

# ECEN 765

## Machine Learning with Networks

### Fall 2017

TR 8:00-9:15

Office Hour: F 10:00am - noon

Aug 31 R

Lecture I Machine Learning with Networks

---

Sept 5 T

---

Useful link  
PMTK

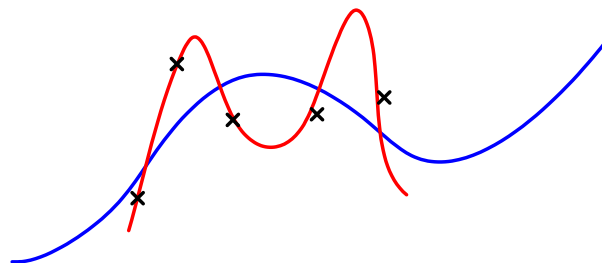
Office Hour  
Friday, 10:00am - noon

### Lecture 1 Machine Learning with Networks

ML  $\begin{cases} \textit{Know your question} \\ \textit{Know your data} \end{cases}$

PAC learnability theory  
(Leslie Valiant)  
Probably Approximately Correct  
VC-dimension

Sparse learning: feature selection  
Example



- ① reduce the number of parameters
- ② reduce the number of features

	Feature I	Feature II	...
Param I			
Param II			
⋮			

only use Feature I & II, not all features

Feature selection is different from dimension reduction

Kernel methods:

$$\langle \vec{x}_1, \vec{x}_2 \rangle \rightarrow \langle f(\vec{x}_1), f(\vec{x}_2) \rangle$$

Ensemble learning (e.g., Adaboost)

Deep learning

Bayesian learning

Difference between ML & Statistical

Inference: ML does not know the model, SI knows the model (by hypothesis) and does the hypothesis test

ML is not accurate (with uncertainty)

DL is accurate but does not know the confidence

BL is accurate and also knows the confidence

## Lecture 2 Probability and Statistics for Machine Learning

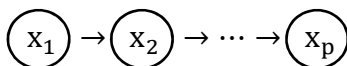
Union bound

$$A_i, \forall i \quad \Pr\left(\bigcup_i A_i\right) \leq \sum_i \Pr(A_i)$$

$$p(x_1, x_2, \dots, x_p) = p(x_1)p(x_2, \dots, x_p|x_1)$$

$$2^p - 1 \text{ params} \quad = p(x_1)p(x_2|x_1)p(x_3, \dots, x_p|x_1, x_2)$$

$$= \prod_{t=2}^p p(x_t|x_{t-1}) p(x_1) \quad \left[ \begin{array}{l} \text{if } x_t \text{ is ONLY dependent on } x_{t-1}, \\ \text{and independent on others} \end{array} \right]$$



1st order Markov chain

important in probability graphical model

PGM's

① Statistical inference / decision making

② Parameter learning

### ③ Structure learning

**Example.** (Laplace's sunrise problem: What is the probability that the sun will rise tomorrow?)

$$\mathcal{D}: \hat{y} = \hat{f}(x)$$

[With the input value  $x$  of the data and the model we estimated (learned), we have inference

$$\hat{y} = \hat{f}(x)]$$

$$\text{Given } x, \Pr(\hat{y} = y) = \theta$$

[For each sample's input  $x^i$ , we have an estimate  $\hat{y}^i$  based on our model, comparing it with the sample's true label/value  $y^i$ , it can be accurate or inaccurate. Suppose the probability of the estimation being accurate for each sample is (the same)  $\theta$ .]

$$\mathcal{T} = \{x^1, x^2, \dots, x^n; y^1, y^2, \dots, y^n\}$$

[The existing data of inputs and outputs.]

$$\mathcal{TD} = \{n \text{ testing data points, all accurate}\}$$

[The event that all previous estimates are accurate.]

$$\Pr(\hat{y}^{n+1} = y^{n+1} | \mathcal{TD}) = ?$$

[Suppose all previous  $n$  testing data are accurate, what is the probability that the next estimation will be accurate?]

[If we already know the sun rose in the last  $n$  days, what is the probability that the sun will also rise tomorrow?]

- ① MLE (maximum likelihood estimation)
- ② MAP (maximum a posteriori estimation)
- ③ Bayesian learning

Machine learning only cares about the final accuracy, i.e., how many samples in the data have accurate estimate based on our model. Machine learning does not care about the randomness / probability distribution of the accuracy, i.e., it does not view the probability of being accurate as a random variable. Thus, **ML can be accurate, but does not know the confidence of the accuracy.** Bayesian learning take the further step of studying the randomness / probability distribution of the accuracy (i.e., the probability distribution of the probability of being accurate), and by assuming some prior distribution of the accuracy, it calculates the posterior distribution of the accuracy based on the existing data. Thus, **BL can be accurate and also know the confidence of the accuracy.**

As an example, suppose we have  $n$  testing data points, and among them  $m$  points are accurate. Denote this event as  $\mathcal{TD}_m = \{n \text{ testing data points, } m \text{ accurate}\}$ . Suppose the accuracy is  $\Pr(\hat{y} = y) = \theta$  for each sample, and the prior distribution of  $\theta$  is  $\Pr(\theta)$ . Based on the conditional probability [ $n$  i.i.d. events with  $m$  truth, Binomial distribution  $m \sim B(n, \theta)$ ]

$$\Pr(\mathcal{TD}_m | \theta) = \binom{n}{m} \theta^m (1 - \theta)^{n-m}$$

the posterior distribution of the accuracy can be calculated as

$$\begin{aligned}\Pr(\theta|\mathcal{T}\mathcal{D}_m) &= \frac{\Pr(\mathcal{T}\mathcal{D}_m, \theta)}{\Pr(\mathcal{T}\mathcal{D}_m)} \\ &= \frac{\Pr(\mathcal{T}\mathcal{D}_m|\theta)\Pr(\theta)}{\int \Pr(\mathcal{T}\mathcal{D}_m|\theta)\Pr(\theta)d\theta}\end{aligned}$$

If we select  $\theta \sim \text{Beta}(a, b)$ , i.e., assuming

$$\Pr(\theta) = \frac{1}{\text{Beta}(a, b)} \theta^{a-1} (1 - \theta)^{b-1}$$

the posterior distribution will have the same form as the prior distribution, although with different parameter ( $\theta \sim \text{Beta}(a + m, b + n - m) | \mathcal{T}\mathcal{D}_m$ ). Thus, Beta distribution and Binomial distribution are called conjugate priors.

Other recommended books on Bayesian statistics from internet:

1. Richard McElreath. *Statistical Rethinking*.
2. John Kruschke. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS and Stan*.
3. Andrew Gelman and John B. Carlin. *Bayesian Data Analysis*.
4. Devinderjit Sivian and John Skilling. *Data Analysis: A Bayesian Tutorial*.
5. Cameron Davidson-Pilon. *Probabilistic Programming and Bayesian Methods for Hackers*.
6. Jean-Michel Marin and Christian Robert. *Bayesian Core: A Practical Approach to Computational Bayesian Statistics*.
7. William M. Bolstad and James M. Curran. *Introduction to Bayesian Statistics*.
8. E. T. Jaynes and G. Larry Bretthorst. *Probability Theory: The Logic of Science*.

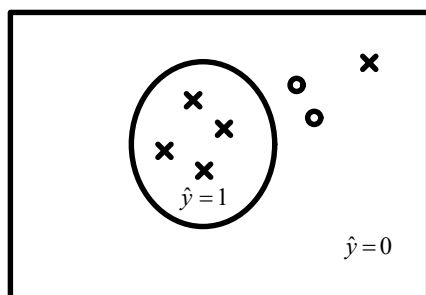
Sept 7 R

machine learning (EE)

- ① System Identification
  - ② Feature Representation
  - ③ Prediction
  - ④ Decision Theory
- } under uncertainty



Bayesian Learning



M:  $\hat{f} \leftarrow$  learned prediction

$$\hat{y} = \hat{f}(x)$$

$$\min_{\hat{f}} \Pr(\hat{y} = \hat{f}(x) \neq y) \quad \text{generalization error}$$

Empirical Risk Minimization

$$\min_{\hat{f}} \sum_{i=1}^5 1(\hat{y}^i \neq y^i) \leftarrow \text{training set}$$

Structural Risk Minimization

**Example.**

Testing data  $\rightarrow \mathcal{D} = \{(x^1, y^1), \dots, (x^5, y^5)\}$

parameter  $\rightarrow \theta = \Pr(\hat{y} \neq y)$

Question:  $\hat{\theta} = ?$

Random Variable:  $0 \leq \theta \leq 1$

data likelihood:  $\Pr(\mathcal{D}|\theta) = \theta^5$

$n$  testing points,  $m$  accurate

$$\text{Prediction} = \binom{n}{m} \theta^m (1 - \theta)^{n-m}$$

**MLE:**

$$\hat{\theta} = \operatorname{argmax}_{\theta} \Pr(\mathcal{D}|\theta)$$

$$= \operatorname{argmax}_{\theta} \binom{n}{m} \theta^m (1 - \theta)^{n-m}$$

$m$  and  $n$  are from the test data set  $\mathcal{D}$

necessary condition to get the extreme point is

$$\frac{\partial \Pr(\mathcal{D}|\theta)}{\partial \theta} = 0$$

$$\Rightarrow m(1 - \theta) = (n - m)\theta$$

$$\Rightarrow m = n\theta$$

$$\Rightarrow \hat{\theta} = \frac{m}{n}$$

**MAP:**

$$\hat{\theta} = \operatorname{argmax}_{\theta} \Pr(\theta|\mathcal{D})$$

prior probability:  $\Pr(\theta)$

if  $\theta \sim \text{Beta}(a, b)$ ,

$$\Pr(\theta) = \frac{1}{\text{Beta}(a, b)} \theta^{a-1} (1 - \theta)^{b-1}, \quad 0 \leq \theta \leq 1$$

$$\begin{aligned} \Pr(\theta|\mathcal{D}) &= \frac{\Pr(\mathcal{D}, \theta)}{\Pr(\mathcal{D})} \\ &= \frac{\Pr(\mathcal{D}|\theta)\Pr(\theta)}{\int \Pr(\mathcal{D}|\theta)\Pr(\theta)d\theta} \end{aligned}$$

$$\begin{aligned}
&= \frac{\binom{n}{m} \theta^m (1-\theta)^{n-m} \cdot \frac{1}{\text{Beta}(a,b)} \theta^{a-1} (1-\theta)^{b-1}}{\int_0^1 \binom{n}{m} \theta^m (1-\theta)^{n-m} \cdot \frac{1}{\text{Beta}(a,b)} \theta^{a-1} (1-\theta)^{b-1} d\theta} \\
&= \frac{\theta^{m+a-1} (1-\theta)^{n-m+b-1}}{\int_0^1 \theta^{m+a-1} (1-\theta)^{n-m+b-1} d\theta} \\
&= \frac{1}{\text{Beta}(m+a, n-m+b)} \theta^{m+a-1} (1-\theta)^{n-m+b-1} \\
&(\theta \sim \text{Beta}(m+a, n-m+b) | \mathcal{D}) \\
&\frac{\partial \text{Pr}(\theta | \mathcal{D})}{\partial \theta} = 0 \\
&\Rightarrow \hat{\theta} = \frac{m+a-1}{n+a+b-2}
\end{aligned}$$

Decision:

$$\text{Pr}(y^{n+1} | \mathcal{D})$$

Bayesian Learning:

$$\begin{aligned}
\text{Pr}(y^{n+1} | \mathcal{D}) &= \int_0^1 \text{Pr}(y^{n+1}, \theta | \mathcal{D}) d\theta \\
&= \int_0^1 \text{Pr}(y^{n+1} | \theta, \mathcal{D}) \text{Pr}(\theta | \mathcal{D}) d\theta \\
&= \int_0^1 \text{Pr}(y^{n+1} | \theta) \text{Pr}(\theta | \mathcal{D}) d\theta \quad [\text{conditionally independent}] \\
&= \int_0^1 \text{Pr}(\hat{y}^{n+1} = y^{n+1} | \theta) \text{Pr}(\theta | \mathcal{D}) d\theta \\
&= \int_0^1 \theta \frac{1}{\text{Beta}(m+a, n-m+b)} \theta^{m+a-1} (1-\theta)^{n-m+b-1} d\theta \\
&= \frac{\text{Beta}(m+a+1, n-m+b)}{\text{Beta}(m+a, n-m+b)} \\
&= \frac{\Gamma(m+a+1)}{\Gamma(m+a) \cdot (n+a+b)} \\
&= \frac{m+a}{n+a+b}
\end{aligned}$$

If  $\theta \sim \text{Uniform}$ , then  $a=b=1$ ; furthermore, if  $m=n$  (meaning all previous results are correct), then the posterior probability is  $\frac{n+1}{n+2}$ , different from 1! [Bayesian argument of Laplace's sunrise problem]

For ML: posterior probability is 1 ( $\theta=1$  based on previous data)

For BL: posterior truth probability is  $\frac{n+1}{n+2}$ , not 1

$\Pr(\hat{y}^{\text{future}}|\mathcal{D})$ : posterior predictive distribution

effective characteristic

Bayesian Decision Theory

Cost:  $L(\hat{y}, y^{\text{desired}}) = (\hat{y} - y^{\text{desired}})^2 \rightarrow$  squared loss function

$$\min_{\hat{x}} \int_Y L(\hat{y}, y^{\text{desired}}) \Pr(\hat{y}|\mathcal{D}) d\hat{y}$$

Dirichlet-Multinomial Conjugate

$$\Pr(\mathcal{D}|\theta) = \binom{50}{10 \ 30 \ 10 \ 0 \ 0} \theta_A^{10} \theta_B^{30} \theta_C^{10} \theta_D^0 \theta_F^0$$

$$\theta_A + \theta_B + \theta_C + \theta_D + \theta_F = 1, 0 \leq \theta_i \leq 1$$

$$\Pr(\theta) = \text{Dirichlet}(\vec{a})$$

$$= \frac{1}{Z(\vec{a})} \prod_{k=1}^K \theta_k^{a_k-1} \mathbb{1}\left(\sum_k \theta_k = 1\right)$$

latent Dirichlet allocation model

Gaussian-Inverse Wishart

$$\hat{y} \sim N(\vec{\mu}, \Sigma)$$

Sept 12 T

---

## Bayesian Learning

### Example 1.

Question: What is the prediction accuracy? [can be called coin flipping probability]

Data: Tested the algorithm on  $n$  testing points with all accurate prediction.

