidth, control a larger or narrower window.

2.2 Why Square Error?

Assume $y^i = \theta^Tx^i + \epsilon^i$, where $\epsilon^i \sim N(0, \sigma^2)$ models effects of random noise
$p(y^i|x^i;\theta) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(y^i - \theta^Tx^i)^2}{2\sigma^2}}$
$\iff y^i|x^i;\theta \sim N(\theta^Tx^i, \sigma^2)$

Difference between Likelihood (L) and Probability (P): L variess parameters, P varies datapoints. Assume the datapoints are IID,
The "likelihood" of θ: $L(\theta) = p(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{i=1}^{m} p(y^i|x^i;\theta)$
$l(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{(\dots)} = \sum_{i=1}^{m} [\log \frac{1}{\sqrt{2\pi}\sigma} + \log e^{(\dots)}]$
$= m \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_{i=1}^{m} \frac{(y^i - \theta^T x^i)^2}{2\sigma^2}$

MLE: maximum likelihood estimation. Choose θ to maximize $L(\theta)$
i.e. choose θ to minimize $\frac{1}{2} \sum_{i=1}^{m} (y^i - \theta^T x^i)^2$, which is actually $J(\theta)$

2.3 Classification

Binary Classification: $y \in \{0, 1\}$

2.3.1 Logistic Regression

We want $h_\theta(x) \in [0, 1]$, so we define
$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$
Sigmoid Function: $g(z) = \frac{1}{1+e^{-z}}$
An important thing is that: the partial derivative of the sigmoid function is a **concave** function, which means it has the global maximum.

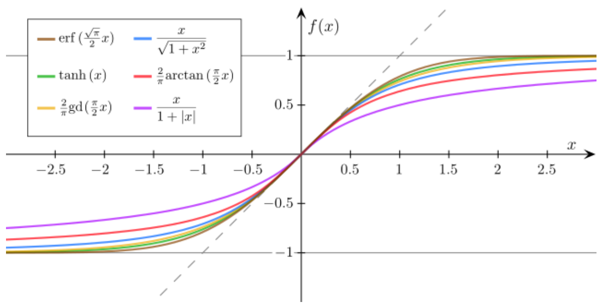


Figure 1: The graph of sigmoid function.

Different from Linear Regression $h_\theta(x) = \theta^T x$, Logistic Regression is $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.
$p(y = 1|x;\theta) = h_\theta(x)$, $p(y = 0|x;\theta) = 1 - h_\theta(x)$
In sum, $p(y|x;\theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$

$L(\theta) = p(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{i=1}^{m} h_\theta(x^i)^{y^i} (1 - h_\theta(x^i))^{1-y^i}$
$l(\theta) = \log(L(\theta)) = \sum_{i=1}^{m} y^i \log h_\theta(x^i) + (1 - y^i) \log(1 - h_\theta(x^i))$
Choose θ to maximize $l(\theta)$, we use Batch gradient ascent.

Batch Gradient Ascent:
$\theta_j := \theta_j + \alpha \frac{\partial}{\partial \theta_j} l(\theta)$
$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$, same as the one in linear regression. (A larger category called generalized linear model (GLM))
Note: no **normal equation** for logistic regression solution.

2.3.2 Newton's Method

The basic idea: have some function f, we want to fit θ, **s.t.** $f(\theta) = 0 \iff$ want maximizing $l(\theta)$ $\iff$ want $l'(\theta) = 0$

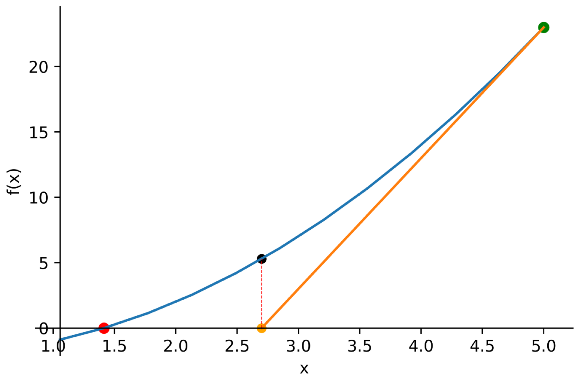


Figure 2: Newton's Method.

