Homework #0
RELEASE DATE: 09/06/2010
DUE DATE: NONE

1 Probability and Statistics

(1) (combinatorics)
Let $C(N,K) = 1$ for $K = 0$ or $K = N$, and $C(N,K) = C(N-1,K) + C(N-1,K-1)$ for $N \geq 1$. Prove that $C(N,K) = \frac{N!}{K!(N-K)!}$ for $N \geq 1$ and $0 \leq K \leq N$.

(2) (counting)
What is the probability of getting exactly 6 heads when flipping 10 fair coins?

What is the probability of getting a full house (XXXYY) when randomly drawing 5 cards out of a deck of 52 cards?

(3) (conditional probability)
If your friend flipped a fair coin three times, and tell you that one of the tosses resulted in head, what is the probability that all three tosses resulted in heads?

(4) (Bayes theorem)
A program selects a random integer X like this: a random bit is first generated uniformly. If the bit is 0, X is drawn uniformly from $\{0, 1, \ldots, 7\}$; otherwise, X is drawn uniformly from $\{0, -1, -2, -3\}$. If we get an X from the program with $|X| = 1$, what is the probability that X is negative?

(5) (union/intersection)
If $P(A) = 0.3$ and $P(B) = 0.4$,
what is the maximum possible value of $P(A \cap B)$?
what is the minimum possible value of $P(A \cap B)$?
what is the maximum possible value of $P(A \cup B)$?
what is the minimum possible value of $P(A \cup B)$?

(6) (mean/variance)
Let mean $\overline{X} = \frac{1}{N}\sum_{n=1}^{N} X_n$ and variance $\sigma_X^2 = \frac{1}{N-1}\sum_{n=1}^{N}(X_n - \overline{X})^2$. Prove that

$$\sigma_X^2 = \frac{N}{N-1}\left(\frac{1}{N}\sum_{n=1}^{N} X_n^2 - \overline{X}^2\right).$$

(7) (Gaussian distribution)
If X_1 and X_2 are independent random variables, where $p(X_1)$ is Gaussian with mean 2 and variance 1, $p(X_2)$ is Gaussian with mean -3 and variance 4. Let $Z = X_1 + X_2$. Prove $p(Z)$ is Gaussian, and determine its mean and variance.

2 Linear Algebra

(1) (rank)
What is the rank of $\begin{pmatrix} 1 & 2 & 1 \\ 1 & 0 & 3 \\ 1 & 1 & 2 \end{pmatrix}$?

(2) (inverse)
What is the inverse of $\begin{pmatrix} 0 & 2 & 4 \\ 2 & 4 & 2 \\ 3 & 3 & 1 \end{pmatrix}$?

(3) (eigenvalues/eigenvectors)
What are the eigenvalues and eigenvectors of $\begin{pmatrix} 3 & 1 & 1 \\ 2 & 4 & 2 \\ -1 & -1 & 1 \end{pmatrix}$?

(4) (singular value decomposition)
For a real matrix M, let $\mathrm{M} = \mathrm{U}\Sigma\mathrm{V}^T$ be its singular value decomposition. Define $\mathrm{M}^\dagger = \mathrm{V}\Sigma^\dagger\mathrm{U}^T$, where $\Sigma^\dagger[i][j] = \frac{1}{\Sigma[i][j]}$ when $\Sigma[i][j]$ is nonzero, and 0 otherwise. Prove that $\mathrm{MM}^\dagger\mathrm{M} = \mathrm{M}$.

(5) (PD/PSD)
A symmetric real matrix A is positive definite (PD) *iff* $\mathbf{x}^T\mathrm{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$, and positive semi-definite (PSD) if ">" is changed to "$\geq$". Prove:

(a) For any real matrix Z, ZZ^T is PSD.

(b) A is PD *iff* all eigenvalues of A are strictly positive.

(6) (inner product)
Consider $\mathbf{x} \in R^d$ and some $\mathbf{u} \in R^d$ with $\|\mathbf{u}\| = 1$.
What is the maximum value of $\mathbf{u}^T\mathbf{x}$?
What is the minimum value of $\mathbf{u}^T\mathbf{x}$?
What is the minimum value of $|\mathbf{u}^T\mathbf{x}|$?

(7) (distance)
Consider two parallel hyperplanes in R^d:

$$H_1 : \mathbf{w}^T\mathbf{x} = +3,$$
$$H_2 : \mathbf{w}^T\mathbf{x} = -2,$$

where $\mathbf{w}$ is the norm vector. What is the distance between H_1 and H_2?

3 Calculus

(1) (differential and partial differential)
Let $f(x) = \ln(1 + e^{-2x})$. What is $\frac{df(x)}{dx}$? Let $g(x, y) = e^x + e^{2y} + e^{3xy^2}$. What is $\frac{\partial g(x,y)}{\partial y}$?

(2) (chain rule)
Let $f(x, y) = xy$, $x(u, v) = \cos(u + v)$, $y(u, v) = \sin(u - v)$. What is $\frac{\partial f}{\partial v}$?

(3) (integral)
What is $\int_5^{10} \frac{2}{x-3} dx$?

(4) (gradient and Hessian)
Let $E(u, v) = (ue^v - 2ve^{-u})^2$. Calculate the gradient ∇E and the Hessian $\nabla^2 E$ at $u = 1$ and $v = 1$.

(5) (optimization)
For some given $A > 0, B > 0$, solve

$$\min_\alpha Ae^\alpha + Be^{-2\alpha}.$$

(6) (vector calculus) Let $\mathbf{w}$ be a vector in R^d and $E(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathrm{A}\mathbf{w} + \mathbf{b}^T\mathbf{w}$ for some matrix A and vector $\mathbf{b}$. Prove that the gradient $\nabla E(\mathbf{w}) = \mathrm{A}\mathbf{w} + \mathbf{b}$ and the Hessian $\nabla^2 E(\mathbf{w}) = \mathrm{A}$.

(7) (quadratic programming) Following the previous question, if A is positive definite (PD), prove that the solution of $\mathrm{argmin}_{\mathbf{w}} E(\mathbf{w})$ is $-\mathrm{A}^{-1}\mathbf{b}$.

Homework #1

TA in charge: Chen-Wei Hung

RELEASE DATE: 09/13/2010

DUE DATE: 09/27/2010, 4:00 pm IN CLASS

TA SESSION: TBA (please check the website)

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

1.1 Learning Exercises

In the following exercises, you need to make your arguments *convincing* to get the points.

(1) (10%) Do Exercise 1.1 of LFD.

(2) (10%) Do Exercise 1.5 of LFD.

1.2 Perceptron Learning Algorithm

(1) (5%) Do Exercise 1.3-1 of LFD.

(2) (5%) Do Exercise 1.3-2 of LFD.

(3) (5%) Do Exercise 1.3-3 of LFD.

1.3 Experiments with Perceptron Learning Algorithm (*)

(1) (10%) Generate a data set of size 20 as directed by Exercise 1.4 of LFD, and plot the examples $\{(\mathbf{x}_n, y_n)\}$ as well as the target function f on a plane. Be sure to mark the examples from different classes differently, and add labels to the axes of the plot.

(2) (10%) Run Perceptron Learning Algorithm (PLA) on the data set above. Report the number of updates that PLA takes before converging. Plot the examples $\{(\mathbf{x}_n, y_n)\}$, the target function f, and the final hypothesis g in the same figure. Comment on whether f is close to g.

(3) (5%) Repeat everything in (2) with another randomly generated data set of size 20. Compare your results with (2).

(4) (5%) Repeat everything in (2) with another randomly generated data set of size 100. Compare your results with (2).

(5) (5%) Repeat everything in (2) with another randomly generated data set of size 1000. Compare your results with (2).

(6) (10%) Modify your PLA such that it takes $\mathbf{x}_n \in \mathbb{R}^{10}$ instead of $\mathbb{R}^2$. Randomly generate a data set of size 1000 with $\mathbf{x}_n \in \mathbb{R}^{10}$ and feed the data set to the algorithm. How many updates does PLA take to converge?

(7) (10%) Repeat PLA on the same data set of (6) for 100 experiments. In the iterations of each experiment, pick $\mathbf{x}(t)$ randomly instead of deterministically. Let u_k be the number of updates that PLA takes to converge within the k-th experiment. Plot a histogram of u_k.

1.4 More on Perceptron Learning

Dr. Learn observes that PLA seems to always converge and wants to prove this fact. The doctor comes up with the following idea. Assume that the target function $f(\mathbf{x}) = \text{sign}(\mathbf{w}^* \cdot \mathbf{x})$. That is, $y_n = \text{sign}(\mathbf{w}^* \cdot \mathbf{x}_n)$ for all $n = 1, 2, \cdots, N$.

(1) (5%) Using the fact that $\mathbf{w}(t)$ classifies $\mathbf{x}(t)$ incorrectly, prove $\mathbf{w}(t+1) \cdot \mathbf{w}^* \geq \mathbf{w}(t) \cdot \mathbf{w}^*$.

(2) (5%) Aha, the doctor thinks. The results indicate that after each update, $\mathbf{w}(t+1)$ would be closer to $\mathbf{w}^*$ than $\mathbf{w}(t)$ is. That is, when taking $t \to \infty$, the resulting hypothesis g would be very close to f and hence can also classify all examples perfectly (i.e., converges). Is the argument of the doctor correct? Why or why not?

ML HW#1. R98922081 徐伍文

100

1.1.

(1) 1° input X → medical history, symptoms

output Y → name of the disease

$f: X \to Y$: medical diagnosis, target function is to decide which disease that patient has by patrent's medical history and symptoms.

2° input X → handwritten digit

output Y → numbers, like 0.1.2.3.....9

$f: X \to Y$: recognize the handwritten digit to a specific number.

3° input X → e-mail user's history. user's friend list
title and contents of the coming e-mail

output Y → whether the coming e-mail is spam or not.

$f: X \to Y$: decide the coming e-mail is spam or not by considering e-mail user's history and contents of the coming e-mail.

4° input X → price, temperature, day of the week

output Y → the electric load by certain price, temperature and day of the week.

$f: X \to Y$: is to pridict the electric load with the input (price, temperature, day of the week) by learning from the history that how the electric load varies with these input values.

5° input X → ultrosound images / gray_level intensity, shape, texture.... other features of the candidate tumor.

output Y → seeking the candidate tumor and decide it's a real tumor or not

$f: X \to Y$: Tumor detection. Learning from those ultrosound images with tumors and gathering some features of them, target function is to detect tumors from patrents' ultrosound images.

(2) 1° Learning Approach. Because we don't know much about the medical test, we have to know how this medical test will affect people in different ages. Therefore, learning approach is required to find out which range of age that this medical test is suited.

2° Design Approach. We know that we can decide a number whether it's a prime or not by examining if it has any other factor except 1 under it's squre root. If the number doesn't have any other factor excep 1, the number will be classified into "Primes".

3° Learning Approach. There are many kinds of frauds in credical charges and we don't know the certain rule. Therefore, we have to learn from these frauds' history to find rules to detect potential fraud for future.

4° Design Approach. Without considering any other factor, like air resistance, the falling object is mainly affected by gravity and we know the formula: $H = \vec{V}_0 t + \frac{1}{2}gt^2$. Namely, if we know the height that this falling object fell from and it's initial velocity, we can estimate the time it takes to hit the ground. Therefore, this problem can be solved in Design Approach with considering some other factors.

5° Design Approach. We know that the road with larger traffic volume should have longer period for traffic lights. And the traffic volume also varies with different times, in other words, traffic volume becomes larger in rush hour. We know about which road with larger traffic volume and when the rush hour is before we decide the cycle for traffic lights. The problem will become easier in Design Approach.

1.2.