The mathematical question is:

model:  $\theta \in [0, 1]$

① prior:  $\theta \sim \text{Beta}(a, b)$

$a, b$ : hyperparameters

hierarchical Bayesian model

everything is parametric, e.g.,  $\theta$  is parametric by  $a$  and  $b$

② likelihood:  $p(\mathcal{D}|\theta) = \binom{n}{n} \theta^n (1 - \theta)^{n-n} = \theta^n$

Computation:

③ posterior:  $p(\theta|\mathcal{D}) \sim \text{Beta}(n + a, b)$

$a, b$  are pseudo-counts

④ Inference (prediction & decision making)

$$p(\hat{y}^{\text{next}} = y^{\text{next}}|\mathcal{D})$$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta) \cdot p(\theta)}{\int p(\mathcal{D}|\theta) \cdot p(\theta) d\theta}$$

Bayesian statistics calculation: calculate the integral (like by MCMC)

If you are making decision based on threshold, you need to calculate the integral to get the absolute value of the posterior probability. If you only need to know the relative value to compare different posterior probability for different result and then make decision, you might not need to calculate the integral.

*Frequency learning*

$p(\mathcal{D}|\theta) = p(\{(x^1, y^1), (x^2, y^2), \dots, (x^n, y^n)\}|\theta)$  (i. i. d. given  $\theta$ )

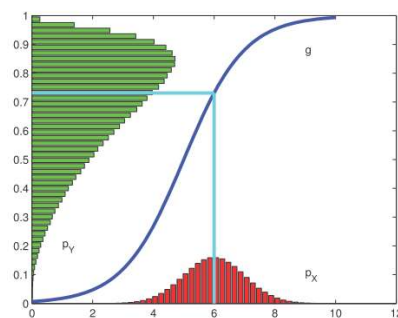
$$= \prod_{i=1}^n p((x^i, y^i)|\theta)$$

→ sampling distribution

$\begin{cases} \text{Empirical Risk Minimization} \rightarrow \text{FL} \\ \text{Structural Risk Minimization} \rightarrow \text{BL} \end{cases}$

In FL, you are imposing structure model

**Example:**



**Figure 5.2** Example of the transformation of a density under a nonlinear transform. Note how the mode of the transformed distribution is not the transform of the original mode. Based on Exercise 1.4 of (Bishop 2006b). Figure generated by bayesChangeOfVar.

Figure from K. Murphy, *Machine Learning: A Probabilistic Perspective*

If  $y$  is has a nonlinear relationship with  $x$ . As the probability of unit area on both  $X$  and  $Y$  spaces should be the same for them to integral to unity over the whole space at the same time, that is, they should have the same unit on probability (but maybe different unit on probability density):

$$\int |p(x)dx| = \int |\tilde{p}(y)dy|$$

$$\Rightarrow |p(x)dx| = |\tilde{p}(y)dy|$$

where  $p(x)$  and  $\tilde{p}(y)$  are the probability density of  $x$  and  $y$  respectively. Thus,

$$\tilde{p}(y) = \left| \frac{dy}{dx} \right| \cdot p(x)$$

because the probability density function is defined to be positive semi-definite. Now if there is a nonlinear relationship between them such as

$$y = f(x) = \frac{1}{1 + e^{-x}}$$



Then the derivative in the RHS of the previous equation might influence the mode of the probability density, meaning that if the mode of  $p(x)$  locates at some  $x^*$ , the mode of  $\tilde{p}(y)$  might be located at some position  $y^*$  different from  $f(x^*)$ , which is undesired.

The fundamental reason of this problem is that Frequency learning cares about the local probability, but not taking a global view. On the contrary, Bayesian learning takes into account all points by taking integral over the parameter on the whole space. This way, the influence of the derivative is removed, which can be clearly seen from the following identity:

$$|p(x)dx| = |\tilde{p}(y)dy|$$

This is important when calculating probability, expectation, etc. for Bayesian learning.

## Standard Statistical Inference

*Always put things not interesting as Null Hypothesis*

### Confusion Matrix

	Accept $H_0$	Reject $H_0$
$H_0$	TN	FP
$H_1$	FN	TP

Positive means that our conclusion is to reject the null hypothesis.

FP is called the type 1 error, meaning that the real result is negative ( $H_0$ ), but our conclusion is positive (to reject  $H_0$ ).

FN is called the type 2 error, meaning that the real result is positive ( $H_1$ ), but our conclusion is negative (to accept  $H_0$ ).

negative events: accept  $H_0$

positive events: reject  $H_0$

$$\begin{cases} \text{precision} = \frac{|TP|}{|TP| + |FP|} \\ \text{recall/sensitivity} = \frac{|TP|}{|FN| + |TP|} \\ \text{specificity} = \frac{|TN|}{|TN| + |FP|} \end{cases}$$

**Precision** describes the accuracy (proportion of correct results) of our conclusion among all the results that *we conclude to be positive*. The higher the precision, the more proportion of all our positive conclusions is really positive.

**Recall/sensitivity** describes the accuracy (proportion of correct results) of our conclusion among all the *real positive* results. The higher the sensitivity, the more proportion of all the real positive results will be detected by us.

**Specificity** describes the accuracy (proportion of correct results) of our conclusion among all the *real negative* results. The higher the specificity, the less proportion of real negative results mistakenly alarmed to be positive by us.

ROC curve

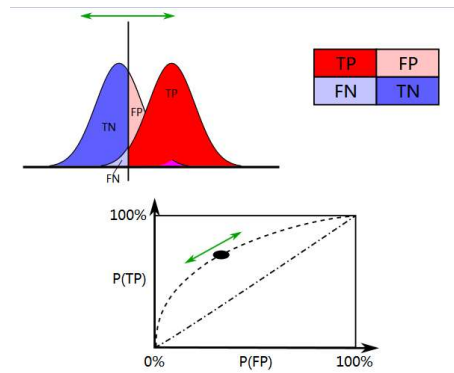


Figure from Internet

The horizontal axis is FPR (False Positive Ratio), which equals to 1-Specificity; the vertical axis is Sensitivity.

PR curve (how your accurate precision are actually true)

The horizontal axis is Precision; the vertical axis is Recall (Sensitivity).

More powerful means more sensitivity.

$$\alpha = 1 - \text{Specificity}$$

$$\text{Power} = 1 - \beta = \text{Sensitivity}$$

$$\beta = P(\text{Accept } H_0 | H_1) = \frac{|FN|}{|FN| + |TP|} = 1 - \text{Sensitivity}$$

We want low  $\alpha$  and high  $1 - \beta$ .

Hypothesis Testing:

model:  $\theta$ : random variable / model parameter of interest

(Bayesian)

(Frequentist)

data: X: toss the coins, e.g. 10 times, 6 heads, and X is the number of heads from 10 tosses, i.e., X=6 in this case.

Rejection Region based on X:

$$\text{testing statistics} \begin{cases} T(X) \in R: \text{Reject } H_0 \\ \text{otherwise: Accept } H_0 \end{cases}$$

$$R = \{T(X) = X > 8\}$$

$$P(T(X) > 8 | \theta = 0.5) \quad \text{Type I error} \\ = \alpha \quad \text{significance level}$$

Power of type II error:

$$\text{power} = 1 - P(T(X) \leq 8 | \theta \neq 0.5)$$

In this example, we are conducting experiment by tossing the coin, and we want to examine if the coin is fair ( $\theta \doteq 0.5$ ). Our testing  $T(X)$  is the number of heads from the tosses ( $T(X) = X$ ).

The null hypothesis  $H_0$  (things not interesting) is  $\theta = 0.5$ . We can make our rejection region R to

be  $\{T(X) = X > 8\}$ , and thus Type I error (FP) will be  $\{\text{Reject } H_0 | H_0\} = \{X > 8 | \theta = 0.5\}$ , and the probability of Type I error is  $\alpha$ , which is also called significance level in statistics. Type II error is  $\{\text{Accept } H_0 | H_1\}$ , which will be  $\{X \leq 8 | \theta \neq 0.5\}$ , and the probability of Type II error is the sensitivity.

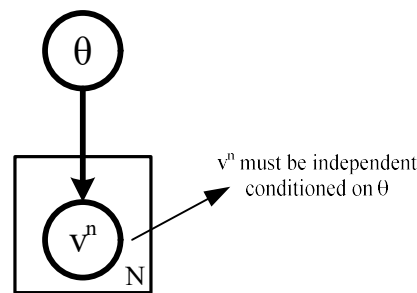
$$1 - \alpha = \text{specificity}$$

$$1 - \beta = \text{power} = \text{sensitivity}$$

subjective & difficult to do the calculation & Hypothesis Test

### Lecture 3 From Bayesian Statistics, to Learning, and to Model Selection

Plate Representation of PGMs



Naïve Bayes Classification (NBC)

**Example.** (Frequency Learning)

Question: predict  $C$  (label outcome) given feature  $(X)$

Data:  $\vec{X} = (x_1, x_2, \dots, x_5)$ ;  $C = \{S, E\}$

model:

$$\begin{aligned} P(\vec{X}, C) &= p(\vec{X} | C) p(C) \\ &= \prod_{i=1}^5 p(X_i | C) p(C) \end{aligned}$$

where  $p(C)$  is percentage of people in the whole country, and  $X_i$  are conditionally independent.

As all features are binary character,  $p(C)$  need only one parameter  $\theta$ , and  $p(\vec{X} | C)$  need  $(2^5 - 1) \times 2$  parameters. But if we employ Naïve Bayes method, since  $p(X_i | C)$  only needs 2 parameters ( $p(X_i = 1 | C = 1)$  and  $p(X_i = 1 | C = 0)$ ), the total number of parameters will be  $2 \times 5 + 1 = 11$ .

Now instead of one  $\theta$ , we have 11  $\theta$ 's, and we will assume they are independent. Section 10.2 have Bayesian learning example.

## Naïve Bayes Classifier

Question:  $\Pr(c^* = 1 | \vec{x}^*; \mathcal{D})$ Data:  $\mathcal{D} = \{(\vec{x}^1, c^1), (\vec{x}^2, c^2), \dots, (\vec{x}^n, c^n)\}$ model:  $\theta = \{P(c = 1); P(\vec{x}|c = 1); P(\vec{x}|c = 0)\}$ 

Assumption:

$$p(\vec{x}|c = 1/0) = \prod_{i=1}^D p(x^i|c = 1/0) \quad \text{totally D different features}$$

$$p(x^i|c = 1/0) = \theta_{x^i|c}$$

$$\textcircled{1} \begin{cases} p(x^i = 1|c = 0) \\ p(x^i = 1|c = 1) \end{cases}$$

$$\textcircled{2} x^i|c = 1 \sim \mathcal{N}(\mu_1^i, \sigma_1^{i^2})$$

$$\text{e.g. prior: } \mu_1^i \sim \mathcal{N}(\mu^0, \sigma^{0^2}), \sigma_1^{i^2} \sim \text{InvGamma}(\alpha, \nu)$$

Standard NBC:  $p(c^*|x^*; \theta^*)$ 

Frequency NBC: point estimate

calculate the probability

Bayesian NBC (Bayesian Decision Theory):

nothing to do with the point estimate of  $\theta$ , looking at the whole distribution of  $\theta$ 

Loss function:

$$\ell(\hat{c}, c) = \begin{cases} 1; & \hat{c} \neq c \\ 0; & \hat{c} = c \end{cases}$$

(zero-one loss)

mean-square loss:  $\ell(\hat{y}, y) = (\hat{y} - y)^2$ 

Minimize expected loss:

$$L(\hat{c}) = E_{c|\mathcal{D}}[\ell(\hat{c}, c)] = \int \ell(\hat{c}, c) p(c|\mathcal{D}) dc$$

$$\hat{c} = \hat{c}(\vec{x}) \Rightarrow L(\hat{c}) = \int \ell(\hat{c}(\vec{x}), c) p(c|\vec{x}; \mathcal{D}) dc$$

we need to get  $\min L(\hat{c})$ , or

$$L(\hat{c}) = \sum_{c \in \{0,1\}} \ell(\hat{c}, c) p(c|\vec{x}; \mathcal{D})$$

$$= \begin{cases} p(c = 1|\vec{x}; \mathcal{D}), & \hat{c} = 0 \\ p(c = 0|\vec{x}; \mathcal{D}), & \hat{c} = 1 \end{cases} \rightarrow \text{Bayes error}$$

To get

$$\min_{\hat{c}} L(\hat{c})$$

we have:

if  $p(c = 1|\bar{x}; \mathcal{D}) < p(c = 0|\bar{x}; \mathcal{D})$ ,  $\hat{c} = 0$   
 if  $p(c = 1|\bar{x}; \mathcal{D}) > p(c = 0|\bar{x}; \mathcal{D})$ ,  $\hat{c} = 1$

$$\begin{aligned} p(c^* = 1|\bar{x}^*; \mathcal{D}) &\leftarrow \text{predictive posterior} \\ &= \int_{\Theta} p(c^* = 1, \theta|\bar{x}^*; \mathcal{D}) d\theta \\ &= \int_{\Theta} p(c^* = 1|\theta; \bar{x}^*; \mathcal{D}) p(\theta|\bar{x}^*; \mathcal{D}) d\theta \\ &= \int_{\Theta} p(c^* = 1|\bar{x}^*; \theta) p(\theta|\mathcal{D}) d\theta \end{aligned}$$

$p(\theta|\mathcal{D})$  is posterior

- 1)  $\theta$  does not depend on new coming input data, so  $p(\theta|\bar{x}^*; \mathcal{D}) = p(\theta|\mathcal{D})$ ;
- 2) new label  $c^*$  does not depend on original data  $\mathcal{D}$  because all data are independent, so  $p(c^* = 1|\theta; \bar{x}^*; \mathcal{D}) = p(c^* = 1|\bar{x}^*; \theta)$ .