$\theta^{t+1} := \theta^t - \frac{f(\theta^t)}{f^{'}(\theta^t)}$, let $f(\theta) = l^{'}(\theta)$
$\theta^{t+1} := \theta^t - \frac{l^{'}(\theta^t)}{l^{''}(\theta^t)}$

"**Quadratic convergent**": 0.01 error $\longrightarrow$ 0.0001 error $\longrightarrow$ 0.00000001 error.

When θ is a vector: $(\theta \in \mathbb{R}^{n+1})$
$\theta^{t+1} := \theta^t + \alpha H^{-1} \nabla_\theta l(\theta)$
where

$$\nabla_\theta l(\theta) = \begin{bmatrix} \frac{\partial l}{\partial \theta_{00}} & \frac{\partial l}{\partial \theta_{01}} & \cdots & \frac{\partial l}{\partial \theta_{0n}} \\ \frac{\partial l}{\partial \theta_{10}} & \frac{\partial l}{\partial \theta_{11}} & \cdots & \frac{\partial l}{\partial \theta_{1n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial l}{\partial \theta_{n0}} & \frac{\partial l}{\partial \theta_{n1}} & \cdots & \frac{\partial l}{\partial \theta_{nn}} \end{bmatrix}$$

$H \in \mathbb{R}^{n+1 \times n+1}$ is the Hessian Matrix, $H_{ij} = \frac{\partial^2 l}{\partial \theta_i \partial \theta_j}$
Note: If the dataset is large, the size of H will be too large to compute.

3 Lecture 4

3.1 Perceptron

Binary step function:

$$g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{1}$$

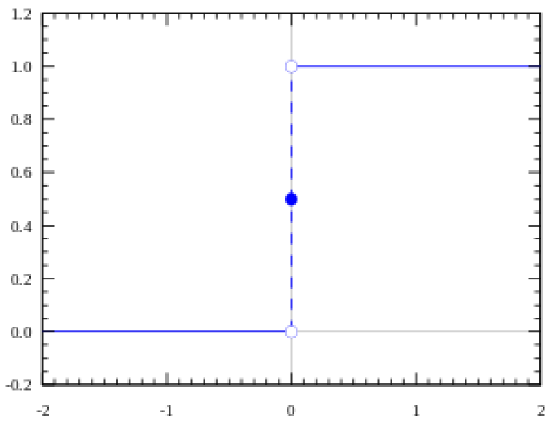


Figure 3: The graph of binary step.

Batch Gradient Ascent: $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$

3.2 Exponential Family

3.2.1 Defination and Examples of Exponential Family

Probability Density Function (PDF): $p(y;\eta) = b(y)\exp[\eta^T T(y) - a(\eta)]$
y: data
η: natural parameter
$T(y)$: sufficient statistic
$b(y)$: base measure
$a(\eta)$: log partition

Example 1: Bernoulli (Binary Data): $p(y;\phi) = \phi^y(1-\phi)^{(1-y)}$
ϕ = probability of the event
$p(y;\phi) = \exp[y\log(\frac{\phi}{1-\phi}) + \log(1-\phi)]$
$b(y) = 1$
$T(y) = y$
$\eta = \log\frac{\phi}{1-\phi} \Rightarrow \phi = \frac{1}{1+e^{-\eta}}$
$a(\eta) = -\log(1-\phi) = -\log(1-\frac{1}{1+e^{-\eta}}) = \log(1+e^\eta)$

Example 2: Gaussian (assume variance = 1): $p(y;\mu) = \frac{1}{\sqrt{2\pi}}e^{(-\frac{(y-\mu)^2}{2})} = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}\exp(\mu y - \frac{\mu^2}{2})$
$b(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$
$T(y) = y$
$\eta = \mu$
$a(\eta) = \frac{\mu^2}{2} = \frac{\eta^2}{2}$

3.2.2 Properties of Exponential Family

- MLE w.r.t η is concave, NLL is convex
- $E[y;\eta] = \frac{\partial}{\partial\eta}a(\eta)$
- $Var[y;\eta] = \frac{\partial^2}{\partial\eta^2}a(\eta)$

Real - Gaussian; Binary - Bernoulli; Count - Poisson; $\mathbb{R}^+$ - Gamma, Exponential; Distribution - Beta, Dirichlet (Bayesian).