(1) $\vec{x}(t)$ is misclassified by $\vec{w}(t)$, i.e. $y(t) \neq \text{sign}(\vec{w}(t)\cdot\vec{x}(t))$

1° $y(t)$ = positive. $\Rightarrow y(t) \neq \text{sign}(\vec{w}(t)\cdot\vec{x}(t)) \Rightarrow \vec{w}(t)\cdot\vec{x}(t) < 0 \Rightarrow y(t)(\vec{w}(t)\cdot\vec{x}(t)) < 0$

2° $y(t)$ = negative $\Rightarrow y(t) \neq \text{sign}(\vec{w}(t)\cdot\vec{x}(t)) \Rightarrow \vec{w}(t)\cdot\vec{x}(t) > 0 \Rightarrow y(t)(\vec{w}(t)\cdot\vec{x}(t)) < 0$

Because $\vec{w}(t)\cdot\vec{x}(t)$ has an opposite sign to $y(t)$, $y(t)\cdot(\vec{w}(t)\cdot\vec{x}(t)) < 0$

(2) $\vec{w}(t+1) = \vec{w}(t) + y(t)\vec{x}(t)$

$$\Rightarrow y(t)(\vec{w}(t+1)\cdot\vec{x}(t)) = y(t)[(\vec{w}(t) + y(t)\vec{x}(t))\cdot\vec{x}(t)]$$

$$= y(t)[\vec{w}(t)\cdot\vec{x}(t) + y(t)\|\vec{x}(t)\|^2]$$

$$= y(t)(\vec{w}(t)\cdot\vec{x}(t)) + \underbrace{(y(t)\|\vec{x}(t)\|)^2}_{\text{positive } (\|\vec{x}(t)\| \geq 1)} > y(t)(\vec{w}(t)\cdot\vec{x}(t))$$

(3) $\vec{w}(t)$ classifies $\vec{x}(t)$ correctly, i.e. $y(t) = \text{sign}(\vec{w}(t)\cdot\vec{x}(t)) \Leftrightarrow y(t)(\vec{w}(t)\cdot\vec{x}(t)) > 0$

So if $\vec{x}(t)$ is misclassified by $\vec{w}(t)$ and $y(t)(\vec{w}(t)\cdot\vec{x}(t)) < 0$

As we know from (2) that the updated weight vector $\vec{w}(t+1)$ let the value of

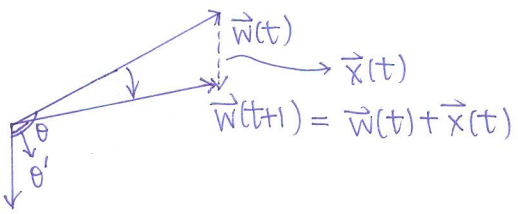
$y(t)(\vec{w}(t+1)\cdot\vec{x}(t))$ be greater than the value of $y(t)(\vec{w}(t)\cdot\vec{x}(t))$, that is to say, the updated weight vector lead the value from "negative" in a direction to "positive" to achieve the result of "$y(t)(\vec{w}(t)\cdot\vec{x}(t)) > 0$".

In a view of angle between $\vec{x}(t)$ and $\vec{w}(t)$ and take $\vec{x}_n \in \mathbb{R}^2$ for example:

$\vec{x}(t) = [\vec{x}_n \; ; \; 1] \in \mathbb{R}^3$ and $\vec{x}(t)$ is misclassified by $\vec{w}(t)$

1° $y(t)$ = positive $\Rightarrow$ $\vec{x}(t)\cdot\vec{w}(t) < 0$ $\Rightarrow$ $\theta > 90°$, θ = angle between $\vec{x}(t)$ and $\vec{w}(t)$

$$\vec{w}(t+1) = \vec{w}(t) + y(t)\vec{x}(t)$$
$$= \vec{w}(t) + \vec{x}(t)$$

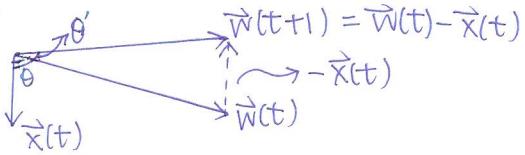


$\Rightarrow$ θ becomes smaller which is in a direction $\vec{x}(t)$ of letting $\theta < 90°$ in order to achieve the result of $\vec{x}(t)\cdot\vec{w}(t) > 0$

i.e. $y(t) = \text{sign}(\vec{x}(t)\cdot\vec{w}(t))$

2° $y(t)$ = negative $\Rightarrow$ $\vec{x}(t)\cdot\vec{w}(t) > 0$ $\Rightarrow$ $\theta < 90°$, θ = angle between $\vec{x}(t)$ and $\vec{w}(t)$

$$\vec{w}(t+1) = \vec{w}(t) + y(t)\vec{x}(t)$$
$$= \vec{w}(t) - \vec{x}(t)$$



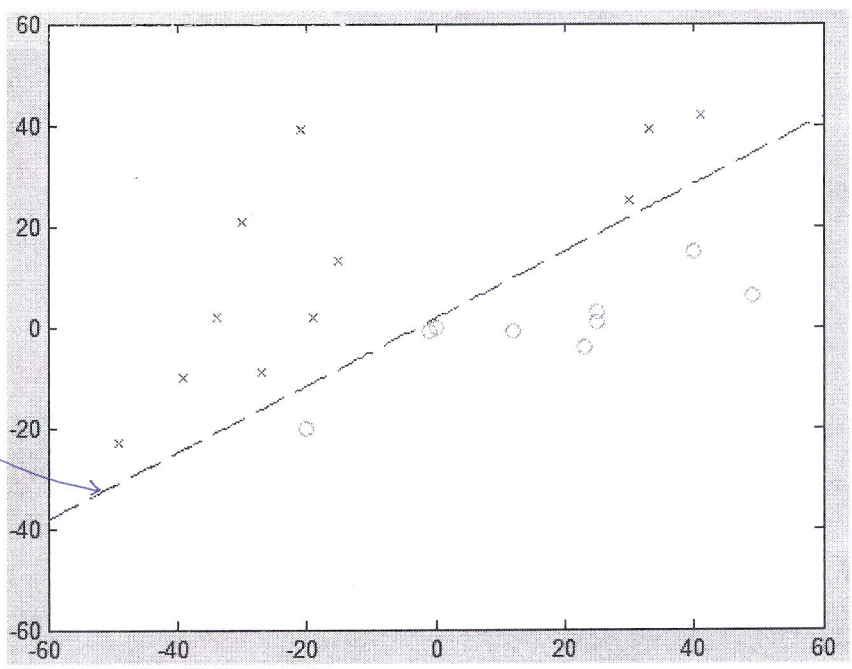
$\Rightarrow$ θ becomes bigger which is in a direction of letting $\theta > 90°$ in order to achieve the result of $\vec{x}(t)\cdot\vec{w}(t) < 0$, i.e. $y(t) = \text{sign}(\vec{x}(t)\cdot\vec{w}(t))$

Note: In I.2., we ignore the situation of sign(s), $s = 0$

1.3.

(1) $\vec{x}_n = \{(12,-1),(23,-4),(25,1),(-19,2),(-1,-1),(-30,21),(49,6)$
$(25,3),(-27,-9),(-15,13),(-49,-23),(41,42),(-39,-10),$
$(-20,-20),(30,25),(40,15),(33,39),(-34,2),(0,0)(-21,39)\}$

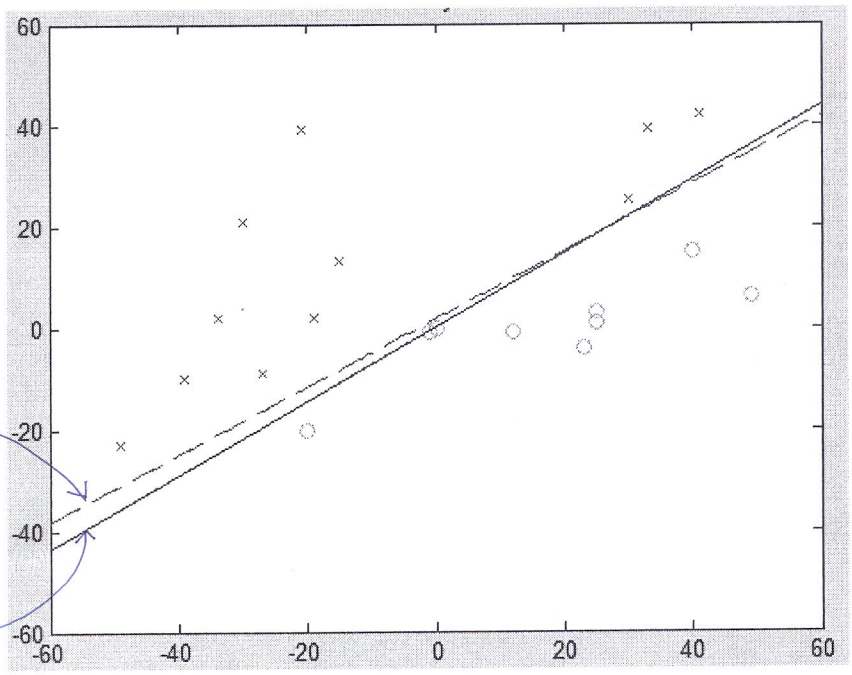
target function
$2x-3y+5=0$



(2) takes 13 times before converging.

13

g: 59*x + -81*y + 9 =0



2x-3y+5=0

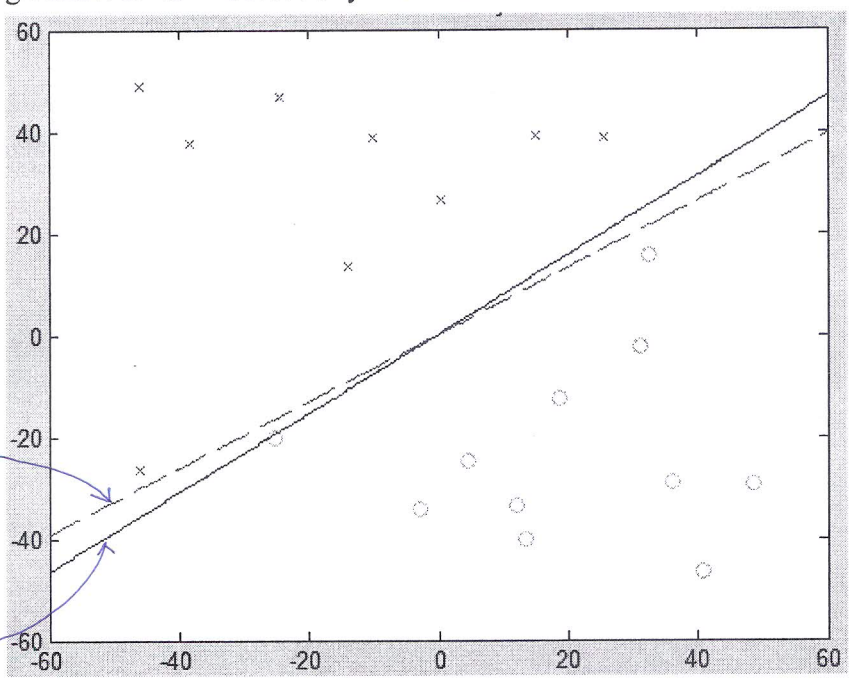
59x-81y+9=0

(3) takes 17 times before converging with data set of size 20.

17

f : 14.5072*x + -22.103*y + 1.7861 =0

g : 62.2789*x + -80.0797*y + 13 =0



f

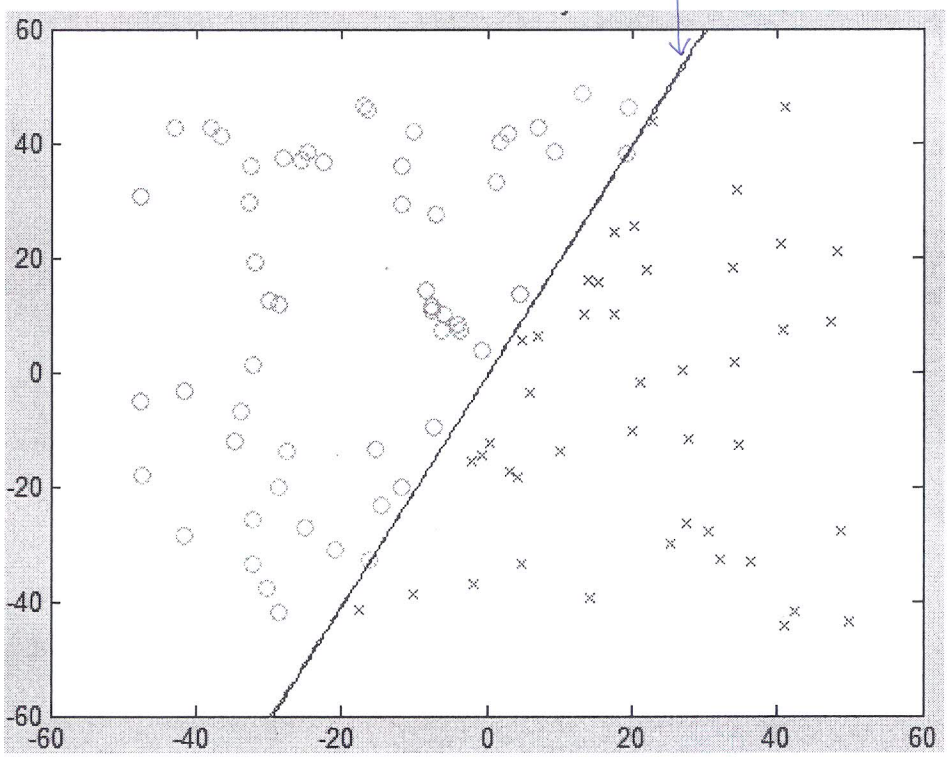
g

(4) It takes 220 times before converging with data set of size 100.
g is very close to f and they seem to be merged together in the graph.