[Thanks to Tingjui Chang for explanation of these two identities.]

$$= \int_{\Theta} \frac{p(\bar{x}^*|c^* = 1; \theta) p(c^* = 1|\theta)}{p(\bar{x}^*|c^* = 1; \theta) p(c^* = 1|\theta) + p(\bar{x}^*|c^* = 0; \theta) p(c^* = 0|\theta)} \cdot p(\theta|\mathcal{D}) d\theta$$

Section 10.2:

$$p(c^* = 1|\bar{x}^*; \mathcal{D}) \propto p(\bar{x}^*|c^* = 1; \mathcal{D}) p(c^* = 1|\mathcal{D})$$

$$p(c^* = 1|\mathcal{D}) = \int_{\theta_{c^*=1}} p(c^* = 1|\theta_{c^*=1}) p(\theta_{c^*=1}|\mathcal{D}) d\theta_{c^*=1}$$

Optimal Bayes Classification (Lori Dalton)

Bayesian Model Selection/Averaging

Don't want to bias to any model, so typically all models have the same probability ( $p(M_i) = p(M_j)$ )

Bayesian Computation

- ① Inference:  $p(c^*|\theta)$
- ② Decision Making:  $E[\ell(\hat{c}, c)]$
- ③ Model Selection:  $p(M|\mathcal{D})$  or  $p(\mathcal{D}|M)$

$$\text{Bayes factor} = \frac{p(\mathcal{D}|M_i)}{p(\mathcal{D}|M_j)}$$

typically larger than some value, like 10, to prefer one of them (if only larger than 1, it would be not safe)

$$\begin{aligned} p(\mathcal{D}|M_i) &= \int_{\Theta_i} p(\mathcal{D}|\theta_i) p(\theta_i|M_i) d\theta_i \\ &= \int_{\Theta_i} e^{-f(\theta_i)} d\theta_i \end{aligned}$$

$$\approx \int e^{-\theta_i^T A_i \theta_i + b_i^T \theta_i} d\theta_i$$

Laplace's method

find the mode and use quadratic form near the mode

$$f(\theta) = f(\theta^m) + \nabla f(\theta^m) \cdot (\theta - \theta^m) + \frac{1}{2} (\theta - \theta^m)^T \nabla^2 f(\theta^m) (\theta - \theta^m) + O(|\theta - \theta^m|^2)$$

$\theta^m$  is the mode, so  $\nabla f(\theta^m) = \vec{0}$

$$f(\theta) \approx f(\theta^m) + \frac{1}{2} (\theta - \theta^m)^T H (\theta - \theta^m)$$

$$p(\mathcal{D} | M_i) \approx \int_{\theta_i} e^{-[f(\theta^m) + \frac{1}{2}(\theta - \theta^m)^T H (\theta - \theta^m)]} d\theta = e^{-f(\theta^m)} \det(2\pi H^{-1})^{\frac{1}{2}}$$

$$\log p(\mathcal{D} | M_i) \approx \log p(\mathcal{D} | \theta^m, M_i) + \log p(\theta^m | M_i) + \frac{K}{2} \log 2\pi - \frac{K}{2} \log N$$

Sept 19 T

① prior  $p(\theta)$

② likelihood

$\Rightarrow$  ③ posterior  $p(\theta | \mathcal{D}) \rightarrow \text{MLE, MAP} \rightarrow \hat{\theta}$

model is important

If Bayesian Learning:

④  $p(\hat{y} | \mathcal{D})$  predictive posterior  $\rightarrow$  model is not important

⑤ Bayesian Decision Theory

cost/risk:

loss function:  $\ell(\tilde{y}, \hat{y})$

$y$  is true value,  $\hat{y}$  is estimate

$$\min_{\tilde{y}} \{E[\ell(\tilde{y}, \hat{y})]\}$$

$$= \int_{\mathcal{Y}} \ell(\tilde{y}, \hat{y}) p(\hat{y} | \mathcal{D}) d\hat{y}$$

utility:  $\mathcal{U}(\tilde{y}, \hat{y})$

$$\max_{\tilde{y}} E[\mathcal{U}(\tilde{y}, \hat{y})]$$

Bayesian Model Selection/Averaging

Bayesian factor:  $\frac{p(\mathcal{D} | M_1)}{p(\mathcal{D} | M_2)}$

$$p(\hat{y} | \mathcal{D}) = \sum_{i=1}^m p(\hat{y} | M_i; \mathcal{D}) p(M_i | \mathcal{D})$$

$$p(M_i|\mathcal{D}) = \frac{p(\mathcal{D}|M_i)p(M_i)}{p(\mathcal{D})}$$

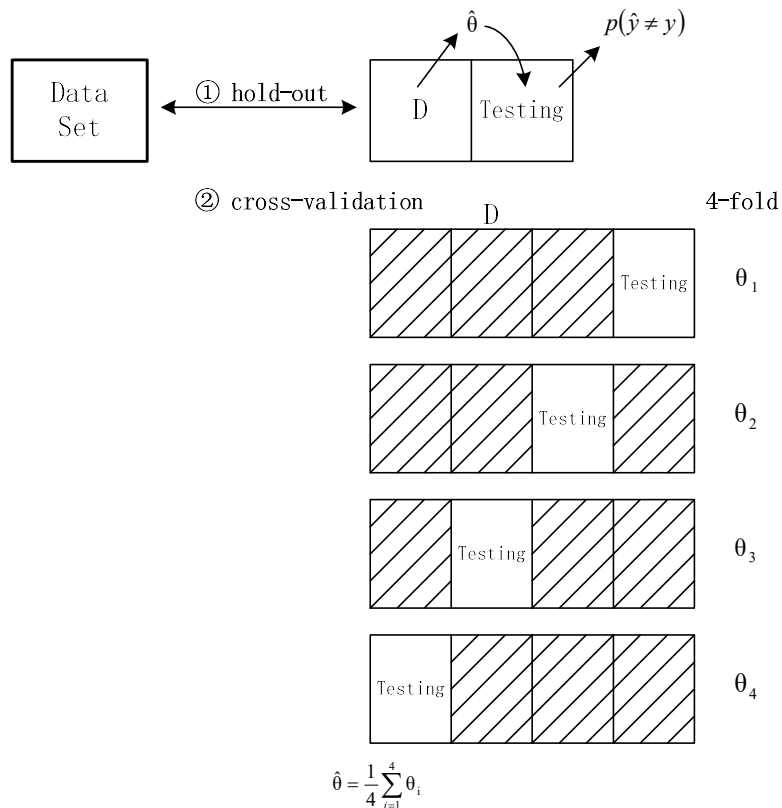
As data has already been observed, model is not independent on data, so cannot assume  $p(M_i|\mathcal{D}) = p(M_i)$ .

{ *Random Forest*  
*Weighted kNN*

① bootstrap ← cross-validation

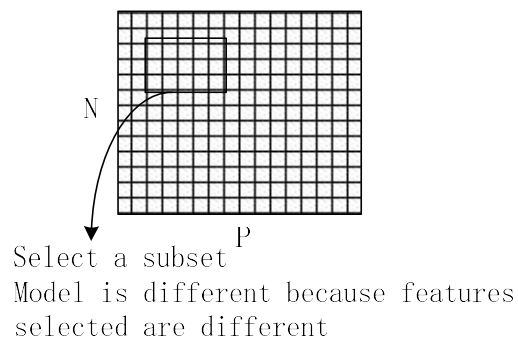
② bagging ← { *Random Forest*  
*Bootstrap Aggregation*

③ boosting



Model is the same for the four

③ Random Forest



kNN: k=1 most complicated (most number of classes)

k=∞ simplest (all things are the same)

To compute  $p(o_a, o_b | H_{\text{indep}})$  :

$$\begin{aligned}
 p(o_a, o_b | H_{\text{indep}}) &= \int_{A,B} p(o_a, o_b, \alpha, \beta | H_{\text{indep}}) d\alpha d\beta \\
 &= \int_{A,B} p(o_a, o_b | \alpha, \beta, H_{\text{indep}}) p(\alpha, \beta | H_{\text{indep}}) d\alpha d\beta \\
 &= \int_{A,B} p(o_a, o_b | \alpha, \beta, H_{\text{indep}}) p(\alpha | H_{\text{indep}}) p(\beta | H_{\text{indep}}) d\alpha d\beta \\
 &\qquad\qquad\qquad \leftarrow \alpha, \beta \text{ are independent} \\
 &= \int_{A,B} p(o_a | \alpha, H_{\text{indep}}) p(o_b | \beta, H_{\text{indep}}) p(\alpha | H_{\text{indep}}) p(\beta | H_{\text{indep}}) d\alpha d\beta \\
 &\qquad\qquad\qquad p(o_a | \alpha, H_{\text{indep}}) \text{ \& } p(o_b | \beta, H_{\text{indep}}) \text{ are likelihood} \\
 &\qquad\qquad\qquad p(\alpha | H_{\text{indep}}) \text{ \& } p(\beta | H_{\text{indep}}) \text{ are prior probability}
 \end{aligned}$$

## Example 2.

two machine learning algorithms (Deep Learning vs SVM)

N fold cross-validation

$$\left. \begin{aligned} \hat{\theta}_{\text{DL}} &= \frac{1}{N} \sum_{i=1}^N \theta_{\text{DL}}^i \\ \hat{\theta}_{\text{SVM}} &= \frac{1}{N} \sum_{i=1}^N \theta_{\text{SVM}}^i \end{aligned} \right\} \begin{array}{l} \text{observed} \\ \text{accuracy} \\ \text{(empirical)} \end{array}$$

DL:  $\theta_{\text{DL}}$ , SVM:  $\theta_{\text{SVM}} \rightarrow$  true prediction accuracy

$H_0: \theta_{\text{DL}} - \theta_{\text{SVM}} \leq 0$

$H_1: \theta_{\text{DL}} - \theta_{\text{SVM}} > 0$

want to reject the null  $\Rightarrow$  DL model is better than SVM

$T(\theta_{\text{DL}}^i, \theta_{\text{SVM}}^i)$  (t test)

$= \hat{\theta}_{\text{DL}} - \hat{\theta}_{\text{SVM}} > c$

$T(X) > c$  is the rejection region

$p(T(X) | \theta_{\text{DL}}, \theta_{\text{SVM}})$

testing data:

$(x^1, y^1) \xrightarrow{\text{DL algorithm } (\theta_{\text{DL}})} \theta_{\text{DL}}^{1(y^1=\hat{y}^1)} \cdot (1 - \theta_{\text{DL}})^{1(y^1 \neq \hat{y}^1)}$

$\vdots$

$(x^n, y^n) \xrightarrow{\text{DL algorithm } (\theta_{\text{DL}})} \theta_{\text{DL}}^{1(y^n=\hat{y}^n)} \cdot (1 - \theta_{\text{DL}})^{1(y^n \neq \hat{y}^n)}$

$\theta_{\text{DL}}^i \sim \binom{n}{m} \theta_{\text{DL}}^m (1 - \theta_{\text{DL}})^{n-m}$

m is the number of good prediction, and n-m is the number of bad prediction



$\bar{\theta}_{DL} \sim \mathcal{N}(\theta_{DL}, \sigma^2)$  averaging over a large number of testing

$\hat{\theta}_{DL} \sim \mathcal{N}(\theta_{DL}, \sigma^2)$ ,  $\hat{\theta}_{SVM} \sim \mathcal{N}(\theta_{SVM}, \sigma^2)$

(assume they have the same  $\sigma$ ; not true in reality)

$\mu = \theta_{DL} - \theta_{SVM} < 0$  (null hypothesis)

model parameter

$T(X) = \hat{\theta}_{DL} - \hat{\theta}_{SVM} \rightarrow$  testing statistics

$\beta = \Pr(T(X) > c | \theta_{DL} - \theta_{SVM} > 0)$

$T(X) = \hat{\theta}_{DL} - \hat{\theta}_{SVM} \sim \mathcal{N}(\mu, 2\sigma^2)$

$$T(X) > c \Leftrightarrow \frac{T(X) - \mu}{\sqrt{2}\sigma} > \frac{c - \mu}{\sqrt{2}\sigma}$$

$$\beta = 1 - \Phi\left(\frac{c - \mu}{\sqrt{2}\sigma}\right)$$

size of test  $T(X) > c$

$$\begin{aligned}\alpha &= \max_{\mu \leq 0} \left(1 - \Phi\left(\frac{c - \mu}{\sqrt{2}\sigma}\right)\right) \\ &= 1 - \Phi\left(\frac{c}{\sqrt{2}\sigma}\right)\end{aligned}$$

$\alpha$  is the size (significant level, confidence) of the test

$\mu \leq 0$  is the null hypothesis  $H_0$

$$\alpha \Rightarrow c = \sqrt{2}\sigma\Phi^{-1}(1 - \alpha)$$

$$\hat{\theta}_{DL} = \frac{1}{N} \sum_{i=1}^N \theta_{DL}^i, \theta_{DL}^i \sim \mathcal{N}(\theta_{DL}, \sigma^2)$$

$$\hat{\theta}_{DL} \sim \mathcal{N}\left(\theta_{DL}, \frac{\sigma^2}{N}\right)$$

$$\Rightarrow c = \frac{1}{\sqrt{N}} \sqrt{2}\sigma\Phi^{-1}(1 - \alpha)$$

test more to get smaller threshold to reject (more confidence to reject due to smaller  $\alpha$  with the same  $c$ )

Bayesian Hypothesis Test:  $p(\mu > 0 | \mathcal{D})$

Sept 21 R

---

Multiple Hypothesis Testing

$\alpha$  = Probability of FP (false positive)

M (4000 genes) tests

Probability of at least one false positive  $\leq 1 - (1 - \alpha)^M \leq \alpha M$   
compound rate

Bonferonni Correction

instead of using  $\alpha$ , use  $\frac{\alpha}{M}$  in the beginning

related to overfitting

if the desired p-value is  $p_0$ , then the actual p-value used for the testing have threshold  $\frac{p_0}{M}$

if the number of model parameter change with the number of data, it is non-parametric learning  
semi-parametric model

**Bayes error is the minimum that can be achieved**

$$e_{\text{Bayes}} \leq e_{\text{nn}} \leq 2e_{\text{Bayes}} + c$$

$c$  is related to  $c$ -Lipshitz function

(For nearest-neighbor algorithm)

$$\text{For kNN: } e_{\text{Bayes}} \leq e_{\text{nn}} \leq \left(1 + \sqrt{\frac{8}{k}}\right) e_{\text{Bayes}} + c$$