3.3 Generalized Linear Model (GLM)

3.3.1 GLM Assumptions

Assumptions/Design Choices
(1) $y|x;\theta \sim$ Exponential Family (η)
(2) $\eta = \theta^T x$, $\theta \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$
(3) Test Time: output $h_\theta(x) = E[y|x;\theta]$

3.3.2 GLM Pipeline

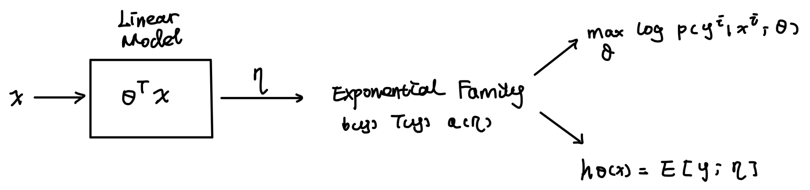


Figure 4: The pipeline for GLM.

3.3.3 GLM Training

Learning Update Rule: $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$

Terminology:
η - natural parameter
$g(\eta) = \mu = E(y;\eta)$ - canonical response function
$\eta = g^{-1}(\mu)$ - canonical link function

Three parameterizations:
Model Param: θ (**Learn Parameters**)
Natural Param: η (Design Choice: $\theta^T x$)
Canonical Param: ϕ - Bernoulli, μ σ - Gaussian, λ - Poisson (Canonical Response: $g(\cdot)$)

3.3.4 Review Logistic Regression

$h_\theta(x) = E[y|x;\theta] = \phi$ (the mean of the Bernoulli distribution)
$h_\theta(x) = \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$

3.4 Softmax Regression

Task: multi-class Classification
k - number of classes, $x^i \in \mathbb{R}^n$, Label $y^i \in \{0,1\}^k$ (one-hot vector)
Each class has its own parameters: $\theta_{class} \in \mathbb{R}^n$
$p(y=i|x;\theta) = \frac{e^{\theta_i^T x}}{\sum_{j=1}^k e^{\theta_j^T x}}$
$l(\theta) = \sum_{i=1}^m log \prod_{l=1}^k (\frac{e^{\theta_l^T x^i}}{\sum_{j=1}^k e^{\theta_j^T x^i}})^{\{y^i=l\}}$

4 Lecture 5

4.1 Discriminative v.s. Generative Learning Algorithms

Discriminative: Learn $p(y|x)$ which is the mapping from x to y.

Generative: Learn $p(x|y)$, $p(y)$ (class prior)
According to Bayes rule: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$, $p(x) = \sum_{i=0}^{k} p(x|y=i)p(y=i)$

4.2 Gaussian Discriminant Analysis (GDA)

4.2.1 Assumptions and Fundamental Knowledge

Suppose $x \in \mathbb{R}^n$ (convention: drop $x_0 = 1$)
Assume $p(x|y)$ is Gaussian

Basic Knowledge for Multivariate Gaussian:
$z \sim N(\boldsymbol{\mu}, \Sigma)$, $(z \in \mathbb{R}^n)$
$p(z) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\boldsymbol{\mu})^T\Sigma^{-1}(x-\boldsymbol{\mu}))$ $\qquad E[z] = \boldsymbol{\mu}$
$Cov(z) = E[(z-\mu)(z-\mu)^T] = E_{zz^T} - E[z]E[z]^T$