220

f : -45.1125*x + 22.5928*y + 20.1328 =0

g : -194.8122*x + 96.4597*y + 76=0

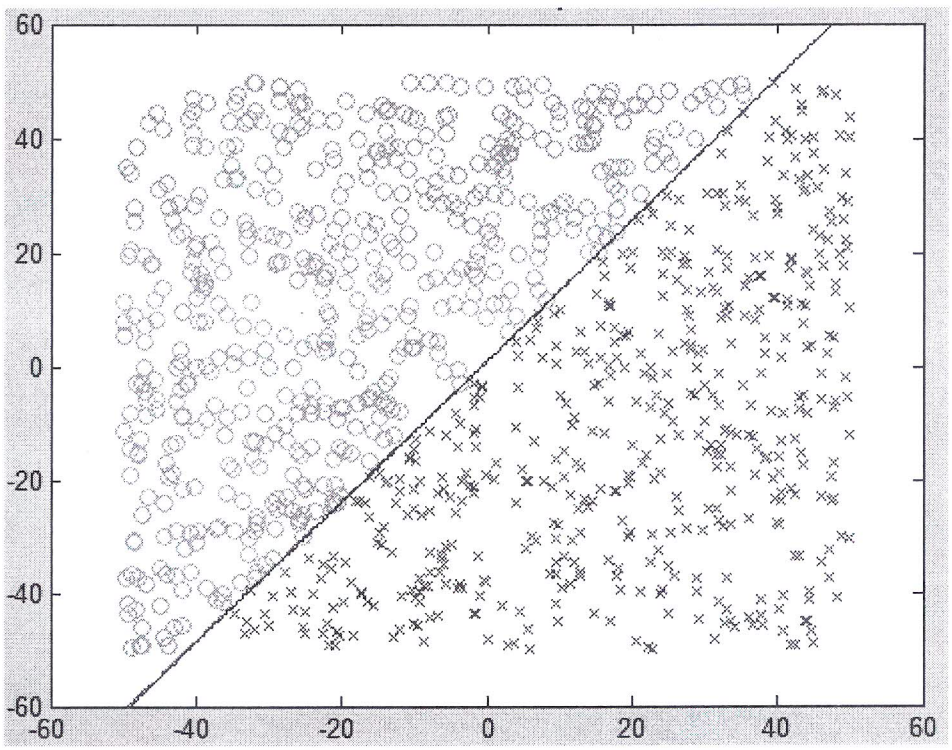


(5) It takes 330 times before converging with data set of size 1000.

330

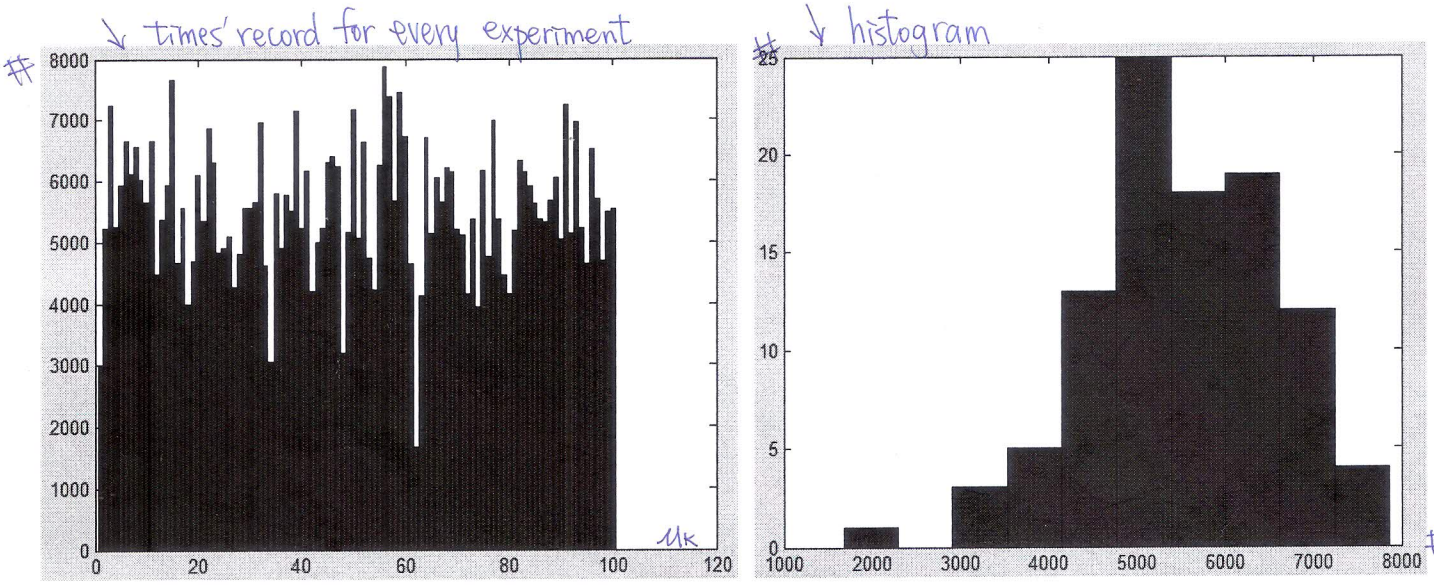
f : -31.1035*x + 25.1475*y + -27.842 =0

g : -360.8783*x + 290.5677*y + -266 =0



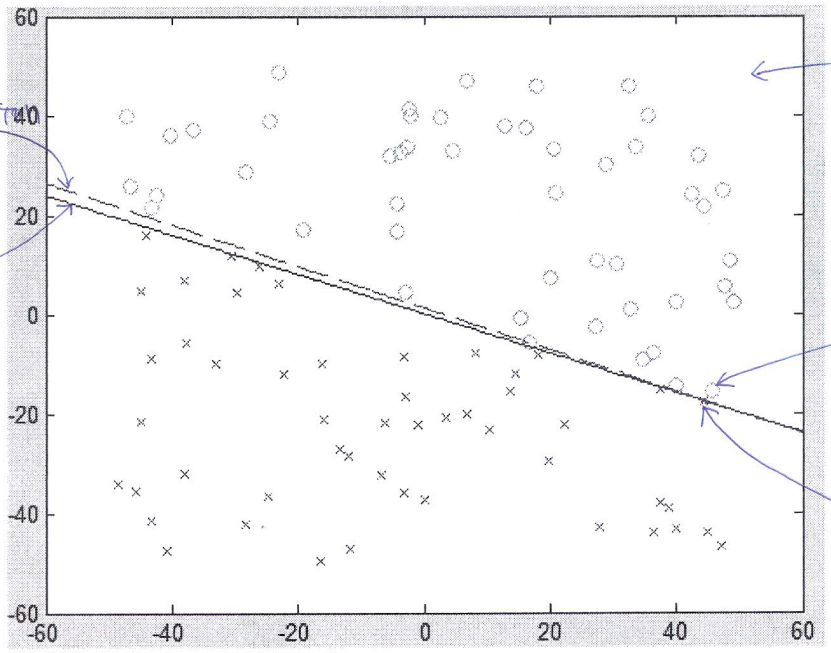
(6) $\vec{x}_n \in \mathbb{R}^{10}$, and size of the data set is 1000
It takes 9340 times before converge if we pick $\vec{x}(t)$ deterministically.

(7) It take 5537 times in average for 100 experments in the condition of picking $\vec{x}(t)$ randomly instead of deterministically. We can find out that it seems take less times if we pick $\vec{x}(t)$ randomly than deterministically. There is even no any time that it take more times to converge than 9340 in this case.



The reason for this result may be because if we choose $\vec{x}(t)$ in a certain order, it becomes easier to fall in many interations between $\vec{x}_i$ and $\vec{x}_{i+d}$. For example, $\vec{w}(t)$ didn't misclassify $\vec{x}$ until $\vec{x}_{i+d}$, and the new update $\vec{w}(t+1)$ classified $\vec{x}_{i+d}$ correctly. However, same time, there were some points $\vec{x}_j$ $(i \le j < i+d)$ which were misclassified by this new update $\vec{w}(t+1)$ so that we have to update $\vec{w}$ starting all over again from $\vec{x}_m$ $(m = \min(j))$. And even if we modified our $\vec{w}(t+1)$ to $\vec{w}(t+n)$ to make it correct for $\vec{x}_m$ in classification. It may happen that this $\vec{w}(t+n)$ misclassified $\vec{x}_{i+d}$ and the new update $\vec{w}(t+n+1)$ for $\vec{x}_{i+d}$ misclassified some point $\vec{x}_k$ $(k < i+d)$ so that we have to go back from $\vec{x}_k$ to update the new $\vec{w}$ again. Therefore, it takes lots of iterations between an interval to make the new update $\vec{w}$ classify these points correctly. Of course, this situation may also happen if we choose to pick $\vec{x}(t)$ randomly. However, if we pick $\vec{x}(t)$ randomly, it may have chances to access other points which were not in the interval and it may reduce some iterations. The situation above is more likely to happen if we pick $\vec{x}(t)$ in a certain order. In many times experiments, the number of times before converging is case-by-case. Sometimes it takes more times to pick $\vec{x}(t)$ randomly than deterministically because the random choosing $\vec{x}(t)$ doesn't make $\vec{w}$ better. In my observation, due to the situation I've mentioned abuoe is more likely to happen in the deterministical case, it usually takes more times than if we choose $\vec{x}(t)$ randomly.

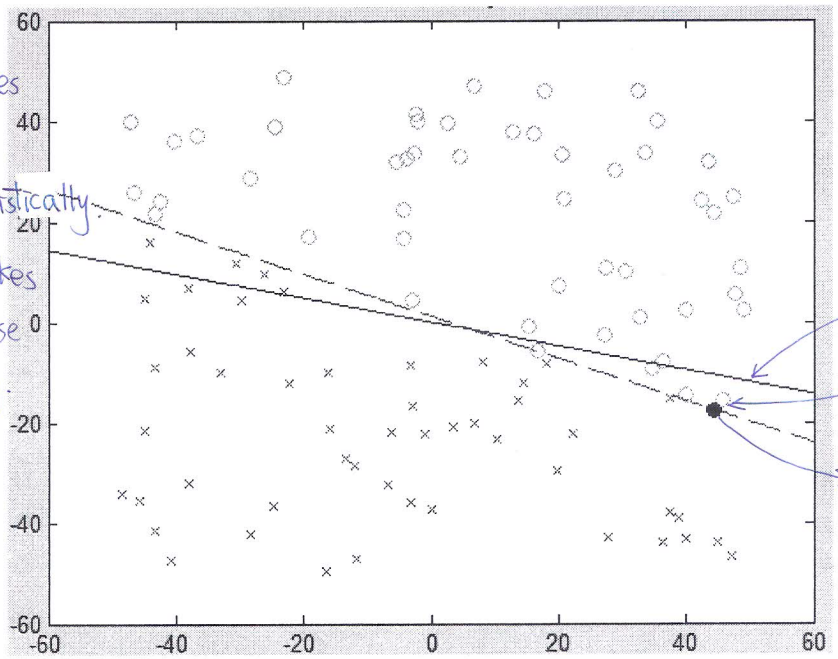
The following is a typical case that if we pick $\vec{x}(t)$ in a certain order, it may be likely to fall in many iterations between an interval.