Basic concepts:

$c$ -Lipshitz function:  $|f(x) - f(x')| \leq c|x - x'|$

$d(x, x') = \langle x - x', x - x' \rangle \leftarrow$  inner product

$d'(x, x') = \langle \psi(x) - \psi(x'), \psi(x) - \psi(x') \rangle$   
 $= K(x, x') \leftarrow$  Kernel function

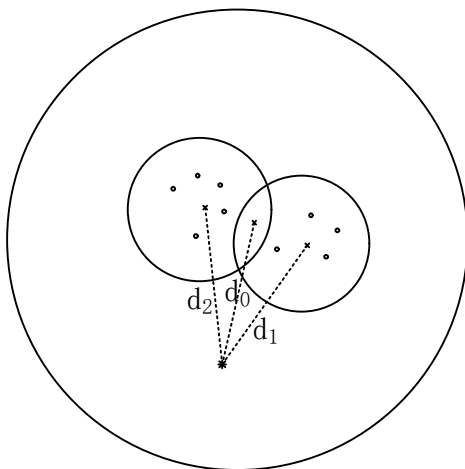
$$d = (x - x')^T \Sigma^{-1} (x - x')$$

$\Sigma$  is symmetric, so  $\Sigma = B^T B$

$$d = (Bx - Bx')^T (Bx - Bx')$$

using  $B$ , we can remove noise

Procrustes distance



if  $d_0 > d_1$  &  $d_0 > d_2$ , does not need to compute lower level distance (like tree)

missing values:

① imputation

② matrix completion  $\leftarrow$  compressive sensing

ADMM

model selection: hold the testing data, and do cross-validation on the training data and select the optimal  $k$  in the projection, need to explain the testing error.

$$(*) \quad e_{\text{Bayes}} \leq e_{\text{nn}} \leq 2e_{\text{Bayes}} + c \quad (\text{Binary classification})$$

**Proof:**

$$\begin{aligned} e_{\text{nn}} &= E_D[y(x) \neq y_{\text{nn}}(x)|x] \\ &= E_D[\Pr(y(x) \neq y(x_{\text{nn}})|x)] \end{aligned}$$

Let  $\eta(x) = \Pr(y(x) = 1|x)$  is  $c$ -Lipshitz.

$x$  can be only 0 or 1

$$\begin{aligned} \Pr(y(x) \neq y(x_{\text{nn}})|x) &= \eta(x)(1 - \eta(x_{\text{nn}})) + \eta(x_{\text{nn}})(1 - \eta(x)) \\ &\quad \eta(x): y(x) = 1, \eta(x_{\text{nn}}): y(x_{\text{nn}}) = 1 \\ &= \eta(x) + \eta(x_{\text{nn}}) - 2\eta(x)\eta(x_{\text{nn}}) \\ &= 2\eta(x)(1 - \eta(x)) + (\eta(x) - \eta(x_{\text{nn}}))(2\eta(x) - 1) \end{aligned}$$

Let  $\eta = \eta(x)$ ,  $\eta' = \eta(x_{\text{nn}})$ ,

$$\begin{aligned} &E_D[y(x) \neq y_{\text{nn}}(x)|x] \\ &= E_D[2\eta(1 - \eta) + (\eta - \eta')(2\eta - 1)] \\ &= E_D[2\eta(1 - \eta)] + E_D[(\eta - \eta')(2\eta - 1)] \\ &\leq E_D[2\eta(1 - \eta)] + 2E_D[\eta - \eta'] \\ &\leq E_D[2\eta(1 - \eta)] + 2cE_D[|x - x_{\text{nn}}|] \\ &\equiv E_D[2\eta(1 - \eta)] + 2c' \\ &\Rightarrow (*) \Leftrightarrow E_D[\eta(1 - \eta)] \leq e_{\text{Bayes}} = E_D[\min(\eta, 1 - \eta)] \end{aligned}$$

$$E_D[\eta(1 - \eta)] = E_D[\max * \min] \leq E_D[\min]$$

The left side is trivial, because

$$e_{\text{nn}} = E_D[\eta(1 - \eta') + \eta'(1 - \eta)] \geq E_D[\eta * \min + \min * (1 - \eta)] = e_{\text{Bayes}}$$

Q.E.D.

$$\psi(x) \begin{cases} \text{PCA} \\ \text{manifold embedding} \rightarrow \text{like spiral data model (use Laplacian embedding)} \\ \text{kernel space} \end{cases}$$

Sept 26 T

---

UML Chapters 2, 3, 5, 6, 9

① parametric learning  $\rightarrow$  non-parametric learning (number of parameters)

② NBC  $\rightarrow$  kNN (supervised learning)

$\uparrow$

number of parameter is fixed

③ PCA (unsupervised)  $\sim$  exponential family PCA  $\rightarrow$  generalized linear models

$\downarrow$

matrix factorization ( $X \approx BZ$ )

unimodal distribution assumption (only one Gaussian distribution)  
noise model of the data:

$$\mathbf{x}^n \approx \tilde{\mathbf{x}}^n + \boldsymbol{\varepsilon}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

In this expression,  $\tilde{\mathbf{x}}^n$  is some data reconstructed by linear combination of basis vectors in a subspace, or equivalently, it is the projection of all data on one subspace, and  $\boldsymbol{\varepsilon}$  is the error between the projected data and the original data.

probabilistic interpretation:

$$\mathbf{x}^n \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

$$\boldsymbol{\mu} \sim \mathbf{c} + \mathbf{B}\mathbf{z}^n$$

$$\mathbf{z}^n = \sum_j z_j^n \mathbf{b}^j$$

in the expression of  $\boldsymbol{\mu}$ ,  $\mathbf{z}^n$  should be interpreted as the coordinates of the data point  $\mathbf{x}^n$  with respect to the basis vectors  $\{\mathbf{b}^j\}$ , and the mean of coordinates of all data points should be zero because the projected data are randomly scattered around the center  $\mathbf{c}$ , i.e.,

$$\frac{1}{N} \sum_{n=1}^N \mathbf{B}\mathbf{z}^n = \mathbf{0}$$

MLE:

$$\max \Pr(\mathbf{X} | \mathbf{c}, \mathbf{Z}, \mathbf{B}, \Sigma)$$

$$= \prod_{n=1}^N \Pr(\mathbf{x}^n | \mathbf{c}, \mathbf{z}^n, \mathbf{B}, \Sigma)$$

$$= \prod_{n=1}^N \frac{1}{\sqrt{2\pi} |\Sigma|^{1/2}} e^{-\frac{1}{2} (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)^T \Sigma^{-1} (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)}$$

negative log-likelihood

$$-\log \Pr(\mathbf{X} | \mathcal{D}) = \log(\sqrt{2\pi} |\Sigma|^{1/2}) + \frac{1}{2} (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)^T \Sigma^{-1} (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)$$

constant minimize this term

If we set  $\Sigma = \sigma^2 I$  (i.e., assuming white noise), the second term will be proportional to

$$\frac{1}{2\sigma^2} (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)^T (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)$$

residual

$\mathbf{c}, \mathbf{Z}, \mathbf{B}$  are parameters.

1) minimum reconstruction error

2) maximum variance

both view can derive the same result

supervised learning has goal: minimize prediction generalized error  
for unsupervised learning, what is the goal?

④ Optimization (closed-form solutions)

⑤ Extensions of PCA:

1) Kernel PCA

2) Probability interpretation → Probabilistic PCA → Factor Analysis

3) NMF → compressed sensing

4) ePCA → types of data (binary, categorical count, ...)

5) supervised PCA

→ CCA/PLS (partial least square)

(Canonical Correlation Analysis)

6) ICA

→ independent component analysis (using entropy)  
most uncertainty

$$\min_{\mathbf{c}} \sum_{n=1}^N |\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n|^2$$

given                      both fixed

physically:  $\mathbf{c} = \frac{1}{N} \sum_{n=1}^N (\mathbf{x}^n - \mathbf{B}\mathbf{z}^n)$  is the optimal solution for the above problem

$$\approx \frac{1}{N} \sum_{n=1}^N \mathbf{x}^n \quad (\text{centering})$$

( $\because \mathbf{x}^n = \mathbf{c} + \mathbf{B}\mathbf{z}^n + \boldsymbol{\varepsilon}$ , we can move the origin to  $\mathbf{c}$ )

1. Residual minimization view -- minimize the reconstruction error

$$\begin{aligned} & \min_{\mathbf{c}} \sum_{n=1}^N (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n)^T (\mathbf{x}^n - \mathbf{c} - \mathbf{B}\mathbf{z}^n) \\ &= \min_{\mathbf{c}} \sum_{n=1}^N [(\mathbf{x}^n - \mathbf{B}\mathbf{z}^n)^T (\mathbf{x}^n - \mathbf{B}\mathbf{z}^n) - 2\mathbf{c}^T (\mathbf{x}^n - \mathbf{B}\mathbf{z}^n) + \mathbf{c}^T \mathbf{c}] \\ & \quad (\text{a quadratic function of } \mathbf{c}) \\ & \Rightarrow N\mathbf{c}^* = \sum_{n=1}^N (\mathbf{x}^n - \mathbf{B}\mathbf{z}^n) \end{aligned}$$

Frechet mean

Express the above results in matrix form (after centering):

$$\begin{aligned} & X_{P \times N}, B_{P \times D}, Z_{D \times N} \\ & \min E(B, Z) = \text{tr}((X - BZ)^T (X - BZ)) \\ & \text{s. t. } B^T B = I \end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial Z} &= \nabla_Z \text{tr}((X - BZ)^T(X - BZ)) \\
&= 2(X - BZ)^T B \\
&= 0 \\
\Rightarrow X^T B &= Z^T B^T B = Z^T \\
\Rightarrow Z &= B^T X \\
\Rightarrow E(B) &= \text{tr}((X - BB^T X)^T(X - BB^T X)) \\
&= \text{tr}(X^T X - X^T B B^T X - X^T B B^T X + X^T B B^T B B^T X) \\
&= \text{tr}(X^T (I - BB^T) X) \quad (B^T B = I) \\
&= \text{tr}(X X^T (I - BB^T)) \\
\frac{1}{N-1} X X^T &= S_{P \times P} \quad \text{sample covariance}
\end{aligned}$$

The original optimization problem is reduced to

$$\begin{aligned}
&\min \text{tr}(S(I - BB^T)) \\
&\text{s. t. } B^T B = I
\end{aligned}$$

Lagrangian function

$$\mathcal{L}(B) = \text{tr}(S(I - BB^T)) + \text{tr}(\Lambda(B^T B - I))$$

where  $\Lambda$  is the Lagrangian multiplier matrix

$\Lambda: \Lambda_{D \times D}$  (usually) assumed to be diagonal

$$\Rightarrow SB = B\Lambda$$

$$\rightarrow Sb = b\lambda$$

$\rightarrow$  eigenvectors of  $S$

$$\Rightarrow \text{tr}(SBB^T) = \text{tr}(BAB^T) = \text{tr}(B^T B\Lambda) = \text{tr}(\Lambda)$$

and the minimum residual is:

$$\text{tr}(S) - \text{tr}(\Lambda)$$

## 2. Variance maximization view – maximize explained variance with selected components

We want to maximize the variance of the projection of all the data  $X$  on some direction  $\mathbf{b}$ , i.e. (projection is obtained through inner product for one data)

$$\max_{\mathbf{b}} \text{Var}(X^T \mathbf{b}) = \max_{\mathbf{b}} E(\mathbf{b}^T X X^T \mathbf{b})$$

$$\text{s. t. } \mathbf{b}^T \mathbf{b} = 1$$

$$X X^T = (N-1)S$$

If  $P$  (number of all features) can be very large

$$S_{P \times P}: X X^T \mathbf{b} = \lambda \mathbf{b}$$

can not be done on computer, so we need to use some alternative method.

SVD method:

$$X = U_{P \times D} D_{D \times D} V_{D \times N}^T$$

(the subscript  $D$  should not be confused with the matrix  $D$ , they have different meaning)

$$S = X X^T$$

$$\begin{aligned}
&= \mathbf{U} \mathbf{D} \mathbf{V}^T \mathbf{V} \mathbf{D}^T \mathbf{U}^T \\
&= \mathbf{U} \mathbf{D}^2 \mathbf{U}^T \\
&\mathbf{X}^T \mathbf{X} \mathbf{X}^T \mathbf{b} = \lambda \mathbf{X}^T \mathbf{b} \\
&\underbrace{\mathbf{X}^T \mathbf{X}}_{N \times N} \mathbf{X}^T \mathbf{b} \rightarrow \tilde{\mathbf{b}} \\
&\lambda \mathbf{b} = \mathbf{X} \mathbf{X}^T \mathbf{b} = \mathbf{X} \tilde{\mathbf{b}} \\
&\Rightarrow \mathbf{b} = \lambda^{-1} \mathbf{X} \tilde{\mathbf{b}}
\end{aligned}$$

Sept. 28 R

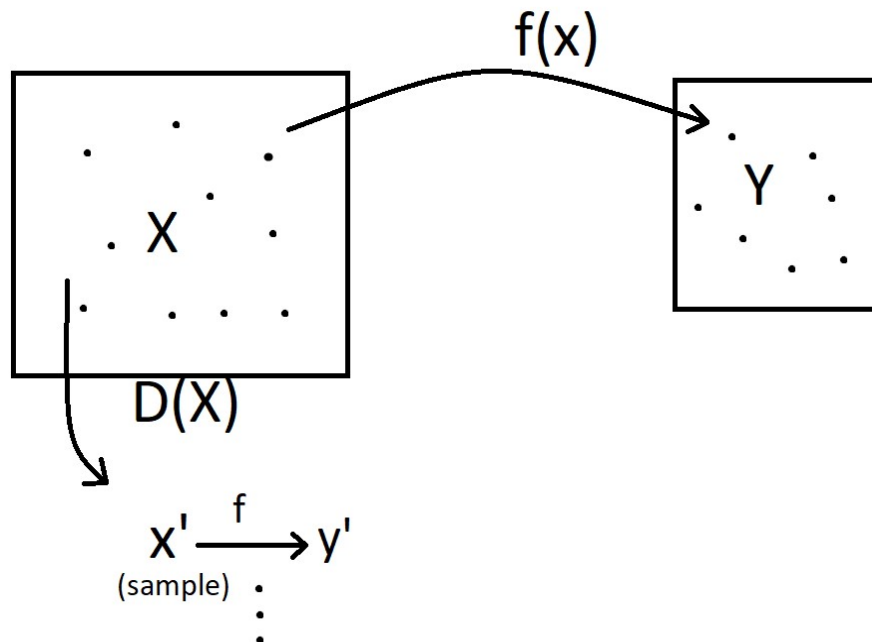
BRML 9.5 is not covered in midterm exam I

Chapters 1, 8, 13 on BRML or 5, 6 on KMP

UML: 2, 3, 5, 6, 9

PAC learnability (**Probably Approximately Correct Learning**)

Model/Hypothesis class: different from hypothesis test. This is about a function, not about parameters. [can make analogy with the relationship and difference between function and functional]



$$\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots\}$$

estimate  $h$ : (hypothesis function)

$$h(x) \in \mathcal{H}, \quad \mathcal{H} \text{ is model space}$$

don't have the function itself, and cannot use functional method to compare the distance between functions only with some function values

VC-dimension

(1) Shattering

Given a  $\mathcal{H}$ , if  $\mathcal{H}$  shatters the training data  $\mathcal{D}$ , it means  $\mathcal{H}$  can generate any possible label combinations of points of  $\mathcal{D}$

(2) VC-dimension is the property of  $\mathcal{H}$  (not property of the data), VC-dimension is the maximum number of points that  $\mathcal{H}$  can shatter.