4.2.2 GDA model

Parameters: $\boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma$ (use same covariance matrix)
$p(x|y=0) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\boldsymbol{\mu_0})^T\Sigma^{-1}(x-\boldsymbol{\mu_0}))$
$p(x|y=1) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\boldsymbol{\mu_1})^T\Sigma^{-1}(x-\boldsymbol{\mu_1}))$
$p(y) = \phi^y(1-\phi)^y$ $(p(y=1) = \phi)$

Training set: $\{(x^i, y^i)\}_{i=1}^m$
Joint Likelihood: $L(\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma) = \prod_{i=1}^{m} p(x^i, y^i; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}) = \prod_{i=1}^{m} p(x^i|y^i)p(y^i)$

Discriminative Learning (**Conditional Likelihood**): $\prod_{i=1}^{m} p(y^i|x^i;\theta)$

Maximize Likelihood Estimation (MLE):
$\max_{\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma} l(\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma)$
Results of MLE:
$\phi = \frac{\sum_{i=1}^{m} y^i}{m} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^i=1\}}{m}$ $(\mathbb{1} : \mathbb{1}\{\text{true}\} = 1, \mathbb{1}\{\text{false}\} = 0)$
$\boldsymbol{\mu_0} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\}x^i}{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\}} = \frac{\text{sum of feature vectors for samples with y=0}}{\text{number of samples with y=0}}$
$\boldsymbol{\mu_0} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^i=1\}x^i}{\sum_{i=1}^{m} \mathbb{1}\{y^i=1\}}$
$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^i - \mu_{y^i})(x^i - \mu_{y^i})^T$

Prediction:
$\arg\max_y p(y|x) = \arg\max_y \frac{p(x|y)p(y)}{p(x)}$ ($p(x)$ is constant)
$= \arg\max p(x|y)p(y)$

4.2.3 Compare GDA to Logistic Regression

For fix parameters: $\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma$, plot $p(y|x; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma)$ as a function of x.
By Bayes Rule: $p(y=1|x; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma) = \frac{p(x|y=1; \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma) p(y=1; \phi)}{p(x; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma)}$.
In the end, $p(y=1|x; \phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma) = \frac{1}{1+exp(-\theta^T x)}$, where θ is a function of $\phi, \boldsymbol{\mu_0}, \boldsymbol{\mu_1}, \Sigma$.

The assumption of GDA $\Rightarrow p(y=i|x) = \frac{1}{1+e^{-\theta^T x}}$
However, $p(y=i|x) = \frac{1}{1+e^{-\theta^T x}} \not\Rightarrow$ GDA assumptions.

GDA	Logistic Regression
stronger assumptions	weaker assumptions
perform better when assumptions are correct	fine with all the scenarios (distributions)
small dataset	large dataset
computationally efficient	fits the era of the big data

Table 1: The comparison between GDA and Logistic Regression.

4.3 Naive Bayes

For a spam email classification problem:
1. create a word dictionary (size m).
If represent x by a binary feature vector, $x \in \{0,1\}^n$. ($x_i = \mathbb{1}\{\text{word i appears in email}\}$)
This cause 2^n possible values of x, which is not possible for large dictionary.
2. Assume x_i**'s are conditionally independent given y**, which means

$$p(x_1, ..., x_{10000}|y) = p(x_1|y)p(x_2|x_1, y)p(x_3|x_1, x_2, y)...p(x_{10000}|..., y)$$
$$\overset{\text{assume}}{=\!=\!=} p(x_1|y)p(x_2|y)...p(x_{10000}|y) = \prod_{i=1}^{n} p(x_i|y)$$

Parameters:
$\phi_{j|y=1} = p(x_j = 1|y=1)$
$\phi_{j|y=0} = p(x_j = 1|y=0)$
$\phi_y = p(y=1)$

Joint Likelihood: $L(\phi_i, \phi_{j|y}) = \prod_{i=1}^{m} p(x^i, y^i; \phi_j, \phi_{j|y})$

MLE results: $\phi_y = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^i=1\}}{m}$

$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} \mathbb{1}\{x_j^i=1, y^i=1\}}{\sum_{i=1}^{m} \mathbb{1}\{y^i=1\}}$