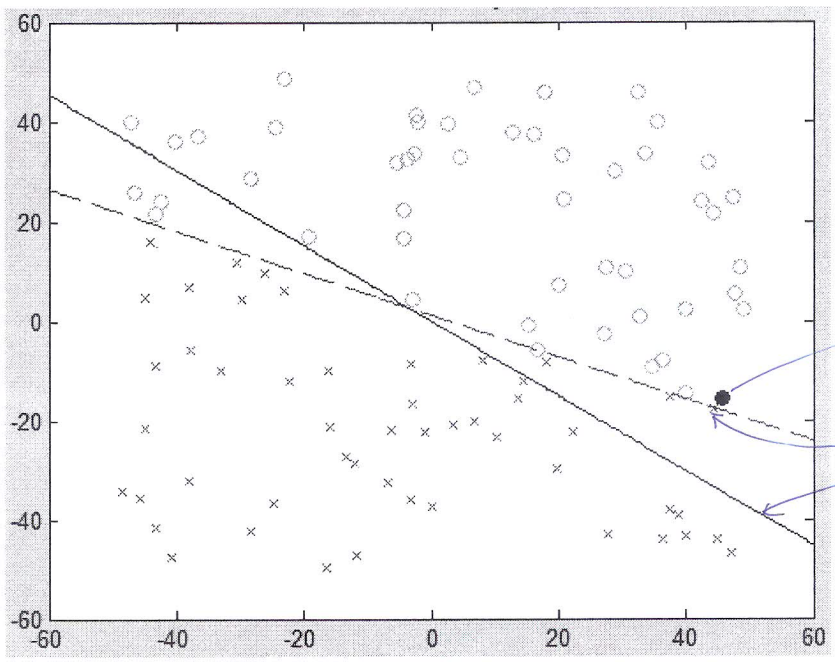


In this case, it take a lot of iterations between No.4 point and No. 19 point.

And it takes 504 times before converging if we pick $\vec{x}(t)$ deterministically.

However, it only takes 20 times if we choose to pick $\vec{x}(t)$ randomly.





1.4

(1) $\vec{w}(t+1) = \vec{w}(t) + y(t)\vec{x}(t)$

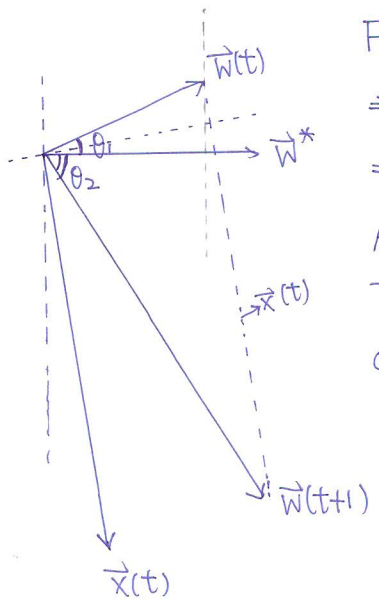
$\Rightarrow \vec{w}(t+1)\cdot\vec{w}^* = (\vec{w}(t)+y(t)\vec{x}(t))\cdot\vec{w}^*$

$= \vec{w}(t)\cdot\vec{w}^* + \underbrace{y(t)(\vec{x}(t)\cdot\vec{w}^*)}_{\geq 0}$ $\because \vec{w}^*$ classifies all $\vec{x}_n$ correctly

$\therefore y(t)(\vec{x}(t)\cdot\vec{w}^*) \geq 0$

$\geq \vec{w}(t)\cdot\vec{w}^*$

(2) Half agree. I think $\vec{w}$ will converge and finally classify all points $\vec{x}_n$ correctly. But it's not neccessary to be very close to f unless input data size is very large. And sometimes $\vec{w}$ will modify too much for some certain points so that it wont make $\vec{w}(t+1)$ get closer to $\vec{w}^*$ than $\vec{w}(t)$ after "each update".

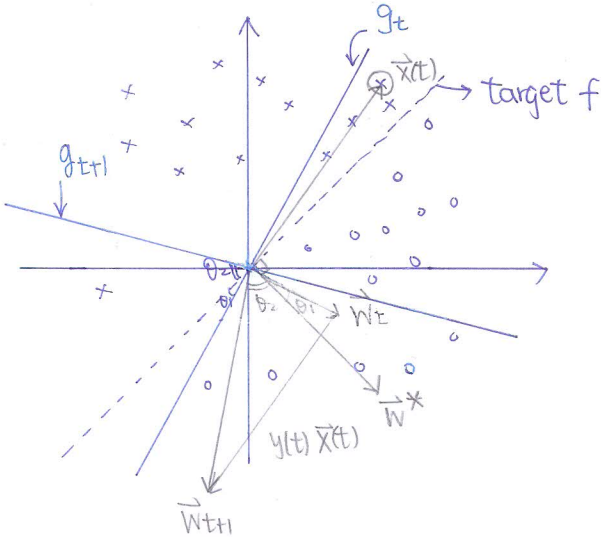


For example, $y(t)$: positive and $\vec{w}(t)$ misclassified $\vec{x}(t)$

$\Rightarrow \vec{w}(t)\cdot\vec{x}(t) < 0$ and $\vec{w}^*(t)\cdot\vec{x}(t) > 0$,

$\Rightarrow \vec{w}(t+1) = \vec{w}(t) + \vec{x}(t)$

As we can see this case from figure, the new update $\vec{w}(t+1)$ make $\vec{w}(t+1)\cdot\vec{x}(t) > 0$ (classified $\vec{x}(t)$ correctly). However, it didn't get closer to $\vec{w}^*$. ($\theta_2 > \theta_1$)



This case can be shown in 2D-graph without considering the constant part. $\Rightarrow$ We modified $\vec{w}(t)$ to $\vec{w}(t+1)$ for $\vec{x}(t)$, however, even though the new updat $\vec{w}(t+1)$ classified $\vec{x}(t)$ correctly, it didn't get closer to $\vec{w}^*$. Therefore, the doctor say " after each update, $\vec{w}(t+1)$ would be closer to $\vec{w}^*$ than $\vec{w}(t)$ is" doesn't happen all the time.

Note: The result $\vec{w}(t+1)\cdot\vec{w}^* \geq \vec{w}(t)\cdot\vec{w}^*$ still holds in this case.

this formula not guarantee $w(t+1)$ is closer to w^* than $w(t)$.
does

Sidework #1
RELEASE DATE: 10/01/2010

DUE DATE: NONE

1.1 Hoeffding's Inequality

The proof that you will write below contains all the essential steps, but are not as rigorously written as the usual math texts.

(1) (Markov's Inequality) Prove that for any non-negative random variable ℓ and any positive constant a,

$$P(\ell \geq a) \leq \frac{E(\ell)}{a}.$$

(2) (Moment-Generating Function—Laplace Transform) Prove that for any finite random variable λ, any positive constant α, and any positive parameter s,

$$P(\lambda \geq \alpha) \leq e^{-s\alpha} E(e^{s\lambda}) ;$$
$$P(\lambda \leq \alpha) \leq e^{s\alpha} E(e^{-s\lambda}) .$$

(3) (Independence Decomposition) Let $z_1, z_2, \cdots, z_N$ be i.i.d. random variables and let $z = \frac{1}{N}\sum_{n=1}^{N} z_n$. For any positive constant α and any positive parameter s, prove that

$$P(z \geq \alpha) \leq (e^{-s\alpha} E(e^{sz_1}))^N ;$$
$$P(z \leq \alpha) \leq (e^{s\alpha} E(e^{-sz_1}))^N .$$

(4) (Bound Tightening) Let z_1 be a binary random variable with $P(z_1 = 0) = 1 - \theta$ and $P(z_1 = 1) = \theta$. Let

$$F(s) = e^{-s\alpha} E(e^{sz_1})$$

For any given α with $\theta < \alpha < 1$, prove that $F(s)$ is minimized on

$$s^* = \ln \frac{\alpha \cdot (1-\theta)}{(1-\alpha) \cdot \theta},$$

where s^* is positive.

(5) (Chernoff Bound) Use the fact that

$$P(z \geq \alpha) \leq (e^{-s^*\alpha} E(e^{s^* z_1}))^N$$

to prove

$$P(z \geq \alpha) \leq e^{-ND(\alpha||\theta)}$$

for $\theta < \alpha < 1$. Here $D(\alpha||\theta) = \alpha \log \frac{\alpha}{\theta} + (1-\alpha) \log \frac{1-\alpha}{1-\theta}$ is the KL-divergence between α and θ.

(6) (One-sided Hoeffding Inequality) Let $\alpha = \theta + \epsilon$ with $\epsilon > 0$, prove that

$$P(z - \theta \geq \epsilon) \leq e^{-2N\epsilon^2}$$

by showing that $D(\theta + \epsilon||\theta) \geq 2\epsilon^2$ for all $\epsilon > 0$.

(7) (Two-sided Hoeffding Inequality) Prove that

$$P(|z - \theta| \geq \epsilon) \leq 2e^{-2N\epsilon^2}.$$

Homework #2

TA in charge: Yu-Xun Ruan

RELEASE DATE: 09/27/2010

DUE DATE: 10/11/2010, 4:00 pm IN CLASS

TA SESSION: 10/07/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

2.1 Simplified No-Free-Lunch Theorem

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{x}_{N+1}, \ldots, \mathbf{x}_{N+M}\}$ and $\mathcal{Y} = \{-1, +1\}$ (binary classification). Here the set of training examples is $\mathcal{D} = \left\{(\mathbf{x}_n, y_n)\right\}_{n=1}^{N}$, where $y_n \in \mathcal{Y}$, and the set of test inputs is $\left\{\mathbf{x}_{N+m}\right\}_{m=1}^{M}$. The *Off-Training-Set error* (*OTS*) with respect to an underlying target f and a hypothesis g is

$$E_{OTS}(g, f) = \frac{1}{M} \sum_{m=1}^{M} [\![g(\mathbf{x}_{N+m}) \neq f(\mathbf{x}_{N+m})]\!] .$$

(1) (5%) Say $f(\mathbf{x}) = 1$ for all $\mathbf{x}$ and $g(\mathbf{x}) = \begin{cases} 1, & \text{for } \mathbf{x} = \mathbf{x}_k \text{ and } k \text{ is even and } 1 \leq k \leq M+N \\ -1, & \text{otherwise} \end{cases}$.
What is $E_{OTS}(g, f)$?

(2) (5%) We say that a target function f can "generate" $\mathcal{D}$ in a noiseless setting if $f(\mathbf{x}_n) = y_n$ for all $(\mathbf{x}_n, y_n) \in \mathcal{D}$. For all possible $f: \mathcal{X} \to \mathcal{Y}$, how many of them can generate $\mathcal{D}$ in a noiseless setting?

(3) (5%) For a fixed g, if all those f in (2) are equally likely in probability, what is the expected off-training-set error $\mathbb{E}_f\{E_{OTS}(g, f)\}$?

(4) (5%) A determistic algorithm A is defined as a procedure that takes $\mathcal{D}$ as an input, and outputs a hypothesis g. Argue that for any two deterministic algorithms A_1 and A_2,

$$\mathbb{E}_f\Big\{E_{OTS}(A_1(\mathcal{D}), f)\Big\} = \mathbb{E}_f\Big\{E_{OTS}(A_2(\mathcal{D}), f)\Big\}.$$

You have now proved that "in a noiseless setting (f generates $\mathcal{D}$), for a fixed $\mathcal{D}$, if all possible f are equally likely, any two deterministic algorithms are the same in terms of $\mathbb{E}_f\{E_{OTS}\}$."

2.2 Marbles and Coins

(1) (5%) Do Exercise 1.9 of LFD.

(2) (5%) Do Exercise 1.10 of LFD.

(3) (5%) Do Exercise 1.11-1 of LFD.

(4) (5%) Do Exercise 1.11-2 of LFD.

(5) (5%) Do Exercise 1.11-3 of LFD.

2.3 Learning Games

(1) (5%) Do Exercise 1.12-1 of LFD.

(2) (5%) Do Exercise 1.12-2 of LFD.

(3) (5%) Do Exercise 1.12-3 of LFD.

(4) (5%) Do Exercise 1.12-4 of LFD.

2.4 Probably Approximately Correct

Read the derivation that links Equation (1.6) to Equation (2.1) on LFD Page **2**-2. In particular, let

$$\epsilon(M, N, \delta) = \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}.$$

(1) (5%) Take $\delta = 0.03$ and $M = 1$, how many examples do we need to make $\epsilon(M, N, \delta) \leq 0.05$?

(2) (5%) Take $\delta = 0.03$ and $M = 100$, how many examples do we need to make $\epsilon(M, N, \delta) \leq 0.05$?

(3) (5%) Take $\delta = 0.03$ and $M = 10000$, how many examples do we need to make $\epsilon(M, N, \delta) \leq 0.05$?