(3)  $\mathcal{H}$ : d-dimensional linear classifier,  $VC(\mathcal{H}) = d + 1$

Peter Hart (Pattern Recognition)

chapter on neural network

the trick to train neural network

The VC-dimension of a hypothesis class  $\mathcal{H}$  is the maximal size of subset of the feature space that can be shattered by  $\mathcal{H}$ .

Before further discussion, we need first clarify several concepts.

1. Data

We want to learn some rule based on some data. The data we use to learn the rule is called **training data**, denoted by  $\mathcal{D}$ . It contains  $N$  samples, each sample consists of feature and label, i.e.,  $\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$ .

We use  $\mathcal{D}|_x$  to denote the set of features of samples in  $\mathcal{D}$ , i.e.,  $\mathcal{D}|_x = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ .

2. Output learner

The rule  $h$  we learned from the training data is called a **prediction rule**, also called a predictor, a hypothesis, or a classifier. We use the rule to label each data based on its features, i.e., the label we predicted for  $\mathbf{x}^n$  is  $h(\mathbf{x}^n)$ .

3. Data generation

We assume the data samples  $\{(\mathbf{x}^n, y^n)\}$  are generated by some **probability distribution**, and denote the probability distribution **over the feature space** as  $\mathcal{P}$ , i.e.  $\mathbf{x} \sim \mathcal{P}$ .

We assume there is some **correct labeling function**, also called true underlying model,  $f$  that for every  $\mathbf{x}^n$ ,  $y^n = f(\mathbf{x}^n)$ .

4. Error of classifier

Given the probability distribution  $\mathcal{P}$  and the correct labeling function  $f$ , for classifier  $h$ , we have the probability of making prediction mistakes  $h(\mathbf{x}^n) \neq f(\mathbf{x}^n)$ , where  $\mathbf{x}^n$  is a randomly chosen example. This prediction error probability is denoted by  $L_{\mathcal{P},f}(h)$ , and is measured with respect to the data distribution  $\mathcal{P}$  and the correct labeling function  $f$ . It is called **the generalization error, the risk, or the true error of  $h$** .

The error the classifier  $h$  incurs over the training data  $\mathcal{D}$  is called **the training error, the empirical error, or empirical risk of the classifier  $h$** . It is denoted by  $L_{\mathcal{D}}(h)$ .

5. Empirical Risk Minimization (ERM)



The learning paradigm of selecting the predictor  $h$  that minimize the empirical risk (training error)  $L_{\mathcal{D}}(h)$  is called empirical risk minimization (ERM). It has the potential problem of suffering from overfitting.

#### 6. Avoiding Overfitting – Restrict Predictor Search Space

Instead of arbitrarily selecting predictor, we choose in advance (before seeing the data) a set of predictors. This set of predictor is called a hypothesis class, and is denoted by  $\mathcal{H}$ .

For a given class  $\mathcal{H}$  and training data  $\mathcal{D}$ , we want to apply ERM learning rule without overfitting problem.

By restricting to choose a predictor from  $\mathcal{H}$ , we bias it towards some particular set of predictors. Such restrictions are often called an inductive bias.

#### 7. Finite Hypothesis Class

The simplest type of restriction on a class is imposing an upper bound on its size (i.e., the number of predictors  $h$  in  $\mathcal{H}$ ).

With the finite hypothesis class assumption ( $\mathcal{H}$  is finite), we apply ERM learning rule (minimizing the empirical risk  $L_{\mathcal{D}}(h)$ ) for training data  $\mathcal{D}$ , and obtain a classifier, denoted by  $h_{\mathcal{D}}$ .

#### 8. Realizability Assumption

The realizability assumption is, in a finite hypothesis class  $\mathcal{H}$ , there exists some perfect classifier that has zero generalization error, i.e, it will almost never make any mistake on prediction with the data generated according to  $\mathcal{P}$  and labeled by  $f$ :

$$\exists h^* \in \mathcal{H} \quad \text{s.t.} \quad L_{\mathcal{P}f}(h^*) = 0$$

Since this perfect classifier will almost make no mistake on prediction, for any training data  $\mathcal{D}$  generated according to  $\mathcal{P}$ , we have  $L_{\mathcal{D}}(h^*) = 0$  with probability 1 (it will rarely make any mistake). By the definition of  $h_{\mathcal{D}}$ , which is has the minimum empirical risk, we must have  $L_{\mathcal{D}}(h_{\mathcal{D}}) = \min_h L_{\mathcal{D}}(h) = L_{\mathcal{D}}(h^*) = 0$ , thus we have the following corollary:

Under the realizability assumption, for every ERM hypothesis  $h_{\mathcal{D}}$ , we have  $L_{\mathcal{D}}(h_{\mathcal{D}}) = 0$ .

#### 9. i.i.d Assumption

The i.i.d assumption is that the training data are independently and identically distributed (i.i.d.) according to the distribution  $\mathcal{P}$ . Remembering  $\mathcal{D}|_x = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ , we denote the i.i.d. assumption as  $\mathcal{D}|_x \sim \mathcal{P}^N$ , where  $N$  is the size of the training data  $\mathcal{D}$ .

With all the three assumptions above, i.e., finite hypothesis class, realizability, and i.i.d. sampling, we can prove the following theorem.

#### Proof of Fundamental Theorem for PAC Learning – Finite Model Classes:

Let  $\mathcal{H}$  be a finite hypothesis class. Fix  $\delta \in (0, 1)$ ,  $\epsilon < \frac{1}{2}$ . If the size of the training data is large enough:  $N > \frac{\log(|\mathcal{H}|/\delta)}{\epsilon}$ , then for any distribution  $\mathcal{P}$  and labeling function  $f$  satisfying the realizability assumption (i.e., there exists some perfect classifier with zero generalization error),

the probability of getting a non-representative training data  $\mathcal{D}$  can be made sufficiently small:

$\Pr(L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon) < \delta$  over the choice of  $\mathcal{D}$  of size  $N$ . In other words, we have confidence  $1 - \delta$  that the generalization error of the learned predictor is not too large (less than  $\varepsilon$ ).

**Proof:**

For each sampled training data set

$$\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$$

based on the i.i.d. assumption, we know that the features of the training data  $\mathcal{D}|_x = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$  is a random variable with probability distribution  $\mathcal{P}^N$ ,  $\mathcal{D}|_x \sim \mathcal{P}^N$ .

For each sampling result  $\mathcal{D}$ , we can have the corresponding ERM classifier  $h_{\mathcal{D}}$  because the number of classifiers is finite (finite hypothesis class assumption). Note that  $h_{\mathcal{D}}$  is completely determined by  $\mathcal{D}$ , and thus by  $\mathcal{D}|_x$ .

With  $h_{\mathcal{D}}$ , we can get the generalization error  $L_{\mathcal{P}_f}(h_{\mathcal{D}})$ , which is completely determined by  $h_{\mathcal{D}}$  given the distribution  $\mathcal{P}$  and labeling function  $f$ . Thus, we see  $L_{\mathcal{P}_f}(h_{\mathcal{D}})$  is completely determined by sampled training data  $\mathcal{D}$ , i.e., it is a function of  $\mathcal{D}$ , and thus a function of  $\mathcal{D}|_x$ .

We are interested in discussing training data that will lead to ERM classifier with large generalization error:  $L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon$ , and we want to upper bound the probability of getting such training data. The set of all such training data is

$$\{\mathcal{D}|_x : L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}$$

Note that this is a set of set, each element of this set is a training data set that leads to ERM classifier with bad generalization error.

We already know that  $\mathcal{D}|_x$  is a random variable with probability distribution  $\mathcal{P}^N$ , the probability of getting the above set can be written as

$$\mathcal{P}^N(\{\mathcal{D}|_x : L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\})$$

We would like to upper bound this probability

$$\mathcal{P}^N(\{\mathcal{D}|_x : L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}) < \delta$$

Denote the set of bad predictors as  $\mathcal{H}_B$ :

$$\mathcal{H}_B = \{h \in \mathcal{H} : L_{\mathcal{P}_f}(h) > \varepsilon\}$$

meaning that all predictors in this set have large generalization error (all predictors in this set are very easy to make mistaking prediction on data sample generated by distribution  $\mathcal{P}$ ).

Denote the set of misleading training data set as  $M$ :

$$M = \{\mathcal{D}|_x : \exists h \in \mathcal{H}_B, L_{\mathcal{D}}(h) = 0\}$$

Meaning that all training data set in this set have probability to result in some seemingly ‘well performed’ predictor with zero empirical error (i.e., have excellent performance of making

prediction on training data) while have bad performance on generalization (such predictor is usually said to be **overfitting**).

We are interested in the set  $\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}$ . We have already shown the realizability assumption implies perfect ERM predictor (with zero empirical risk)  $L_{\mathcal{D}}(h_{\mathcal{D}}) = 0$ . Thus, for any  $\mathcal{D}|_x$ , if it will lead to bad ERM predictor with large generalization error:  $L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon$ , it will satisfy two conditions simultaneously: 1) perfect performance on the training data (zero empirical risk); 2) bad generalization ability. In other words, the ERM predictor is overfitting, and such training data must be misleading. So we have

$$\mathcal{D}|_x \in M, \quad \forall \mathcal{D}|_x \in \{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}$$

that is

$$\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\} \subseteq M$$

Therefore,

$$\mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}) \leq \mathcal{P}^N(M)$$

We can rewrite  $M$  based on its definition

$$M = \{\mathcal{D}|_x: \exists h \in \mathcal{H}_B, L_{\mathcal{D}}(h) = 0\}$$

as

$$M = \bigcup_{h \in \mathcal{H}_B} \{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}$$

The former is for each training data set, check if it has bad ERM predictor, and collect all predictors of misleading type together. The latter is first fix some bad predictor, and collect all training data sets that show perfect empirical risk with this predictor, and finally collect all these training data sets corresponding to different bad predictors together. These are just two method to traverse all bad predictors and all training data sets.

So we have

$$\mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}) \leq \mathcal{P}^N\left(\bigcup_{h \in \mathcal{H}_B} \{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}\right)$$

and thus

$$\mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}) \leq \sum_{h \in \mathcal{H}_B} \mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\})$$

For any training data set  $\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$ , zero empirical risk  $L_{\mathcal{D}}(h) = 0$  implies the classifier  $h$  will make no mistake on training data, meaning the prediction by  $h$  for each sample  $\mathbf{x}^i$  is the same as the true label

$$h(\mathbf{x}^i) = f(\mathbf{x}^i) \quad \forall \mathbf{x}^i$$

So we have

$$\mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}) = \mathcal{P}^N(\{\mathcal{D}|_x: h(\mathbf{x}^i) = f(\mathbf{x}^i) \forall \mathbf{x}^i\})$$

and i.i.d. assumption gives us

$$\mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}) = \prod_{i=1}^N \mathcal{P}(\{\mathcal{D}|_x: h(\mathbf{x}^i) = f(\mathbf{x}^i)\})$$

For each sample  $\mathbf{x}^i$ ,  $\mathcal{P}(\{\mathcal{D}|_x: h(\mathbf{x}^i) = f(\mathbf{x}^i)\})$  is the probability of correct prediction, and we already know the probability of making mistake is  $L_{\mathcal{P}_f}(h_{\mathcal{D}}) = 1 - \mathcal{P}(\{\mathcal{D}|_x: h(\mathbf{x}^i) = f(\mathbf{x}^i)\})$ , so we have

$$\begin{aligned} \mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}) &= \prod_{i=1}^N (1 - L_{\mathcal{P}_f}(h_{\mathcal{D}})) \\ &= (1 - L_{\mathcal{P}_f}(h_{\mathcal{D}}))^N \\ &< (1 - \varepsilon)^N \\ &\leq e^{-\varepsilon N} \end{aligned}$$

and

$$\begin{aligned} \mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{P}_f}(h_{\mathcal{D}}) > \varepsilon\}) &\leq \sum_{h \in \mathcal{H}_B} \mathcal{P}^N(\{\mathcal{D}|_x: L_{\mathcal{D}}(h) = 0\}) \\ &\leq |\mathcal{H}_B| e^{-\varepsilon N} \\ &\leq |\mathcal{H}| e^{-\varepsilon N} \end{aligned}$$

Setting this upper bound to  $\delta$  will get the condition for training data size  $N$  in the theorem. Q.E.D.

This theorem (See UML Section 2.3.1) is to state that, if we impose an upper bound on the size of the hypothesis class (the number of predictors in the class), i.e., we restrict the class to be finite, then the ERM (empirical risk minimization) learner for this hypothesis class will not overfit, provided it is based on a sufficiently large training sample (this size requirement will depend on the size of the hypothesis class.)