$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} \mathbb{1}\{x_j^i=1, y^i=0\}}{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\}}$

At prediction time:
$p(y=1|x) = \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)}$
Problem: when a word does not exist – $\frac{0}{0}$

5 Lecture 6

5.1 Laplace Smoothing

$x \in \{1, ..., k\}$, estimate $p(x = j) = \frac{\sum_{j=1}^{m} \mathbb{1}\{x^i=j\}+1}{m++k}$

$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} \mathbb{1}\{x_j^i=1, y^i=0\}+1}{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\}+2}$

5.2 Multinomial Event Model

$x \in \begin{bmatrix} 600 \\ 800 \\ 1600 \\ 6200 \end{bmatrix}$, $x \in \mathbb{R}^n$

$x_j \in \{1, 2, ..., 10000\}$, $n =$ length of email, $m =$ number of emails

Parameters: $\phi_y = p(y = 1), \phi_{k|y=0} = p(x_j = k|y = 0), \phi_{k|y=1} = p(x_j = k|y = 1)$

MLE results: $\phi_{k|y=0} = \frac{\sum_{i=1}^{m} (\mathbb{1}\{y^i=0\} \sum_{j=1}^{n_i} \mathbb{1}\{x_j^i=k\})}{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\} n_i}$

$\underline{\underline{\text{laplace smoothing}}} = \frac{\sum_{i=1}^{m} (\mathbb{1}\{y^i=0\} \sum_{j=1}^{n_i} \mathbb{1}\{x_j^i=k\})+1}{\sum_{i=1}^{m} \mathbb{1}\{y^i=0\} n_i + 10000}$

The advantages for GDA and Naive Bayes: quick to train, non-iterative.

5.3 Support Vector Machines SVM

SVM helps to find non-linear boundaries for classification problems.

5.3.1 Optimal Margin Classifier

Functional Margin: how confidentaly and accurately to classify an example.

Logistic Regression: $h_\theta(x) = g(\theta^T x)$, predict 1 if $\theta^T x \geq 0$ and 0 otherwise.

We want if $y^i = 1$, hope that $\theta^T x^i \gg 0$ and 0 when $\theta^T x^i \ll 0$. – correct or confident prediction.

Notation: Labels $y \in \{-1, +1\}$,

$$g(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases} \tag{2}$$

$h_{W,b}(x) = g(W^T x + b)$, $W \in \mathbb{R}^n$ and $b \in \mathbb{R}$. (Compare to Logistic Regression, b is θ_0)

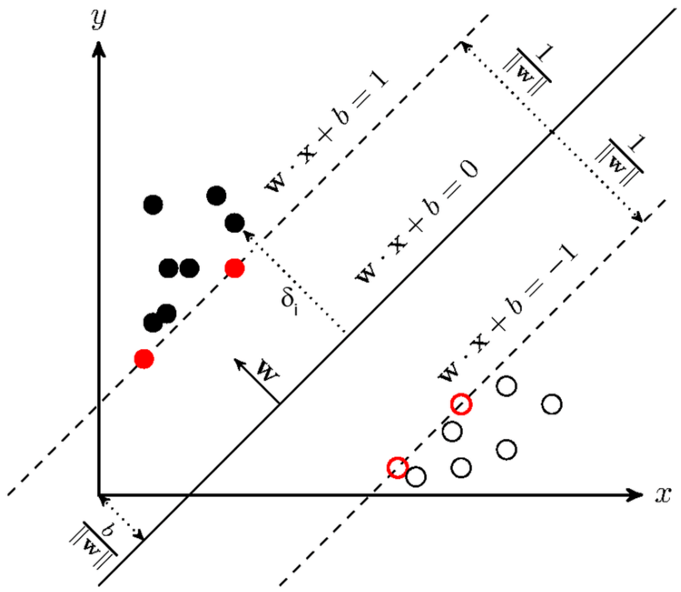
Functional Margin of hyperplane defined by (W, b) w.r.t (x^i, y^i):