The title of this problem, *Probably Approximately Correct*, states what we can interpret from (2.1) if we have enough training examples. "Probably" means the statement is true with a high probability ($\geq 1 - \delta$). "Approximately" means that every $E_{\text{out}}(g)$ is close to $E_{\text{in}}(g)$ (within ϵ). "Correct" means that we can guarantee $E_{\text{out}}(g)$ to be small (by getting some decision function g with small $E_{\text{in}}(g)$).

2.5 Adaptive Perceptron Learning (*)

We know that the perceptron learning rule works like this: In each iteration, pick a random $(\mathbf{x}^{(t)}, y^{(t)})$ and compute $\rho^{(t)} = \mathbf{w}^{(t)} \bullet \mathbf{x}^{(t)}$. If $y^{(t)} \cdot \rho^{(t)} \leq 0$, update $\mathbf{w}$ by

$$\mathbf{w}^{(t+1)} \longleftarrow \mathbf{w}^{(t)} + y^{(t)} \cdot \mathbf{x}^{(t)} \ ;$$

One may argue that the algorithm did not take the "closeness" between $\rho^{(t)}$ and $y^{(t)}$ into consideration. Let's look at another perceptron learning algorithm: In each iteration, pick a random $(\mathbf{x}^{(t)}, y^{(t)})$ and compute $\rho^{(t)} = \mathbf{w}^{(t)} \bullet \mathbf{x}^{(t)}$. If $y^{(t)} \cdot \rho^{(t)} \leq 1$, update $\mathbf{w}$ by

$$\mathbf{w}^{(t+1)} \longleftarrow \mathbf{w}^{(t)} + \eta \cdot \left(y^{(t)} - \rho^{(t)}\right) \cdot \mathbf{x}^{(t)} \ ,$$

where η is some constant. That is, if $\rho^{(t)}$ agrees with $y^{(t)}$ a lot (their product is > 1), the algorithm does nothing. On the other hand, if $\rho^{(t)}$ is further from $y^{(t)}$, the algorithm changes $\mathbf{w}^{(t)}$ more. In this problem, you are asked to implement this algorithm and check its performance.

(1) (5%) Generate a training data set of size 100 as directed in Homework Problem 1.3. Generate a test data set of size 10000 from the same process. Run the algorithm above with $\eta = 100$ on the training data set until it converges (no more possible updates) or a maximum of 1000 updates has been reached to get g. Plot the training data set, the target function f, and the final hypothesis g on the same figure. Estimate the out-of-sample error with the test set.

(2) (5%) Use the data set in (1) and redo everything with $\eta = 1$.

(3) (5%) Use the data set in (1) and redo everything with $\eta = 0.01$.

(4) (5%) Compare the results that you get from (1) to (3).

The algorithm above is a variant of the so-called Adaline (*Adaptive Linear Neuron*) algorithm for perceptron learning.

ML HW #2 R98922081 徐位文

100

2.1.

(1) M: even $\Rightarrow E_{OTS}(g,f) = \frac{1}{2}$

M: odd $\Rightarrow$ $\begin{cases} E_{OTS}(g,f) = \frac{M+1}{2M} & \text{if } N: \text{even} \\ E_{OTS}(g,f) = \frac{M-1}{2M} & \text{if } N: \text{odd} \end{cases}$

(2) 2^M. There are M test inputs, and each input maps to either $+1$ or -1 for its output. Because $f(\vec{x}_n) = y_n$ for all $(\vec{x}_n, y_n) \in D$, and two options for each test input. Therefore, there are 2^M possible functions can generate D in a noiseless setting.

(3) $\because$ all possible functions are equally likely in probability $\Rightarrow Pr[f_i = f] = \frac{1}{2^M}$ for $i = 1 \sim 2^M$

$\therefore \mathbb{E}_f\{E_{OTS}(g,f)\} = \frac{1}{2^M}\left[\sum_{i=1}^{2^M} E_{OTS}(f_i, f)\right]$

$= \frac{1}{2^M}\left[\sum_{k=0}^{M} C_k^M \cdot \frac{k}{M}\right]$ // splits 2^M functions into $(M+1)$ clusters

C_k^M: number of functions with k error digits; $\frac{k}{M}$: E_{OTS} which has k error digits; k represents for the number of error digits

$(C_0^M + C_1^M + C_2^M + \cdots + C_M^M = 2^M)$

$= \frac{\sum_{k=0}^{M} k \cdot C_k^M}{M \cdot 2^M} = \frac{M \cdot 2^{M-1}}{M \cdot 2^M} = \frac{1}{2}$

(4) $\because$ all possible functions are equally likely in probability $\Rightarrow Pr[A_1(D) = f] = \frac{1}{2^M}$

$\Rightarrow \mathbb{E}_f\{E_{OTS}(A_1(D), f)\} = \frac{1}{2^M} \cdot 0 + \frac{C_1^M}{2^M} \cdot \frac{1}{M} + \frac{C_2^M}{2^M} \cdot \frac{2}{M} + \cdots + \frac{C_M^M}{2^M} \cdot \frac{M}{M}$

$\frac{1}{2^M}$: probability that $A_1(D) = f$; 0: $E_{OTS} = 0$; $\frac{C_1^M}{2^M}$: probability that f is one digit different from $A_1(D)$; $\frac{1}{M}$: $E_{OTS} = \frac{1}{M}$, for f is one digit different from $A_1(D)$

$= \sum_{k=0}^{M}\left[\frac{C_k^M}{2^M} \cdot \frac{k}{M}\right] = \frac{1}{2}$

$\because$ all possible functions are equally likely in probability

$\therefore$ same situation for $Pr[A_2(D) = f] = \frac{1}{2^M}$

$\Rightarrow \mathbb{E}_f\{E_{OTS}(A_2(D), f)\} = \sum_{k=0}^{M}\left[\frac{C_k^M}{2^M} \cdot \frac{k}{M}\right] = \frac{1}{2} = \mathbb{E}_f\{E_{OTS}(A_1(D), f)\}$

2.2.

(1) $P = C_1^{10}(0.9)^1(0.1)^9 = 9 \times 10^{-9}$

(2) Hoeffding Inequality $\Pr[|\nu-\mu|>\epsilon] \le 2e^{-2\epsilon^2 N}$, for any $\epsilon > 0$

$\Rightarrow \Pr[|\nu-\mu|>0.8] \le 2\cdot e^{-2(0.8)^2(10)} \doteq 5.522\times10^{-6}$

$\Rightarrow 9\times10^{-9} \le 5.522\times10^{-6}$, the inequality holds.

(3) $P = C_{10}^{10}(\frac{1}{2})^{10}(\frac{1}{2})^{0} = \frac{1}{1024}$

(4) $P' = C_{0}^{1000}(\frac{1}{1024})^{0}(\frac{1023}{1024})^{1000} = (\frac{1023}{1024})^{1000}$

$P = 1 - P' = 1 - (\frac{1023}{1024})^{1000} \doteq 0.6236$

(5) 1000 fair coins $\longrightarrow$ multiple bins

flip each 10 times $\longrightarrow$ randomly choose 10 marbles for sampling

the probability that at least one of the coins will give us 10 heads

$\rightarrow$ the probability that at least one of the bins will have 10 RED marbles from randomly choosing 10 marbles for sampling.

2.3.

(1) No. Unless data set D reflects the same distribution as in outside D, it can't guarantee that S will produce a better hypothesis than C.

(2) Yes. If $\Pr[f(\vec{x})=-1] > 0.5$, it's still possible that all the examples in D have $y_n = +1$ so that the hypothese produced from C performs better than from S with an assumption that the whole number of examples is far larger than the sampling number (at least more than 50 in this case)

(3) the probability that S will produce a better hypothesis than C

$\Leftrightarrow$ the probability that S will choose +1 function as its hypothesis

$\Leftrightarrow$ the probability that the number of $y_n=+1$ in D is more than half of D

$$\downarrow$$

$$P = \sum_{\hat{\imath}=13}^{25}\left[C_{\hat{\imath}}^{25}(0.9)^{\hat{\imath}}(0.1)^{25-\hat{\imath}} \right] \doteq 0.999999837$$

(4) 1° if $\Pr[f(\vec{x})=+1] = P > 0.5$

$\Rightarrow$ it is more likely than not that the number of $y_n=+1$ in D is more than half

$\Rightarrow$ it is more likely than not that S will choose +1 function as its hypothesis

$\Rightarrow$ it is more likely than not that S will produce a better hypothesis than C

2° if $\Pr[f(\vec{x})+1] = P < 0.5$ i.e. $\Pr[f(\vec{x})=-1] > 0.5$

$\Rightarrow$ it is more likely than not that the number of $y_n=-1$ in D is more than half.

$\Rightarrow$ it is more likely than not that S will choose -1 function as its hypothesis

$\Rightarrow$ it is more likely than not that S will produce a better hypothesis than C

3° if $\Pr[[f(\vec{x}) = +1]] = p = 0.5$

⇒ function +1 and −1 will perform the same.

⇒ hypothesis produced from S will perform the same as from C

4° from 1°. 2°. 3°, there is no such p for which it is more likely than not that C will produce a better hypothesis than S.

2.4.

$$\epsilon(M, N, \delta) = \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}} \leq 0.05$$

$$\Rightarrow \frac{1}{2N} \ln \frac{2M}{\delta} \leq (0.05)^2$$

$$\Rightarrow N \geq \frac{\ln \frac{2M}{\delta}}{2 \cdot (0.05)^2}$$

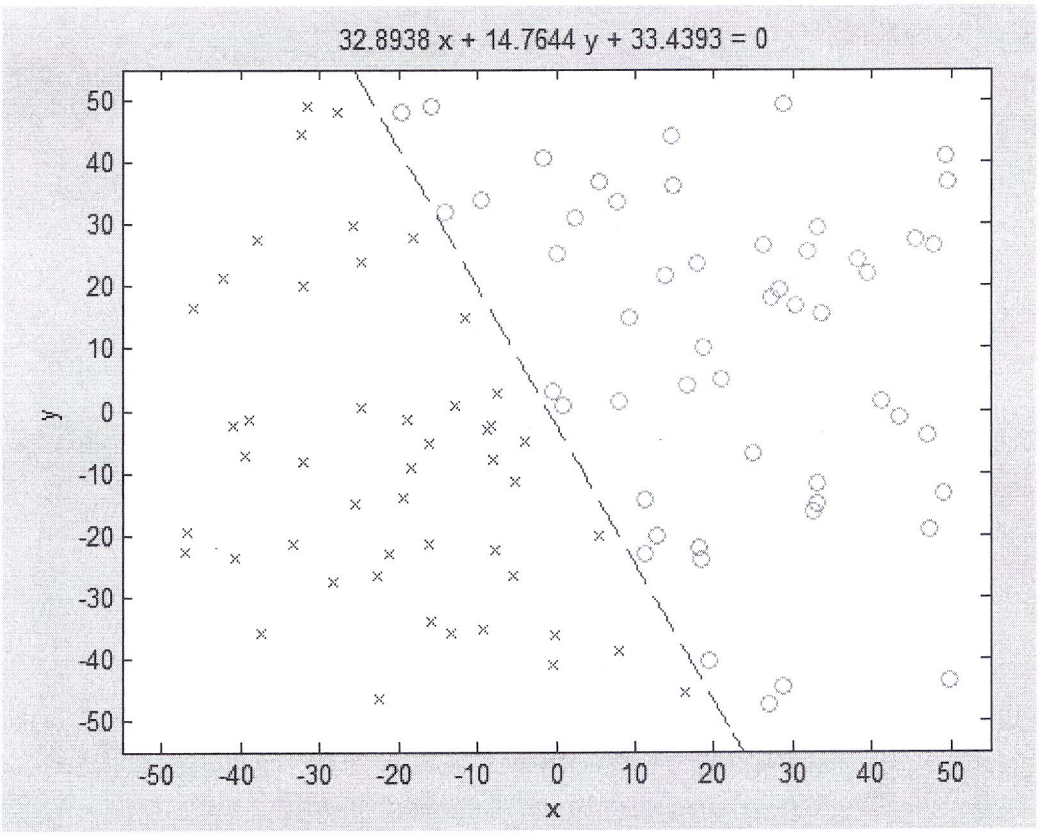
(1) $\delta = 0.03$, $M = 1$ ⇒ $N \geq 839.94$ ⇒ Need at least 840 examples

(2) $\delta = 0.03$, $M = 100$ ⇒ $N \geq 1760.97$ ⇒ Need at least 1761 examples

(3) $\delta = 0.03$, $M = 10000$ ⇒ $N \geq 2682.009$ ⇒ Need at least 2683 examples