Oct. 3 T

---

$\mathcal{D}$  strong PAC learnability

↳ Chapter 4 Agnostic PAC Learnability

↳ Uniform Convergence PAC Learnability (Chapter 5)

↳ Consistency Learnability

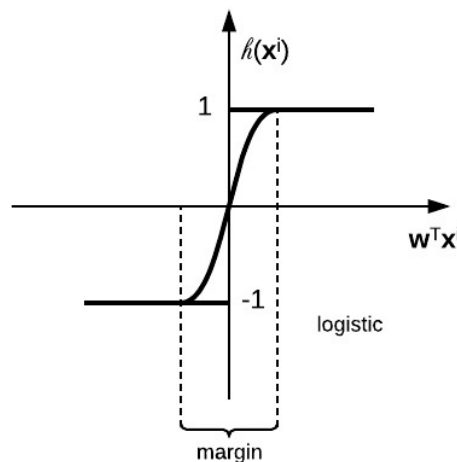
↳ Weak Learnability

**PAC learnability:** A hypothesis class  $\mathcal{H}$  is PAC learnable if there exist a function  $m_{\mathcal{H}} : (0, 1)^2 \rightarrow \mathbb{N}$  and a learning algorithm with the following property: For every  $\varepsilon, \delta \in (0, 1)$ , for every distribution  $\mathcal{P}$  over the feature space, and for every labeling function  $f$  from the feature space to label space  $\{0, 1\}$ , if the realizable assumption holds with respect to  $\mathcal{H}, \mathcal{P}, f$ , then when running the learning algorithm on  $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$  i.i.d. examples generated by  $\mathcal{P}$  and labeled by  $f$ , the algorithm returns a hypothesis  $h$  such that, with probability of at least  $1 - \delta$  (over the choice of the examples),  $L_{\mathcal{P}, f}(h) \leq \varepsilon$ .

In other words, PAC learnability is saying that, if the realizable assumption holds, the generalization error can be almost arbitrary small with sufficient data.

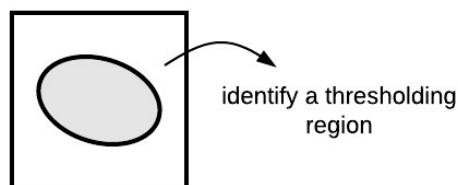
Halfspace

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



use margin to explain why AdaBoost can do better

Otsu thresholding



## ★ Batch Learning v.s. Online Learning

min/max some objective function  $f(\mathbf{w})$  s.t.  $y^i \mathbf{w}^{*T} \mathbf{x}^i > 0, \forall i \in \{1, \dots, N\}$   
 $\downarrow$  Batch learning, put all data  
 dummy  $\frac{1}{2} \|\mathbf{w}\|^2$

separable training dataset:  $\mathcal{D} = \{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$   $y^i \in \{-1, 1\}$   
 $\exists \mathbf{w}, \text{ s.t. } y^i \mathbf{w}^T \mathbf{x}^i > 0, \forall i \in \{1, \dots, N\}$   
 separable  $\leftarrow$  correct classification

**Perceptron Algorithm** (an **online** learning algorithm)

$Y \in \{-1, 1\}$

input: A training set  $\{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$

initialize:  $\mathbf{w}^1 = (0, \dots, 0)$

update:  $\exists i, \text{ s.t. } y^i \mathbf{w}^t \mathbf{x}^i \leq 0, \text{ then } \mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + y^i \mathbf{x}^i$

until no error

**Theorem.** For separable training sets (or realizable cases), the perceptron algorithm stops in a finite number of updates.

Proof:

Since the training data set is finite, its feature set is bounded. Denote the maximum norm of training data features as  $R$ :  $\max_i \|\mathbf{x}^i\| = R$ .

1) Denote the iteration index as  $t$ :

$$\begin{aligned} \|\mathbf{w}^{t+1}\|^2 &= \|\mathbf{w}^t + y^{i_t} \mathbf{x}^{i_t}\|^2 \\ &= \|\mathbf{w}^t\|^2 + 2\mathbf{w}^t \mathbf{x}^{i_t} y^{i_t} + \|\mathbf{x}^{i_t}\|^2 \\ &\leq \|\mathbf{w}^t\|^2 + \|\mathbf{x}^{i_t}\|^2 \\ &\leq \|\mathbf{w}^t\|^2 + R^2 \\ &\leq tR^2 \end{aligned}$$

$(\mathbf{x}^{i_t}, y^{i_t})$  is the sample point in the  $t$ -th step such that  $y^{i_t} \mathbf{w}^t \mathbf{x}^{i_t} \leq 0$ .

2) Cauchy-Schwartz Inequality

$$\frac{\mathbf{w}^{t+1T} \mathbf{w}^*}{\|\mathbf{w}^{t+1}\| \cdot \|\mathbf{w}^*\|} \leq 1$$

$\|\mathbf{w}^*\|$  is constant

$$\mathbf{w}^{t+1T} \mathbf{w}^* = \mathbf{w}^t \mathbf{w}^* + (y^{i_t} \mathbf{x}^{i_t})^T \mathbf{w}^*$$

Introduce  $\delta$ :  $\delta = \min_t \{(y^{i_t} \mathbf{x}^{i_t})^T \mathbf{w}^*\}$

Since  $\mathbf{w}^*$  is a good classifier, we have

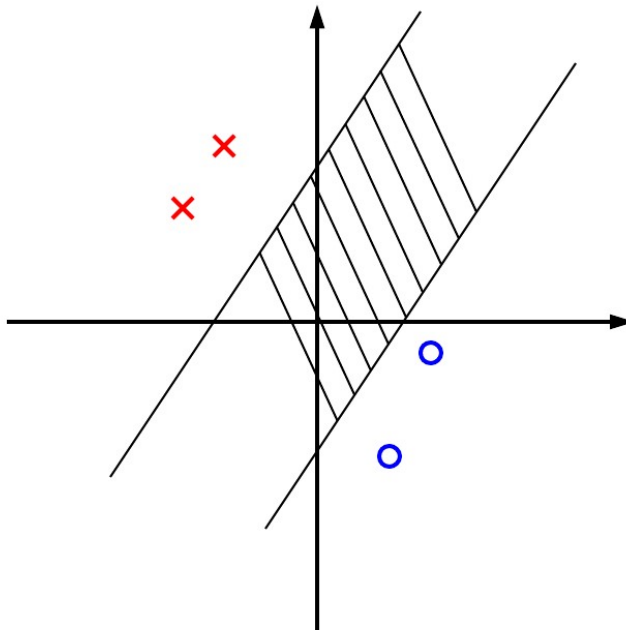
$$(y^{i_t} \mathbf{x}^{i_t})^T \mathbf{w}^* = y^{i_t} \mathbf{w}^{*T} \mathbf{x}^{i_t} > 0$$

Thus we have  $\delta > 0$ .

$$\begin{aligned}
 \mathbf{w}^{t+1^T} \mathbf{w}^* &= \mathbf{w}^{t^T} \mathbf{w}^* + (y^{i_t} \mathbf{x}^{i_t})^T \mathbf{w}^* \\
 &\geq \mathbf{w}^{t^T} \mathbf{w}^* + \delta \\
 &\geq t\delta \\
 \Rightarrow 1 &\geq \frac{\mathbf{w}^{t+1^T} \mathbf{w}^*}{\|\mathbf{w}^{t+1}\| \cdot \|\mathbf{w}^*\|} \geq \frac{t\delta}{\sqrt{t}R \cdot \|\mathbf{w}^*\|} \\
 &\Rightarrow t \leq \left( \frac{R \cdot \|\mathbf{w}^*\|}{\delta} \right)^2
 \end{aligned}$$

$\delta$  is the margin.

Q.E.D.



Midterm exam 1:

Open exam, can take 2 books, class notes

Contains 2 concepts questions and 2 computation questions.

## Boosting

{ *Boosting the confidence*  
*Boosting the efficiency*

## Boosting procedure (Bagging) for confidence

- 1) Sample  $k$  training datasets of size  $N$
- 2) Apply a  $(\varepsilon_0, \delta_0)$ -weak learner to learn  $k$  prediction rules  $h_1, \dots, h_k$   
 $\varepsilon_0$ : accuracy parameter,  $\delta_0$ : confidence parameter
- 3) Sample another training dataset (validation set) of size  $M$  and find the best among the  $k$  prediction rules that show the minimum prediction error on this validation set
- 4) It can be proved that when  $M$  is large enough, we can boost the confidence to achieve  $\varepsilon_0 + \lambda$  accuracy, with  $\lambda$  being arbitrary small

The meaning of this boosting procedure is as following: first we have several weak learners that even if they might have high accuracy ( $\varepsilon_0$  small), our confidence is not large enough ( $\delta_0$  is large). Now, if we have sufficient validation data, we can boost our confidence ( $1 - \delta$ ) to arbitrary large value ( $\delta$  can be arbitrary small).

## Theorem

For any desired  $\delta > 0$ , if we let  $k = \frac{\log(\delta/2)}{\log \delta_0}$  and  $M \geq \frac{\log(4k/\delta)}{2\lambda^2}$ , then with probability at least  $1 - \delta$  over any data set with size  $M$ , the above boosting procedure returns a prediction rule  $h$  with generalization error less than  $\varepsilon_0 + \lambda$ .

Proof:

First we need two inequalities:

## Training step:

$$1) \Pr\left(\min_i L_{\mathcal{P}_f}(h_i) > \varepsilon_0\right) \leq \delta_0^k$$

## Validation step:

$$2) \Pr\left(L_{\mathcal{P}_f}(h) > \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right) \leq \delta'$$

and the goal is to lower bound  $\Pr(L_{\mathcal{P}_f}(h) > \varepsilon_0 + \lambda)$ .

1)  $\min_i L_{\mathcal{P}_f}(h_i) > \varepsilon_0$  happens only when all predictors are bad predictors, so

$$\begin{aligned} \Pr\left(\min_i L_{\mathcal{P}_f}(h_i) > \varepsilon_0\right) &= \Pr(L_{\mathcal{P}_f}(h_1) > \varepsilon_0, \dots, L_{\mathcal{P}_f}(h_k) > \varepsilon_0) \\ &= \prod_i \Pr(L_{\mathcal{P}_f}(h_i) > \varepsilon_0) \\ &\leq \delta_0^k \end{aligned}$$

2) Take the agnostic PAC learnability theorem in UML Chapter 4, we know that we can achieve

$$\Pr\left(L_{\mathcal{P}_f}(h) > \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right) < \delta' \text{ when } M \geq \frac{\log(2k/\delta')}{2\lambda^2}.$$



So we have:

$$\Pr\left(\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right) \geq 1 - \delta_0^k$$

$$\Pr\left(L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right) \geq 1 - \delta'$$

Now, if both  $\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0$  and  $L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda$  happen,  $L_{\mathcal{P}_f}(h) \leq \varepsilon_0 + \lambda$  will also happen, and thus the intersection of the two sets  $\left\{\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right\}$  and  $\left\{L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right\}$  should be a subset of  $\left\{L_{\mathcal{P}_f}(h) \leq \varepsilon_0 + \lambda\right\}$ :

$$\left\{\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right\} \cap \left\{L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right\} \subseteq \left\{L_{\mathcal{P}_f}(h) \leq \varepsilon_0 + \lambda\right\}$$

So

$$\begin{aligned} \Pr(L_{\mathcal{P}_f}(h) \leq \varepsilon_0 + \lambda) &\geq \Pr\left(\left\{\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right\} \cap \left\{L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right\}\right) \\ &= \Pr\left(\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right) + \Pr\left(L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right) \\ &\quad - \Pr\left(\left\{\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right\} \cup \left\{L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right\}\right) \\ &\geq \Pr\left(\min_i L_{\mathcal{P}_f}(h_i) \leq \varepsilon_0\right) + \Pr\left(L_{\mathcal{P}_f}(h) \leq \min_i L_{\mathcal{P}_f}(h_i) + \lambda\right) - 1 \\ &\geq 1 - \delta_0^k - \delta' \end{aligned}$$

Setting  $\delta_0^k = \delta' = \frac{\delta}{2}$  gives

$$\Pr(L_{\mathcal{P}_f}(h) \leq \varepsilon_0 + \lambda) \geq 1 - \delta.$$

Q.E.D.

AdaBoost – Boosting for accuracy

The best way to do AdaBoost is to do extensive search

**Algorithm:**

input:  $\{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$ ; weak learner; number of iteration  $T$

initialize:  $\mathbf{p}^{(1)} = \left(\frac{1}{N}, \dots, \frac{1}{N}\right)$

for  $t = 1:T$

    invoke the weak learner, which could be by ERM or exhaustive search, etc., to derive  $h_t$

    compute the training error:  $\varepsilon_t = \sum_n \rho_n^{(T)} \mathbb{1}(h_t(\mathbf{x}^n) \neq y^n)$

    update the weight  $\alpha_t = \frac{1}{2} \log\left(\frac{1}{\varepsilon_t} - 1\right)$

$$\rho_n^{(T+1)} = \frac{\rho_n^{(T)} e^{-\alpha_t y^n h_t(\mathbf{x}^n)}}{\sum_{n'} \rho_{n'}^{(T)} e^{-\alpha_t y^{n'} h_t(\mathbf{x}^{n'})}}$$

output:  $h(\mathbf{x}) = \text{sign}(\sum_t \alpha_t h_t(\mathbf{x}))$

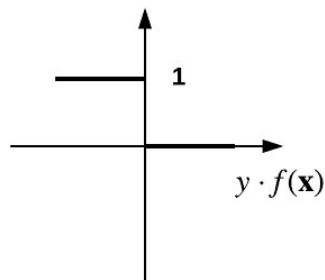
For each weak learner, this algorithm calculates the weighted loss function over the training data set as training error, with some probability distribution for the data  $\{(\mathbf{x}^1, y^1), \dots, (\mathbf{x}^N, y^N)\}$

(instead of equiprobability distribution) as weight; then uses the calculated training error the calculated training error to update the weight and then the probability; it uses some function of the weight sum of weak predictors as the final predictor.