$\hat{\gamma}^i = y^i(W^T x^i + b)$, we want $\gamma^i \gg 0$

Funtional Margine w.r.t training set: $\hat{\gamma} = \min_i \hat{\gamma}^i$, $i = 1, ..., m$

Rescaling the parameters: $(W, b) \rightarrow (\frac{W}{\|W\|}, \frac{b}{\|W\|})$

Geometric Margin: the distance from the data point to the hyperplane.



y
x
$\mathbf{w}\cdot\mathbf{x}+b=1$
$\mathbf{w}\cdot\mathbf{x}+b=0$
$\mathbf{w}\cdot\mathbf{x}+b=-1$
$\frac{1}{\|\mathbf{w}\|}$
$\frac{1}{\|\mathbf{w}\|}$
δ_i
$\mathbf{w}$
$\frac{b}{\|\mathbf{w}\|}$

Figure 5: Geometric Margin.

Geometric Margin of hyperplane (W,b) w.r.t (x^i,y^i):
$\gamma^i = \frac{y^i(W^T x^i + b)}{\|W\|}$

Geometric Margin w.r.t training set: $\gamma = \min\limits_i \gamma^i$

The relationship between geometric and functional margin: $\gamma^i = \frac{\hat{\gamma}^i}{\|W\|}$

$\hat{\gamma}$ is functional margin; γ is geometric margin.

Optimal margin classifier: choose w, b to maximize γ:
$\max\limits_{\gamma,w,b} \gamma$ s.t. $\frac{y^i(W^T x^i + b)}{\|W\|} \geq \gamma$, $(i = 1, ..., m.) \Rightarrow \min\limits_{w,b} \|W\|^2$ s.t. $y^i(W^T x^i + b) \geq 1$

6 Lecture 7

6.1 Optimal Margin Classifiers

If $x^i \in \mathbb{R}^{100}$ (highly dimensional spaces), suppose $W = \sum\limits_{i=1}^{m} \alpha_i y^i x^i$ (W can be represented by the linear combination of the training set)

Intuition 1: Logistic Regression $\theta := 0$, Gradient Descent: $\theta := \theta - \alpha(h_\theta(x^i) - y^i)x^i$
Batch Gradient Descent: $\theta := \theta - \alpha \sum\limits_{i=1}^{m} (h_\theta(x^i) - y^i)x^i$

Intuition 2: W pins the decision boundary, W lies in the span of the training examples

$\min\limits_{W,b} \frac{1}{2}\|W\|^2$ s.t. $y^i(W^T x^i + b) \geq 1$, $W = \sum\limits_{i=1}^{m} \alpha_i y^i x^i$

$\min\limits_{W,b} \frac{1}{2}(\sum\limits_{i=1}^{m} \alpha_i y^i x^i)^T (\sum\limits_{j=1}^{m} \alpha_j y^j x^j) = \min\limits_{W,b} \frac{1}{2} \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{m} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle$

$y^i(W^T x^i + b) \geq 1 \Leftrightarrow y^i(\sum\limits_j \alpha_j y^j \langle x^j, x^i \rangle + b) \geq 1$

6.1.1 Dual Optimization Problem

$\max_\alpha W(\alpha) = \sum\limits_{i=1}^{m} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{m} y^i y^j \alpha_i \alpha_j \left\langle x^i, x^j \right\rangle$

s.t. $\alpha_i \geq 0$ $(i = 1, ..., m)$, $\sum\limits_{i=1}^{m} \alpha_i y^i = 0$

Make predictions:
(1) solve for α_i's
(2) to make prediction, $h_{w,b}(x) = g(w^T x + b) = g((\sum\limits_i \alpha_i y^i x^i)^T x + b) = g(\sum\limits_i \alpha_i y^i \left\langle x^i, x \right\rangle + b)$

6.2 Kernel

6.2.1 Kernel Tricks

Kernel trick:
(1) write algorithm in terms of $\langle x, z \rangle$
(2) map $x \longrightarrow \phi(x)$ (higher dimension)
(3) find a way to compute $K(x, z) = \phi(x)^T \phi(z)$
(4) replace $\langle x, z \rangle$ in algorithm with $K(x, z)$