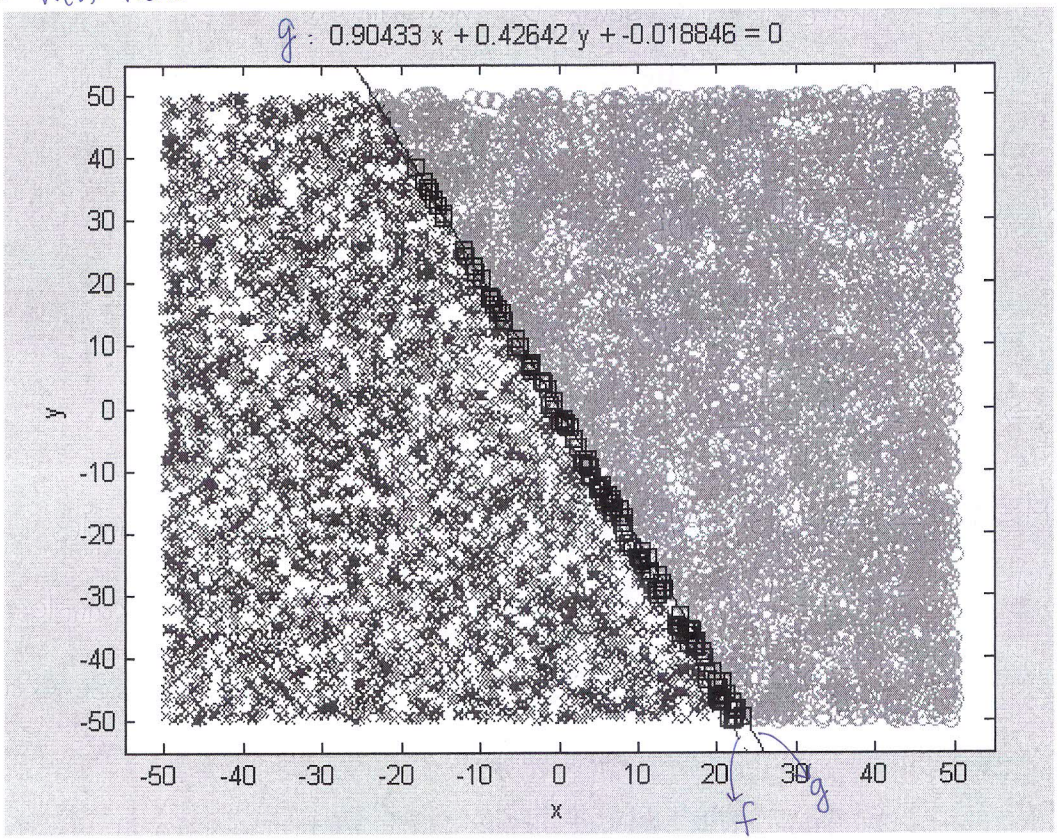
2.5

The following is a training data set of size 100 and the target function f.

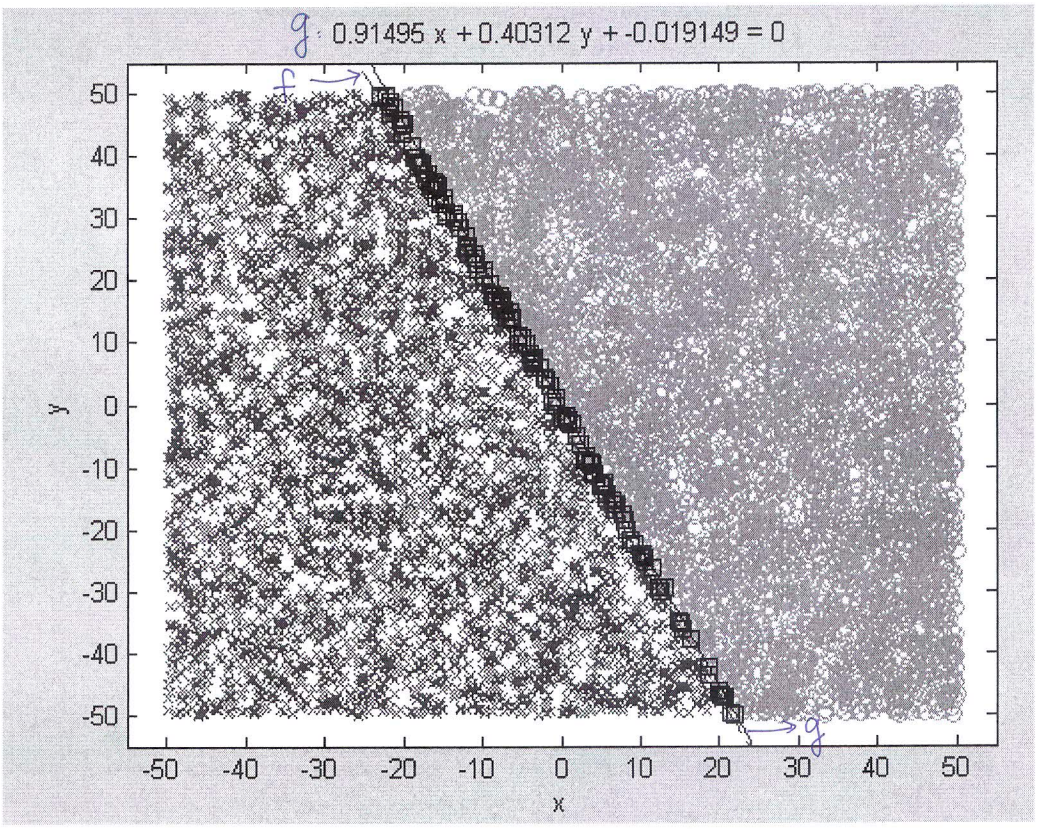


(1) With $\eta = 100$, it takes 14 times to converge and the error number in testing is 116 (squares in the figure).

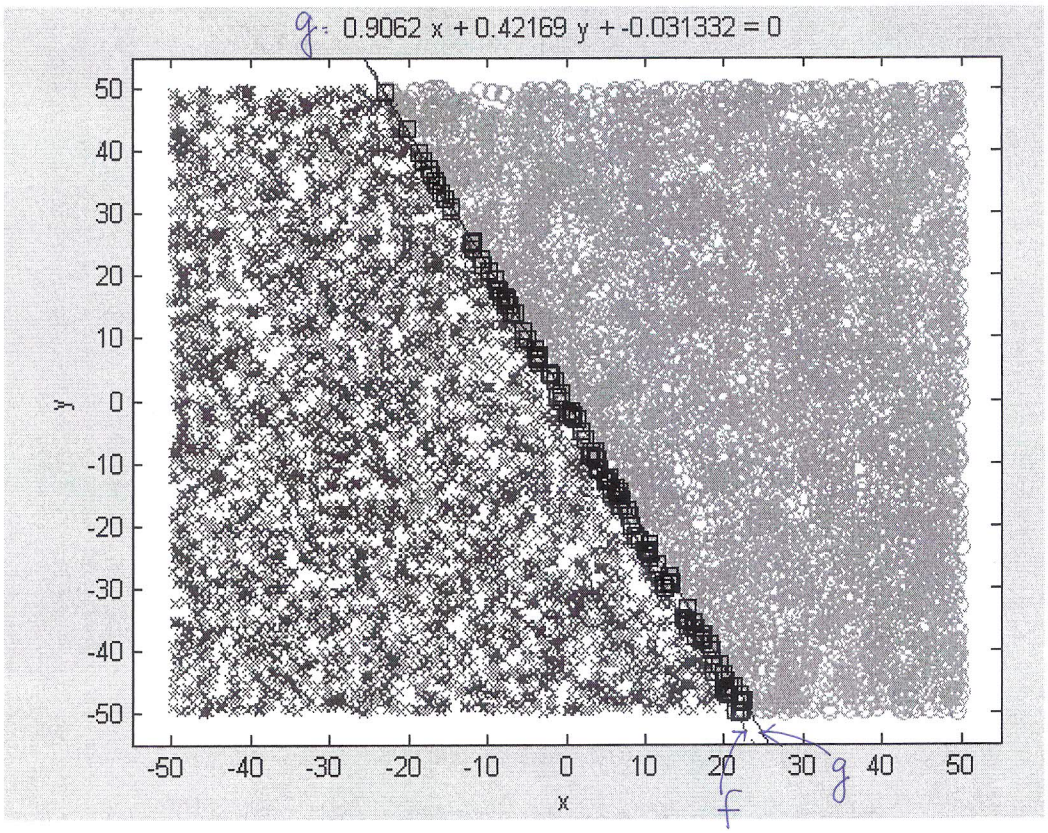
Note: The updated $\vec{w}(t)$ is very easy to overflow, therefore, I normalized $\vec{w}(t)$ when $\vec{w}(t)$ went almost infinite.



(2) With $\eta = 1$, it takes 208 times to converge and the error number in testing is 132.

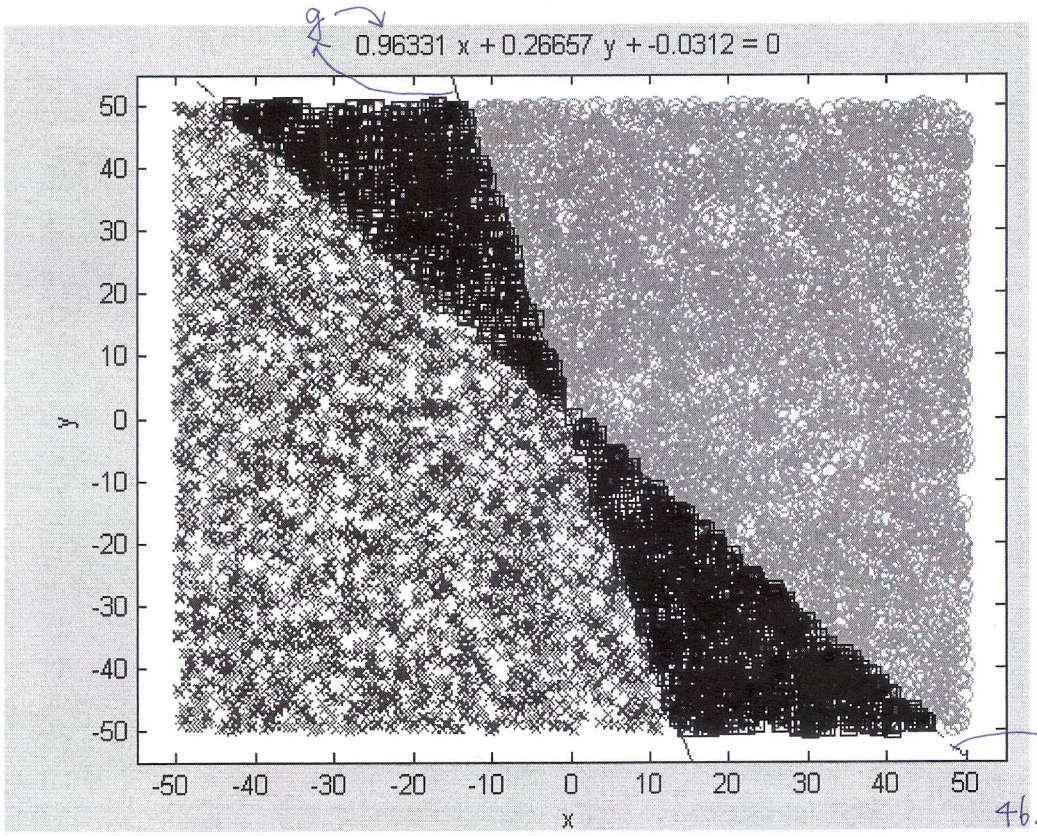


(3) With $\eta = 0.01$. It takes 25 times to converge and the error number in testing is 119.