### Surrogate Loss Function

Classification (zero-one loss)

$$\begin{cases} y \in \{-1, 1\} \\ f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x}) \end{cases}$$

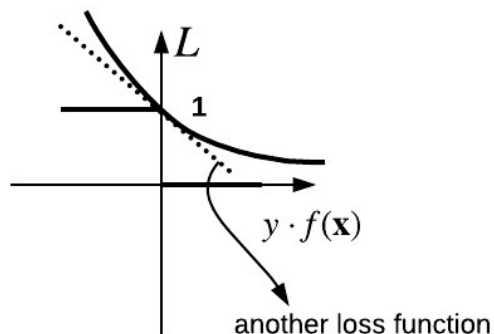


Loss function

$$\begin{aligned} L(y, f) &= \begin{cases} 1, & y \neq f(\mathbf{x}) \\ 0, & y = f(\mathbf{x}) \end{cases} \\ &= \begin{cases} 1, & y \cdot f(\mathbf{x}) < 0 \\ 0, & y \cdot f(\mathbf{x}) > 0 \end{cases} \end{aligned}$$

This loss function is not convex, not continuous, so use other function to approximate it.

Exponential loss function  $e^{-y \cdot f(\mathbf{x})}$



$$f_{t+1} = f_t + \alpha_t h_t$$

$f_t$  is fixed at step  $t$

$\alpha_t$  and  $h_t$  are model parameters

### Convergence Theorem of AdaBoost (Adaptive Boosting)

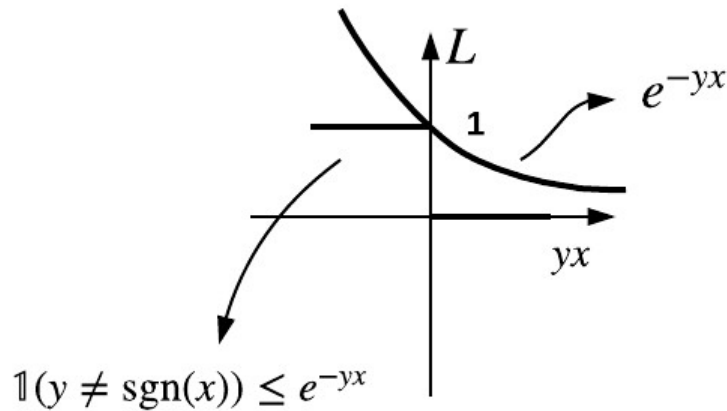
If the weak learner is a  $\left(\frac{1}{2} - \gamma, \delta\right)$ -weak learner, then with probability at least  $1 - \delta$ , the training error of the output hypothesis of AdaBoost  $h(\mathbf{x}) = \text{sign}(\sum_t \alpha_t h_t(\mathbf{x}))$  is at most

$$L_{\mathcal{D}}(h) \leq e^{-2\gamma^2 T}$$

The parameter  $\gamma$  is to guarantee the weak learner is better than random guess.

Proof:

$$\begin{aligned} L_{\mathcal{D}}(h) &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}(y^i \neq h(\mathbf{x}^i)) \\ &= \frac{1}{N} \sum_{i=1}^N \mathbb{1}\left(y^i \neq \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x}^i)\right)\right) \\ &\leq \frac{1}{N} \sum_{i=1}^N e^{-y^i \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^i)} \\ &\stackrel{\text{def}}{=} Z_T \end{aligned}$$



Thus,

$$L_{\mathcal{D}}(h) \leq e^{-2\gamma^2 T}$$

holds true if

$$Z_T \leq e^{-2\gamma^2 T}$$

Now

$$\begin{aligned} Z_T &= \frac{Z_T}{Z_{T-1}} \cdot \frac{Z_{T-1}}{Z_{T-2}} \dots \frac{Z_1}{Z_0} \leq e^{-2\gamma^2 T} \\ \Leftrightarrow \frac{Z_T}{Z_{T-1}} &\leq e^{-2\gamma^2} \text{ with } Z_0 = 1 \left( Z_0 = \frac{1}{N} \sum_{i=1}^N e^{-0} \right) \end{aligned}$$

$$\begin{aligned}\frac{Z_{T+1}}{Z_T} &= \frac{\sum_{i=1}^N e^{-y^i \cdot \sum_{t=1}^{T+1} \alpha_t h_t(\mathbf{x}^i)}}{\sum_{k=1}^N e^{-y^k \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^k)}} \\ &= \sum_{i=1}^N \frac{e^{-y^i \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^i)}}{\sum_{k=1}^N e^{-y^k \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^k)}} \cdot e^{-y^i \cdot \alpha_{T+1} h_{T+1}(\mathbf{x}^i)}\end{aligned}$$

From the definition

$$\begin{aligned}\mathbf{\rho}^{(1)} &= \left(\frac{1}{N}, \dots, \frac{1}{N}\right) \\ \rho_n^{(T+1)} &= \frac{\rho_n^{(T)} \exp(-\alpha_T y^n h_T(\mathbf{x}^n))}{\sum_{n'} \rho_{n'}^{(T)} \exp(-\alpha_T y^{n'} h_T(\mathbf{x}^{n'}))}\end{aligned}$$

it can be proved inductively that

$$\rho_i^{(T+1)} = \frac{e^{-y^i \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^i)}}{\sum_{k=1}^N e^{-y^k \cdot \sum_{t=1}^T \alpha_t h_t(\mathbf{x}^k)}}$$

So we have

$$\begin{aligned}\frac{Z_{T+1}}{Z_T} &= \sum_{i=1}^N \rho_i^{(T+1)} \cdot e^{-y^i \cdot \alpha_{T+1} h_{T+1}(\mathbf{x}^i)} \\ &= \sum_{h_{T+1}(\mathbf{x}^i) = y^i} \rho_i^{(T+1)} \cdot e^{-\alpha_{T+1}} + \sum_{h_{T+1}(\mathbf{x}^i) \neq y^i} \rho_i^{(T+1)} \cdot e^{\alpha_{T+1}} \quad (y^i h_{T+1}(\mathbf{x}^i) = \pm 1) \\ &= e^{-\alpha_{T+1}} \sum_{h_{T+1}(\mathbf{x}^i) = y^i} \rho_i^{(T+1)} + e^{\alpha_{T+1}} \sum_{h_{T+1}(\mathbf{x}^i) \neq y^i} \rho_i^{(T+1)} \\ &= e^{-\alpha_{T+1}} (1 - \varepsilon_{T+1}) + e^{\alpha_{T+1}} \varepsilon_{T+1} \\ &= \sqrt{\frac{\varepsilon_{T+1}}{1 - \varepsilon_{T+1}}} (1 - \varepsilon_{T+1}) + \sqrt{\frac{1 - \varepsilon_{T+1}}{\varepsilon_{T+1}}} \varepsilon_{T+1} \quad \left(\alpha_{T+1} = \frac{1}{2} \log\left(\frac{1}{\varepsilon_{T+1}} - 1\right)\right) \\ &= 2\sqrt{\varepsilon_{T+1}(1 - \varepsilon_{T+1})} \\ &\leq 2\sqrt{\left(\frac{1}{2} - \gamma\right)\left(\frac{1}{2} + \gamma\right)} \\ &= \sqrt{1 - 4\gamma^2} \\ &\leq e^{-2\gamma^2}\end{aligned}$$

Q.E.D.

Dr. Schapire's book chapter explaining AdaBoost

How to derive this algorithm?

$$\begin{aligned}f_t &= f_{t-1} + \alpha_t h_t \\ \min_{\alpha, h} &\sum_{i=1}^N e^{-y^i f_t(\mathbf{x}^i)}\end{aligned}$$

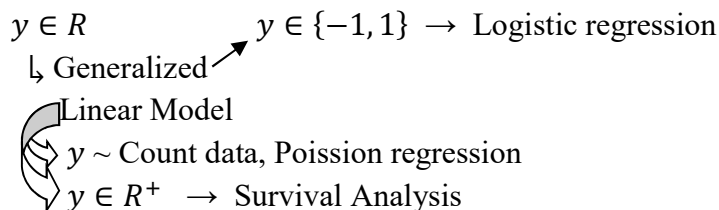
will give it.

Oct. 12 R

---

## Linear Models and Optimization

### ① Regression v.s. Classification



For regression model, cannot use the  $\{0, 1\}$  loss function (which is used to derive the Bayes classification error)

### ② Regression

$$l(y, \hat{y}) = (y - \hat{y})^2 \quad \text{mean square} \rightarrow \text{ERM} \rightarrow \text{least}$$

**utility** (expected loss, need to be minimized):

$$\begin{aligned} U(\hat{y}) &= \mathbb{E}_{y|x; \mathcal{D}}[l(y, \hat{y})] \\ &= \mathbb{E}_y[l(y, \hat{y})|x; \mathcal{D}] \\ \hat{y}^* &= \underset{\hat{y}}{\operatorname{argmin}} U(\hat{y}) \end{aligned}$$

### ③ Bias-Complexity Tradeoff (overfitting)

$$\min_{\hat{y}} \mathbb{E}[(y - \hat{y})^2 | \mathbf{x}], \text{ s.t. } \hat{y} = h(\mathbf{x})$$

The underlying mapping:  $y = f(\mathbf{x}) + \varepsilon$

$$\begin{aligned} &\min_{\hat{y}} \mathbb{E}[(y - \hat{y})^2 | \mathbf{x}] \\ &= \min_{\hat{y}} \mathbb{E}[(y - f(\mathbf{x}) + f(\mathbf{x}) - \hat{y})^2 | \mathbf{x}] \\ &= \min_{\hat{y}} \left\{ \mathbb{E}[(y - f(\mathbf{x}))^2 | \mathbf{x}] + \mathbb{E}[(f(\mathbf{x}) - \hat{y})^2 | \mathbf{x}] + 2\mathbb{E}[(y - f(\mathbf{x}))(f(\mathbf{x}) - \hat{y}) | \mathbf{x}] \right\} \\ &= \min_{\hat{y}} \left\{ \mathbb{E}[(y - f(\mathbf{x}))^2 | \mathbf{x}] + \mathbb{E}[(f(\mathbf{x}) - \hat{y})^2 | \mathbf{x}] + 2\mathbb{E}[(y - f(\mathbf{x}))(f(\mathbf{x}) - \hat{y}) | \mathbf{x}] \right\} \\ &= \sigma^2 + \min_{\hat{y}} \{ \operatorname{Var}[f(\mathbf{x}) - \hat{y} | \mathbf{x}] + \mathbb{E}^2[f(\mathbf{x}) - \hat{y} | \mathbf{x}] \} + 2 \times 0 \quad (f(\mathbf{x}) - \hat{y} \text{ is deterministic}) \\ &= \sigma^2 + \min_{\hat{y}} \{ \operatorname{Var}[\hat{y}] + \operatorname{Bias}^2 \} \rightarrow \text{frequentist learning} \end{aligned}$$

$\sigma^2$  is **underlying variance**

$f(\mathbf{x})$  is constant,  $\operatorname{Var}[\hat{y}]$  is the variance underlying the sample

Bayesian learning:

$$\begin{aligned} &\mathbb{E}_{P(\mathbf{x}, y)}[(y - \hat{y})^2] \\ &= \mathbb{E}_{P(\mathbf{x})} \left[ \mathbb{E}_{P(y | \mathbf{x})}[(y - \hat{y})^2] \right] \end{aligned}$$

$$\begin{aligned}
&= E_{P(\mathbf{x})}[E[(y - \hat{y})^2|\mathbf{x}]] \\
&= E_{P(\mathbf{x})}[E[(y - E(y|\mathbf{x}) + E(y|\mathbf{x}) - \hat{y})^2|\mathbf{x}]]
\end{aligned}$$

$$\min_{\hat{y}} E[(y - \hat{y})^2|\mathbf{x}]$$

given  $\mathbf{x}$ ,  $E(y|\mathbf{x})$  is const,  $\hat{y}$  is also const, so we have

$$\begin{aligned}
&E_{\mathbf{x}} \left[ E \left[ (y - E(y|\mathbf{x}))^2 \middle| \mathbf{x} \right] \right] + E_{\mathbf{x}} [(E(y|\mathbf{x}) - \hat{y})^2] + 0 \\
&= \sigma^2 + E_{\mathbf{x}} [\text{Var}[y|\mathbf{x}]] + E_{\mathbf{x}} [\text{Bias}^2]
\end{aligned}$$

$$(f(\mathbf{x}) = E(y|\mathbf{x}))$$

$E_{\mathbf{x}}[\text{Var}[y|\mathbf{x}]]$  is complexity

complexity-bias tradeoff

neither  $P(\mathbf{x}, y)$  (for BL) nor  $P(y|\mathbf{x})$  (for FL) is given, so have to do empirical risk minimization

④ Model space

models

$$y = h(\mathbf{x}) = \boldsymbol{\theta}^T \boldsymbol{\psi}(\mathbf{x}) + \varepsilon$$

Linear model:

$$\boldsymbol{\psi}(\mathbf{x}) = \mathbf{x}$$

$$h(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} = \sum_{i=1}^p \theta_i x_i$$

(can add an intercept term  $\theta_0$ )

⑤ ERM

$$E = \min_{\boldsymbol{\theta}} \sum_n (y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2$$

$$\Leftrightarrow E = \min_{\boldsymbol{\theta}} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

$\mathbf{X}$ : design matrix

$$\frac{dE}{d\boldsymbol{\theta}} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = 0$$

$$\Rightarrow \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} = \mathbf{X}^T \mathbf{y}$$

$$\Rightarrow \boldsymbol{\theta} = \underbrace{(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}}_{\text{normal equation}}$$

$\mathbf{X}^T \mathbf{X}$  is  $p \times p$ , can have rank less than  $p$ , and the null space is not trivial (infinitely many solutions)

⑥ Ridge Regression

ERM  $\rightarrow$  SRM (structural risk minimization)

$$\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) \rightarrow \min_{\boldsymbol{\theta}} [E(\boldsymbol{\theta}) + R(\boldsymbol{\theta})]$$

penalty regularization term

$$\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) \quad \text{s. t.} \quad R(\boldsymbol{\theta}) \leq 0$$

if  $\mathbf{X}^T \mathbf{X}$  is not invertible, will constrain the null

space to give only one solution

$$R(\boldsymbol{\theta}) = \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}$$

$$\min_{\boldsymbol{\theta}} [(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^T \boldsymbol{\theta}]$$

$$\Rightarrow \boldsymbol{\theta} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

↓  
can be invertible

⑦ model:

$$y = \boldsymbol{\theta}^T \mathbf{x} + \varepsilon$$

$$y \sim \mathcal{N}(\boldsymbol{\theta}^T \mathbf{x}, \sigma^2)$$

$$\max_{\boldsymbol{\theta}} \prod_{n=1}^N P(y^n | \mathbf{x}^n; \boldsymbol{\theta}, \sigma^2)$$

$$\stackrel{\text{def}}{=} P$$

$$P = \prod_{n=1}^N \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2}{2\sigma^2}}$$

$$\log P = \sum_n \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_n \frac{(y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2}{2\sigma^2}$$

maximize  $\log P$  (empirical risk minimization)

⑧  $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, \sigma_{\boldsymbol{\theta}}^2 \mathbf{I})$

not sure which feature will contribute the most

MAP:

$$\max_{\boldsymbol{\theta}} \prod_{n=1}^N P(y^n | \mathbf{x}^n; \boldsymbol{\theta}, \sigma^2) \cdot P(\boldsymbol{\theta})$$

$$(P(\mathcal{D} | \boldsymbol{\theta}) P(\boldsymbol{\theta}))$$

$$\log P = \sum_n \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_n \frac{(y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2}{2\sigma^2} + \log P(\boldsymbol{\theta})$$

$$= \sum_n \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_n \frac{(y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2}{2\sigma^2} + \log \frac{1}{\sqrt{2\pi}\sigma_{\boldsymbol{\theta}}} - \frac{\boldsymbol{\theta}^T \boldsymbol{\theta}}{2\sigma_{\boldsymbol{\theta}}^2}$$

$$\max \log P \Leftrightarrow \min \left\{ \sum_n (y^n - \boldsymbol{\theta}^T \mathbf{x}^n)^2 + \frac{\sigma^2}{\sigma_{\boldsymbol{\theta}}^2} \boldsymbol{\theta}^T \boldsymbol{\theta} \right\}$$

(structural risk minimization)

Why must use Gaussian distribution for  $\boldsymbol{\theta}$ ?