Traditional way:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ \dots \\ x_3x_2 \\ x_3x_3 \end{bmatrix} \quad \phi(z) = \begin{bmatrix} z_1z_1 \\ z_1z_2 \\ z_1z_3 \\ \dots \\ z_3z_2 \\ z_3z_3 \end{bmatrix}$$

n^2 elements need $O(n^2)$ time to compute $\phi(x)$ or $\phi(x)^T \phi(z)$

Kernel trick # 1: $(O(n))$
$K(x, z) = \phi(x)^T \phi(z) = (x^T z)^2 = (\sum\limits_{i=1}^{n} x_i z_i)(\sum\limits_{j=1}^{n} x_j z_j)$
$= \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} (x_i x_j)(z_i z_j)$

Kernel trick # 2: $K(x, z) = (x^T z + c)^2$

Kernel trick # 3: $K(x, z) = (x^T z + c)^d$, $\phi(x)$ has all $\begin{pmatrix} n+d \\ d \end{pmatrix} \approx (n+d)^d$ features of monomials up to order d.

SVM = Optimal Margin Classifier + Kernel Tricks (Map the data points to a higher dimension, then seperate by using linear hyperplane)

6.2.2 How to Make Kernels?

Basic idea: if x, z are similar, $K(x, z) = \phi(x)^T \phi(z)$ is large. Vice versa.
Gaussian Kernel: $K(x, z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$

Theorem(Mercer): K is a valid kernel function (i.e. $\exists\ \phi$ s.t. $K(x, z) = \phi(x)^T \phi(z)$) $\iff$ for any d points $(x^1 ... x^d)$, the corresponding kernel matrix $K \geq 0$.

Linear Kernel: $K(x,z) = x^T z$, $\phi(x) = x$
Gaussian Kernel: $K(x,z) = \exp(-\frac{\|x-z\|^2}{2\sigma^2})$, $\phi(x) \in \mathbb{R}^\infty$

Note: a lot of algorithms can marry with the kernel tricks to reduce the amounts of computation.

6.3 L1-norm Soft Margin SVM

$\min\limits_{W,b,\xi_i} \frac{1}{2}\|W\|^2 + C\sum\limits_{i=1}^{m}\xi_i$ s.t. $y^i(W^T x^i + b) \geq 1 - \xi_i$ $(i = 1, ..., m;\ \xi_i \geq 0)$
Note: more robust to outliers.

Dual Optimization Form:
$\max_\alpha W(\alpha) = \sum\limits_{i=1}^{m}\alpha_i - \frac{1}{2}\sum\limits_{i,j=1}^{m} y^i y^j \alpha_i \alpha_j \langle x^i, x^j\rangle$
s.t. $0 \leq \alpha_i \leq C$ $(i = 1, ..., m)$, $\sum\limits_{i=1}^{m}\alpha_i y^i = 0$

6.4 SVM Kernels

$K(x,z) = (x^T z)^d = \exp(-\frac{\|x-z\|^2}{2\delta^2})$

6.4.1 Protean Sequence Classifier

Protean Sequence: $BAJTSJAIBAJT...$
Protean Codebook: AAAA, AAAB, AAAC, ..., ZZZZ ($20^4 = 160000$)

$$\phi(x) = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \\ ... \\ 4 \\ ... \\ 0 \end{bmatrix}$$

$\phi(x)^T\phi(z) = K(x,z)$

7 Lecture 8

7.1 Bias and Variance

Underfitting problem: high bias.
Overfitting problem: high variance.

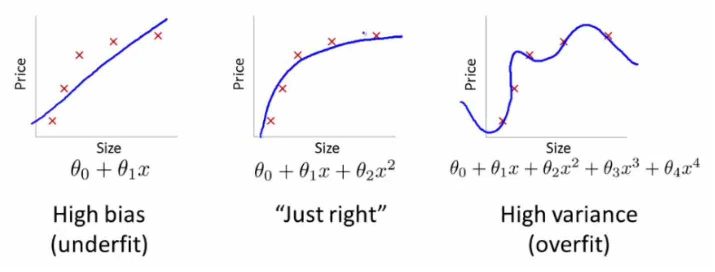


Figure 6: Bias and Variance.