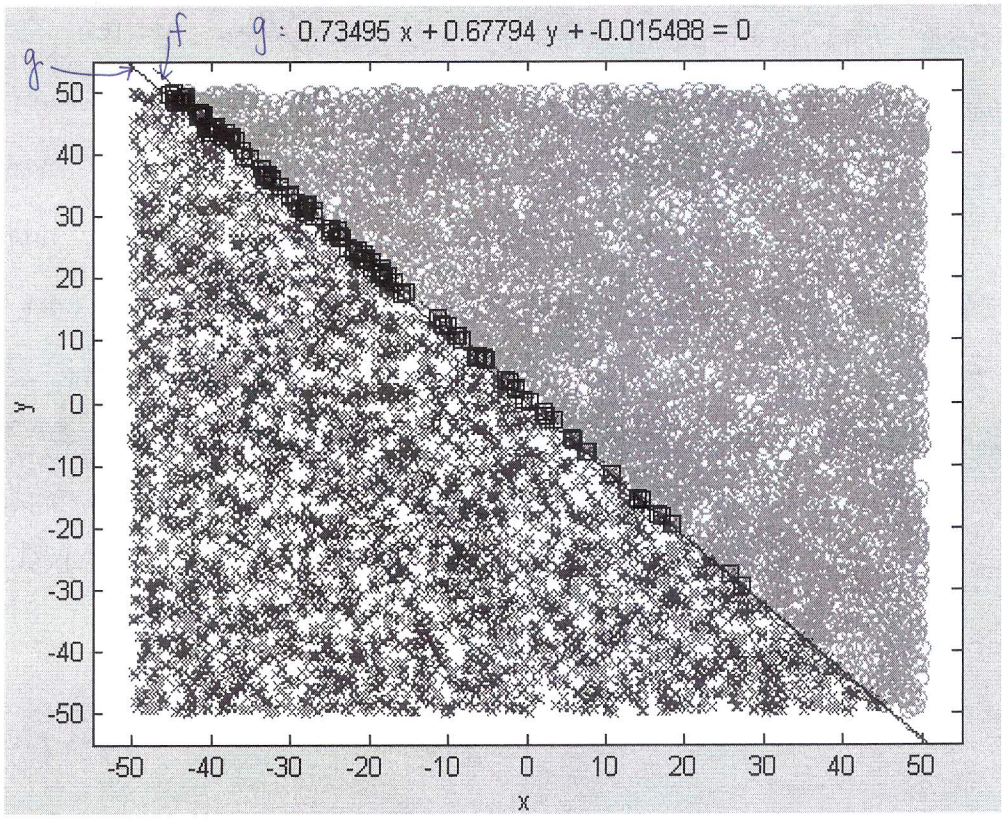


(4) In this experiment, the performance seems the best when $\eta = 100$. It takes less times to converge and its out-of-sample error is the least. However, the situation like this doesn't happen very often. Sometimes with $\eta = 100$, it takes more than 1000 times so we stop, and its out-of-sample error is very huge. The other experiment shown in next page implies that with $\eta = 1$ seems have the best performance. I think the results are case-by-case. It may also depend on the scale of example points. Sometimes, it takes more than 1000 times for $\eta = 100$ so we stop, but its hypothesis is far from the target function so that the out-of-sample error is quite a lot. And it also happens that with $\eta = 0.01$ performs the best in some experiments.

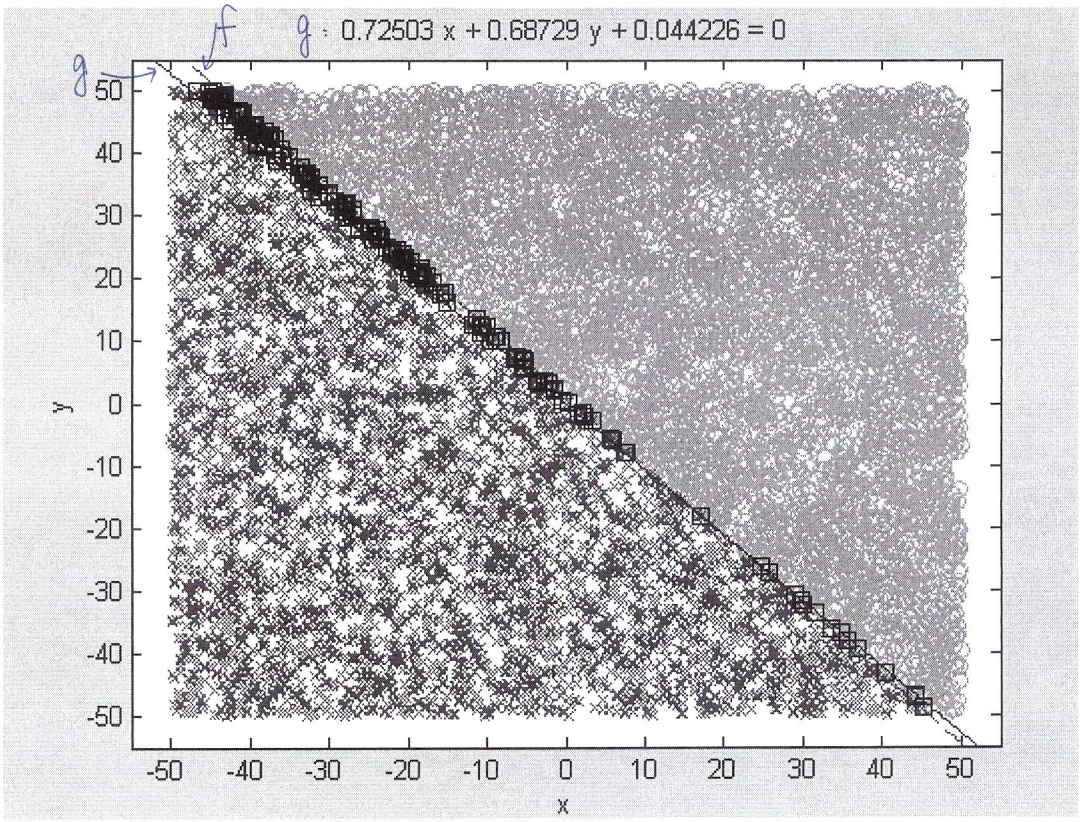
$\eta = 100$, it takes more than 1000 and with 1524 errors.



$\eta = 1$. it takes 92 times to converge and with 107 errors.



$\gamma = 0.01$. It takes 179 times to converge and with 149 errors.



Homework #3

TA in charge: Yao-Nan Chen

RELEASE DATE: 10/11/2010

DUE DATE: 10/25/2010, 4:00 pm IN CLASS

TA SESSION: 10/21/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

3.1 Growth Function and VC Dimension

By "verify" in Exercise 2.3 of LFD, we mean "mathematically verify." That is, you need to take one n that satisfy the condition in Theorem 2.3 and convincingly tell us why equation (2.6) is true for all N.

(1) (10%) Do Exercise 2.2-2, 2.3-2, and 2.4-2 of LFD.

(2) (10%) Do Exercise 2.2-3, 2.3-3, and 2.4-3 of LFD.

(3) (5%) Do Exercise 2.5-1 of LFD.

(4) (5%) Do Exercise 2.5-2 of LFD.

(5) (5%) Do Exercise 2.5-3 of LFD.

(6) (5%) Do Exercise 2.5-4 of LFD.

(7) (10%) In class, we mentioned that the VC dimension of the perceptron hypothesis set corresponds to the number of parameters $(w_0, w_1, \cdots, w_d)$ of the set, and the idea is "usually" true for other hypothesis sets. On the other hand, we will present a counter-example here. Prove that the following hypothesis set for $x \in \mathbb{R}$ is of an infinite VC dimension:

$$\mathcal{H} = \left\{ h_\alpha : h_\alpha(x) = (-1)^{\lfloor \alpha x \rfloor}, \text{ where } \alpha \in \mathbb{R} \text{ and } \lfloor A \rfloor = \max\{n \in \mathbb{Z}, n \le A\} \text{ is the floor function.} \right\}$$

This hypothesis set comes with only one parameter α but "enjoys" an infinite VC dimension.

3.2 Proof of the Upper Bound

In this problem, we will prove that $m_{\mathcal{H}}(N) \le \left(\frac{eN}{d_{\text{VC}}}\right)^{d_{\text{VC}}}$ for $N \ge d_{\text{VC}} \ge 1$.

(1) (5%) Prove that for $N \ge d \ge 1$, $\left(\frac{d}{N}\right)^d \sum_{i=0}^{d} \binom{N}{i} \le \left(1 + \frac{d}{N}\right)^N$.

(2) (5%) Using the result above, prove that for $N \geq d \geq 1$, $\sum_{i=0}^{d} \binom{N}{i} \leq \left(\frac{eN}{d}\right)^d$.

(3) (5%) Using the result above, argue that you have proved that $m_{\mathcal{H}}(N) \leq \left(\frac{eN}{d_{\text{VC}}}\right)^{d_{\text{VC}}}$ for $N \geq d_{\text{VC}} \geq 1$.

3.3 Polynomial Hypotheses

In this problem, we will consider $\mathcal{X} = \mathbb{R}$. That is, $\mathbf{x} = x$ is a one-dimensional variable. For a hypothesis set $\mathcal{H} = \left\{h_{\mathbf{c}} : h_{\mathbf{c}}(\mathbf{x}) = \text{sign}\left(\sum_{i=0}^{D} c_i x^i\right)\right\}$. We will prove that the VC dimension of $\mathcal{H}$ is exactly $(D+1)$ by showing that

(1) (10%) There are $(D+1)$ points on which all 2^{D+1} label patterns can be produced from $\mathcal{H}$.

(2) (10%) There are no $(D+2)$ points on which all 2^{D+2} label patterns can be produced from $\mathcal{H}$.

3.4 Pocket Algorithm (*)

Do exercise 3.2 of LFD.

(1) (5%) Generate a data set of size 100 as directed by the exercise, and plot the examples $\{(\mathbf{x}_n, y_n)\}$ as well as the target function f on a plane. Be sure to mark the examples from different classes differently, and add labels to the axes of the plot. Generate a test set of size 1000 of the same nature.

Next, implement the pocket algorithm and run it on the data set for 1000 updates. Record $E_{\text{in}}(\mathbf{w}(t))$, $E_{\text{in}}(\mathbf{w}^*(t))$, $E_{\text{out}}(\mathbf{w}(t))$, and $E_{\text{out}}(\mathbf{w}^*(t))$ as functions of t (where E_{out} is estimated by the test set). Repeat the experiment for 20 times.

(2) (5%) Plot the average $E_{\text{in}}(\mathbf{w}(t))$ and $E_{\text{in}}(\mathbf{w}^*(t))$ as functions of t and briefly state your findings.

(3) (5%) Plot the average $E_{\text{out}}(\mathbf{w}(t))$ and $E_{\text{out}}(\mathbf{w}^*(t))$ as functions of t and briefly state your findings.

3.5 Mysterious B-function Leads to Bonus

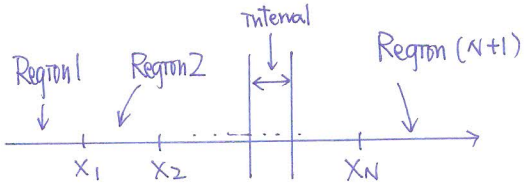
Recall that we proved $B(N,n) \leq \sum_{i=0}^{n-1} \binom{N}{i}$ in class.

(1) (Bonus 5%) Prove the following inequality: $B(N,n) \geq \sum_{i=0}^{n-1} \binom{N}{i}$.

Thus, $B(N,n) = \sum_{i=0}^{n-1} \binom{N}{i}$.

ML HW#3 R98922081 徐伍文

3.1.



100+5

(1) 1° $m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1$

$\binom{N+1}{2}$: two ends of the interval fall in two different regions

$+1$: two ends of the interval fall in the same region and have all-(-1)-pattern

2° When $n=3$, $m_{\mathcal{H}}(3) = \binom{3+1}{2} + 1 = 7 < 2^3$

$$\Rightarrow \sum_{i=0}^{n-1}\binom{N}{i} = \binom{N}{0} + \binom{N}{1} + \binom{N}{2} = 1 + N + \frac{N(N-1)}{2} = \frac{N(N+1)}{2} + 1$$

$$\therefore m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1 = \frac{N(N+1)}{2} + 1 \le \sum_{i=0}^{n-1}\binom{N}{i}, \quad n=3.$$

The statement is true.

3° $n=2$: $m_{\mathcal{H}}(2) = \binom{3}{2} + 1 = 4 = 2^2$ $\Rightarrow$ dvc = 2

$n=3$: $m_{\mathcal{H}}(3) = \binom{4}{2} + 1 = 7 < 2^3$

(2) 1° $m_{\mathcal{H}}(N) = \binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{N} = 2^N$

∵ If we put all N points on the perimeter of a circle, we can choose any one point or two points from the set of N points which are covered by some convex set. Besides, we can choose any k points as the set of the convex region $(3 \le k \le N)$ and there is no any other points inside the region which guarantees that the other $(N-k)$ points are in the case of (-1).

2° ∵ There is no any n that $m_{\mathcal{H}}(n) < 2^n$.