Other options:

- 1) Laplace prior  $\sum |\theta_i|$
- 2) Slab-and-Spike prior indicator function

A simple case: Linear Regression

$X^T X$ : inner product matrix

can use  $\Phi^T \Phi$  for kernel regression

Perceptron Algorithm

$$\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t + c^i \mathbf{x}^i \text{ if } \underbrace{\text{sign}(\boldsymbol{\theta}^t \mathbf{x}^i)}_{\text{half space \& linear}} \neq c^i$$

$c^i$  is outcome label

$c \in \{0, 1\}$

Logistic Regression: (binary)

$$P(c = 1 | \mathbf{x}) = \sigma(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

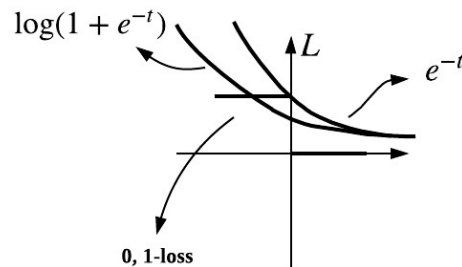
Log-odds ration

$$\begin{aligned} \log \frac{P(c = 1 | \mathbf{x})}{P(c = 0 | \mathbf{x})} &\leq 0 \rightarrow \text{optimal Bayes classifier} \\ &\Leftrightarrow \boldsymbol{\theta}^T \mathbf{x} \leq 0 \\ &\Leftrightarrow \text{sign}(\boldsymbol{\theta}^T \mathbf{x}^i) \neq c^i \end{aligned}$$

do not need infinite iteration for data that is not linear separable, so logistic regression is linear model.

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^N \log(1 + e^{-c'^i \boldsymbol{\theta}^T \mathbf{x}^i})$$

$c' \in \{-1, 1\}$  is logistic loss



$$\begin{aligned} E &= \sum_i \log(1 + e^{-c'^i \boldsymbol{\theta}^T \mathbf{x}^i}) \\ \nabla_{\boldsymbol{\theta}} E &= \sum_i \frac{1}{1 + e^{-c'^i \boldsymbol{\theta}^T \mathbf{x}^i}} \cdot (-e^{-c'^i \boldsymbol{\theta}^T \mathbf{x}^i}) \cdot c'^i \mathbf{x}^i \\ &= - \sum_i \sigma(-c'^i \boldsymbol{\theta}^T \mathbf{x}^i) \cdot c'^i \mathbf{x}^i \\ &= 0 \end{aligned}$$



use numerical method to solve  $\theta$ .

gradient descent

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t - \varepsilon \text{sign}(\nabla f(\mathbf{x}^t))$$

① variable step size

② line search

not a good way  $\leftarrow \varepsilon^* = \underset{\varepsilon}{\operatorname{argmin}} f(\mathbf{x}^{t+1})$

### Line Search

One way to potentially improve on gradient descent is choose some particular direction  $\mathbf{p}_k$  and search along there. We then find the minimum of the one-dimension problem

$$F(\mathbf{x}_k + \lambda \mathbf{p}_k)$$

Finding the optimal  $\lambda^*$  can be achieved using a standard one-dimensional optimization method.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda^* \mathbf{p}_k$$

only one parameter

(univariate minimization)

### Choosing the search directions

It would seem reasonable to choose a search direction that points ‘maximally downhill’ from the current point  $\mathbf{x}_k$ . That is, to set

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$$

However, at least for quadratic function, this is not optimal and lead to potentially zig-zag behavior:

$$\frac{d}{d\lambda} F(\lambda) = \mathbf{p}_k^T \nabla f(\mathbf{x}_{k+1}) = 0$$

This zig-zag behavior occurs for non-axis-aligned surfaces.

One quick fix is known as heavy ball method:

$$\theta_{k+1} = \theta_k - \varepsilon_k \nabla f + \mu_k (\theta_k - \theta_{k-1})$$

Why quadratic function is easy to solve?

$$f(\mathbf{z}) = \frac{1}{2} \mathbf{z}^T \mathbf{A} \mathbf{z} + \mathbf{b}^T \mathbf{z}$$
$$\mathbf{P} \mathbf{A} \mathbf{P}^T \sim \text{diagonalized } \mathbf{A}$$
$$\hat{f}(\hat{\mathbf{x}}) = \frac{1}{2} \hat{\mathbf{z}}^T \mathbf{P} \mathbf{A} \mathbf{P}^T \hat{\mathbf{z}} + \mathbf{b}^T \hat{\mathbf{z}}$$
$$\hat{\mathbf{z}} = \mathbf{P} \mathbf{z}$$

$\mathbf{A}$  is positive-definitive

Gram-Schmidt diagonalization (take time, but hope not to waste time on finding optimum)

Newton’s method (matrix form)

$$\nabla f = -\mathbf{H}_f \Delta \Rightarrow \Delta = -\mathbf{H}_f^{-1} \nabla f$$

(second order approximation:

$$f(\mathbf{z} + \Delta) = f(\mathbf{z}) + \Delta^T \nabla f + \frac{1}{2} \Delta^T \mathbf{H}_f \Delta + O(|\Delta|^3)$$

)

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \varepsilon \mathbf{H}_f^{-1} \nabla f$$

### Local search for optimization

① Gradient descent

$$\mathbf{z}^t = \mathbf{z}^{t-1} - \varepsilon \nabla_{\mathbf{z}} f(\mathbf{z}^{t-1})$$

② Newton's method

$$\mathbf{z}^t = \mathbf{z}^{t-1} - \varepsilon \mathbf{H}_f^{-1}(\mathbf{z}^{t-1}) \nabla_{\mathbf{z}} f(\mathbf{z}^{t-1})$$

For **logistic regression** ( $\mathbf{z} \equiv \boldsymbol{\theta}$ ):

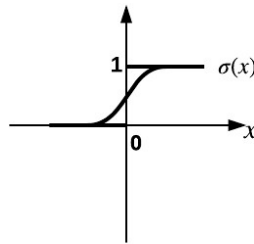
$$\begin{aligned} \boldsymbol{\theta}^t &= \boldsymbol{\theta}^{t-1} + \varepsilon \sum_i \left[ 1 - \sigma(c'^i \boldsymbol{\theta}^{t-1 \text{T}} \mathbf{x}^i) \right] \cdot c'^i \mathbf{x}^i \quad (c'^i \in \{-1, 1\}) \\ &= \boldsymbol{\theta}^{t-1} - \varepsilon \sum_i \left[ \sigma(\boldsymbol{\theta}^{t-1 \text{T}} \mathbf{x}^i) - c^i \right] \cdot \mathbf{x}^i \quad (c^i \in \{0, 1\}) \\ \Rightarrow \boldsymbol{\theta}^{t+1} &= \boldsymbol{\theta}^t - \varepsilon^t \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^t) \\ &= \boldsymbol{\theta}^t + \varepsilon^t \sum_i \underbrace{\left[ c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) \right]}_{\text{error of margin}} \cdot \mathbf{x}^i \end{aligned}$$

this part is the difference from perceptron algorithm  
(using error instead of output label itself)

make it online:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \varepsilon^t \left[ c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) \right] \cdot \mathbf{x}^i$$

only different from perceptron  
algorithm by  $\sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i)$



Looking at it, when **making error**, if  $\sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) = 1$  &  $c^i = 0$  ( $\Leftrightarrow c'^i = -1$ ),

$$c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) = -1$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \varepsilon^t \left[ c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) \right] \cdot \mathbf{x}^i = \boldsymbol{\theta}^t + \varepsilon^t c'^i \mathbf{x}^i$$

if  $\sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) = 0$  &  $c^i = 1$  ( $\Leftrightarrow c'^i = 1$ ),

$$c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) = 1$$

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \varepsilon^t \left[ c^i - \sigma(\boldsymbol{\theta}^t \text{T} \mathbf{x}^i) \right] \cdot \mathbf{x}^i = \boldsymbol{\theta}^t + \varepsilon^t c'^i \mathbf{x}^i$$

so we have, whenever making some error, we update by using  $c'^i$ , and thus get the perceptron algorithm. So, if in online logistic regression we use hard thresholding function (with  $\sigma(\boldsymbol{\theta}^T \mathbf{x}^i) \in \{0, 1\}$ ), it becomes perceptron algorithm.

### Online linear regression

$$E = \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} E = (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T \mathbf{X}$$

$$\boldsymbol{\theta}^t = \boldsymbol{\theta}^{t-1} - \varepsilon \sum_i \underbrace{(y^i - \boldsymbol{\theta}^{t-1T} \mathbf{x}^i)}_{\text{current residual}} \mathbf{x}^i$$

linear regression, logistic regression, perceptron, all have same form of functional.

For linear regression:

$$\mathbf{H} = \mathbf{X}^T \mathbf{X}$$


$$\boldsymbol{\theta}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

For logistic regression:

$$\mathbf{H} = \mathbf{X}^T \mathbf{S} \mathbf{X}$$

$$\mathbf{S} = \text{diag}(\sigma(\boldsymbol{\theta}^T \mathbf{x}) \sigma(-\boldsymbol{\theta}^T \mathbf{x}))$$

calculate S iteratively

BFGS (Quasi-Newton method)  
 four chemists

Oct. 19 R

Support Vector Machine

BRML Chapter 17; UML Chapter 15; KPM Section 14.5

### ※ Machine Learning & Optimization

- |   |   |  |
|---|---|--|
| $\left\{ \begin{array}{l} \text{Global} \\ \text{Optimization} \\ \text{Local} \\ \text{Optimization} \end{array} \right\}$ | ① | Linear regression → Closed-form solution                                     |
|   | ② | Logistic regression → Numerical methods (Local Search Optimization)          |
|   | ③ | SVM → Math programming (Convex Programming)                                  |
|   | ④ | Mixture models → Expectation-Maximization (EM)<br>Majorize-Minimization (MM) |



quasi-Newton method does not require to calculate Hessian matrix at every point (unlike Newton method), so will be faster and need less memory to store

linear models  $\Rightarrow$  linear parameterized models (LPMs)

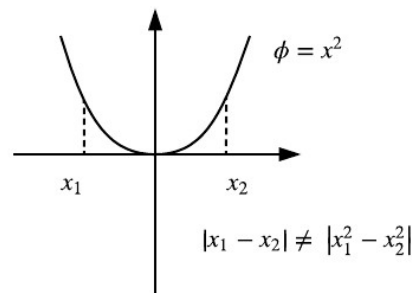
$$h(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} \Rightarrow h(\mathbf{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x})$$

how many bases?

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}')$$

Instead of using function, use inner product and only care about distance.

But nonlinear does not preserve distance



It is defining the Kernel function

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}^T(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}')$$

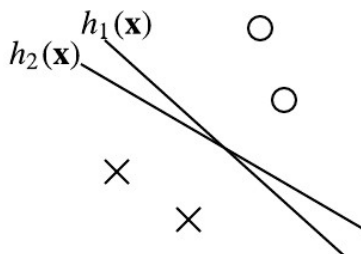
not defining the function  $\boldsymbol{\phi}$ , so the distance is between points of function values, not function itself. Kernel function is used to change the distance measure.

★ Hard-SVM v.s. Soft-SVM

separable      non-separable (relax and max margin)

Hard-SVM is the learning rule in which we return an ERM hyperplane that separates the training set with the largest possible margin, and assume that the training set is linearly separable (which is a rather strong assumption).

Soft-SVM can be viewed as a relaxation of the hard-SVM rule that can be applied even if the training set is not linearly separable. For example, it can allow the hard constraints in hard-SVM to be violated for some of the examples in the training set, by introducing nonnegative slack variables.



For linear separable data, when use perceptron algorithm (which is online), the final result can converge to some line, but not necessarily the optimal line.

★ Margin

Margin is the distance from training data points to the boundary for classification.

Functional margin of  $h(\mathbf{x}) = \boldsymbol{\beta}^T \mathbf{x}$

$$\min_{(\mathbf{x}^i, y^i) \in \mathcal{D}} \{y^i \boldsymbol{\beta}^T \mathbf{x}^i\}$$

UML Chapters 2, 3, 9, 10, 15.

BML Chapters, 11, 17, 20.