7.2 Regularization

Regularzation for linear regression: $\min_\theta \frac{1}{2}\sum_{i=1}^{m} \|y^i - \theta^T x^i\|^2 + \frac{\lambda}{2}\|\theta\|^2$
When λ is too big – underfitting, whereas λ is too small – overfitting

Regularization for logistic regression: $\arg\max_\theta \sum_{i=1}^{m} \log p(y^i|x^i;\theta) - \lambda\|\theta\|^2$

Support Vector Machine (SVM): $\min_{W,b} \|W\|^2$, since W is $\theta_1...\theta_n$, the loss function already involves the regularization idea.

From another perspective: $S: \ \{(x^i, y^i)\}_{i=1}^m$
$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)}$
$\arg\max_\theta p(\theta|S) = \arg\max_\theta p(S|\theta)p(\theta)$
$= \arg\max_\theta (\prod_{i=1}^{m} p(y^i|x^i,\theta))p(\theta)$
Assume $\theta \ \sim \ N(0, \tau^2 I)$, $p(\theta) = \frac{1}{\tau\sqrt{2\pi I}}\exp(-\frac{\theta^2}{2\tau^2 I})$
After log the MLE, we will obtain $\|\theta\|^2$ term which is the regularization term.

Frequentist: $\arg\max_\theta p(S|\theta)$ – MLE (estimate the parameters)
Bayesian: *theta* is unknown, prior distribution $p(\theta)$, $\arg\max_\theta p(\theta|S)$ – MAP (maximum a posterior)

7.3 Data Split

Train/dev/test sets
Choose the form of polynomial, λ, τ, C

7.3.1 Hold-out Cross Validation (large dataset)

$S \to S_{train}$, S_{dev}, S_{test} (dev: development)
(1) Train each model (options for design of polynomial) on S_{train}, and get some hypothesis h.
(2) Mesaure error on S_{dev}, pick model w.r.t lowest error on S_{dev}.
(3) Measure error on S_{test} and report.

7.3.2 k-fold Cross Validation (small dataset)

$m = 100$, $S = \{(x^i, y^i)\}$ $(i = 1, ..., m)$
$k = 10$ is typical, $k = 5$ for illustration (Divide the dataset into 5 pieces).

Algorithm 2 k-fold CV

```
1: for $d = 1$ to 5 do
2:     for $i = 1$ to $k$ do
3:         train (fit parameters) on $k - 1$ pieces and test on the training one piece;
4:     end for
5:     average;
6: end for
```

k-fold Cross Validation makes more efficient use of data, whereas it is computationally very expensive.

7.4 Feature Selection

Algorithm 3 Feature Selection

```
1: Start with $\mathbb{F} = \phi$;
2: for $i$ in feature bags do
3:     Try adding each feature $i$ to $\mathbb{F}$, and see which single feature addition most improves the $S_{dev}$
   performance;
4:     Add the feature to $\mathbb{F}$;
5: end for
```

Illustration:

$x_1, ..., x_5$

(1) ϕ: empty set of features, $h(x) = \theta_0$

(2) $\begin{bmatrix} \phi + x_1 \\ \phi + x_2 \\ ... \\ \phi + x_5 \end{bmatrix} \begin{bmatrix} h_\theta(x) = \theta_0 + \theta_1 x_1 \\ h_\theta(x) = \theta_0 + \theta_1 x_2 \\ ... \\ h_\theta(x) = \theta_0 + \theta_1 x_5 \end{bmatrix}$, x_2 achieves the best performance, $\mathbb{F} = \{x_2\}$.

(3) $\begin{bmatrix} x_2 + x_1 \\ x_2 + x_3 \\ x_2 + x_4 \\ x_2 + x_5 \end{bmatrix}$, $\mathbb{F} = \{x_2, x_4\}$

...