∴ The bound is no use in this case.

3° ∵ $m_{\mathcal{H}}(N) = 2^N$

∴ $dvc(\mathcal{H}) = \infty$

(3) There is a set of k points that is shattered by $\mathcal{H}$.

$\Rightarrow$ dvc $\ge k$

(4) There is a set of k points that is not shattered by $\mathcal{H}$.

$\Rightarrow$ We can't conclude anything. Because we're not sure that whethere there is a set of k points that can be shattered by $\mathcal{H}$.

(5) Any set of k points is shattered by $\mathcal{H}$.

$\Rightarrow$ There is a set of k points that is shattered by $\mathcal{H}$

$\Rightarrow$ dvc $\ge k$

(6) No set of k points is shattered by $\mathcal{H}$

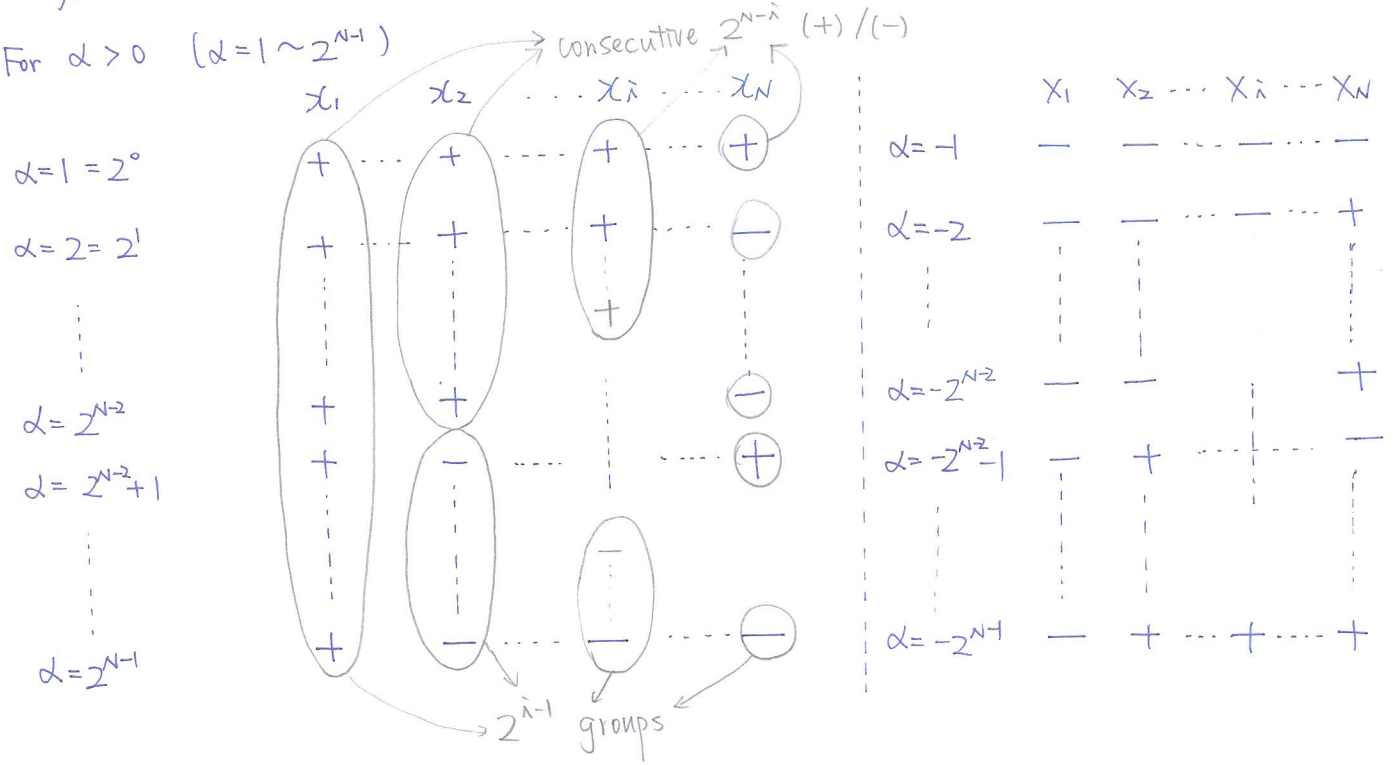
$\Rightarrow$ dvc $< k$ (dvc $\le k-1$)

(7) We want to make a set of N points and show that this set of N points is shattered by $\mathbb{H}$ for all $N \in \mathbb{N}$.

$\Rightarrow$ N points $\rightarrow$ $x_1, x_2, \cdots, x_N$, $0 < x_1 < x_2 < \cdots < x_N < 1$

2^N hypotheses $\rightarrow$ h_α : $\alpha = \pm 1, \pm 2, \cdots, \pm 2^{N-1}$

$\Rightarrow$ For $\alpha > 0$ $(\alpha = 1 \sim 2^{N-1})$



$\because$ $0 < x_1 < x_2 < \cdots < x_N$ $\therefore$ all-positive when $\alpha = 1$

x_i : started from "+" and flipped the label for every consecutive 2^{N-i}.

$\therefore$ There are 2^{i-1} groups of consecutive label. $(2^{N-1} / 2^{N-i} = 2^{i-1})$

For every x_i : $\begin{cases} 2^{N-i} \cdot x_i < 1 \\ (2^{N-1} - 2^{N-i} + 1) \cdot x_i < 2^{i-1} - 1 \end{cases}$

$$\Rightarrow \frac{2^{i-1}-1}{2^{N-1}-2^{N-i}+1} < x_i < \frac{1}{2^{N-i}} \quad , \; i = 1 \sim N$$

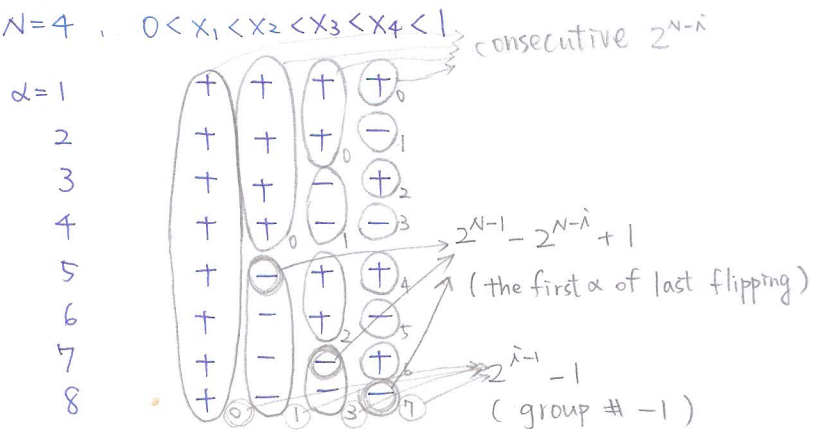
$\Rightarrow$ For $\alpha < 0$ $(\alpha = -1 \sim -2^{N-1})$

We can have the other 2^{N-1} patterns by flipping each label from the corresponding pattern when $\alpha > 0$.

Therefore, there is a set of N points that is shattered by $\mathbb{H}$ for all $N \in \mathbb{N}$.

That is, $m_{\mathbb{H}}(N) = 2^N$ for all N $\Rightarrow$ $\text{dvc}(\mathbb{H}) = \infty$

e.g. $N = 4$, $0 < x_1 < x_2 < x_3 < x_4 < 1$



$8x_1 < 1$, $x_1 > 0$ $\Rightarrow$ $0 < x_1 < \frac{1}{8}$

$4x_2 < 1$, $5x_2 > 1$ $\Rightarrow$ $\frac{1}{5} < x_2 < \frac{1}{4}$

$2x_3 < 1$, $7x_3 > 3$ $\Rightarrow$ $\frac{3}{7} < x_3 < \frac{1}{2}$

$x_4 < 1$, $8x_4 > 7$ $\Rightarrow$ $\frac{7}{8} < x_4 < 1$

3.2.

(1) $(1+\frac{d}{N})^N = \sum_{i=0}^{N} \binom{N}{i} (-1)^{N-i} (\frac{d}{N})^i$

$= \sum_{i=0}^{N} \binom{N}{i} (\frac{d}{N})^i$

$\geq \sum_{i=0}^{d} \binom{N}{i} (\frac{d}{N})^i$

$= (\frac{d}{N})^d \left[\sum_{i=0}^{d} \binom{N}{i} (\frac{d}{N})^{i-d} \right]$

$= (\frac{d}{N})^d \left[\sum_{i=0}^{d} \binom{N}{i} (\frac{N}{d})^{d-i} \right]$ $\because d \geq i \Rightarrow d-i \geq 0$

and $1 \leq d \leq N \Rightarrow \frac{N}{d} \geq 1$

$\Rightarrow (\frac{N}{d})^{d-i} \geq 1$

$\geq (\frac{d}{N})^d \cdot \sum_{i=0}^{d} \binom{N}{i}$

(2) $(1+\frac{d}{N})^N \geq (\frac{d}{N})^d \cdot \sum_{i=0}^{d} \binom{N}{i}$

make $\chi = \frac{N}{d} \Rightarrow N = \chi d$

then $(1+\frac{d}{N})^N = (1+\frac{1}{\chi})^{\chi d}$

$\leq \left[\lim_{x \to \infty} (1+\frac{1}{x})^x \right]^d = e^d$

$\Rightarrow \sum_{i=0}^{d} \binom{N}{i} \leq (\frac{N}{d})^d (1+\frac{d}{N})^N \leq (\frac{eN}{d})^d$

(3) $m_H(N) \leq B(N, dvc+1) \leq \sum_{i=0}^{dvc} \binom{N}{i} \leq (\frac{eN}{dvc})^{dvc}$

3.3.

(1) 1° Make $\vec{c} : (D+1) \times 1$ colume vector

$$\vec{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_D \end{bmatrix}$$

$\vec{x}_v : 1 \times (D+1)$ row vector

$\vec{x}_v = [1, x, x^2, \cdots, x^D]$

then $hc(x) = sign(\sum_{i=0}^{D} c_i x^i)$

$= sign(\vec{x}_v \cdot \vec{c})$

2° Considering $(D+1)$ different points : $x_1, x_2 \cdots x_{D+1}$, We make $X : (D+1) \times (D+1)$ matrix.

$$X = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_{D+1} \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^D \\ 1 & x_2 & x_2^2 & \cdots & x_2^D \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{D+1} & x_{D+1}^2 & \cdots & x_{D+1}^D \end{bmatrix}$$

then let $\vec{r}$: the result of these $(D+1)$ points

$\Rightarrow \vec{r} = sign(X\vec{c})$, $\vec{r} : (D+1) \times 1$ colume vector.

3° $\det(X) = \prod_{1 \le i < j \le (D+1)} (x_j - x_i)$ $\longrightarrow$ Vandermonde Matrix

$\because x_1 \sim x_{D+1}$ are $(D+1)$ different points.

$\therefore (x_j - x_i) \neq 0$ for all $1 \le i < j \le (D+1)$

$\Rightarrow \det(X) \neq 0 \Leftrightarrow X$: invertible

4° $\because X$: invertible

$\therefore \vec{y} = X\vec{c}$ has a solution $\vec{c}$ for any $\vec{y}$

$\Rightarrow \vec{r} = \text{sign}(X\vec{c})$ can have all possible patterns.

$\therefore (D+1)$ points can be shattered.

$\Rightarrow d_{VC} \ge D+1$

(2) There are $(D+2)$ vectors in $(D+1)$-dimensions.

$\therefore \vec{X}_1 \sim \vec{X}_{D+2}$: linearly dependent

$\Rightarrow \vec{X}_{D+2} = \sum_{v=1}^{D+1} \alpha_v \vec{X}_v$, and $\prod_{v=1}^{D+1} \alpha_v \neq 0$

For any hypothesis h_c : $h_c(x_v) = \text{sign}(\alpha_v)$ for each $\alpha_v \neq 0$, $v = 1 \sim D+1$

$\Rightarrow h_c(x_{D+2}) = \text{sign}(\vec{X}_{D+2} \cdot \vec{c})$

$= \text{sign}\left[\sum_{v=1}^{D+1} \alpha_v \cdot (\vec{X}_v \cdot \vec{c})\right]$

$\because h_c(\vec{X}_v) = \text{sign}(\vec{X}_v \cdot \vec{c}) = \text{sign}(\alpha_v)$, $v = 1 \sim D+1$

$\therefore \alpha_v \cdot (\vec{X}_v \cdot \vec{c}) > 0$

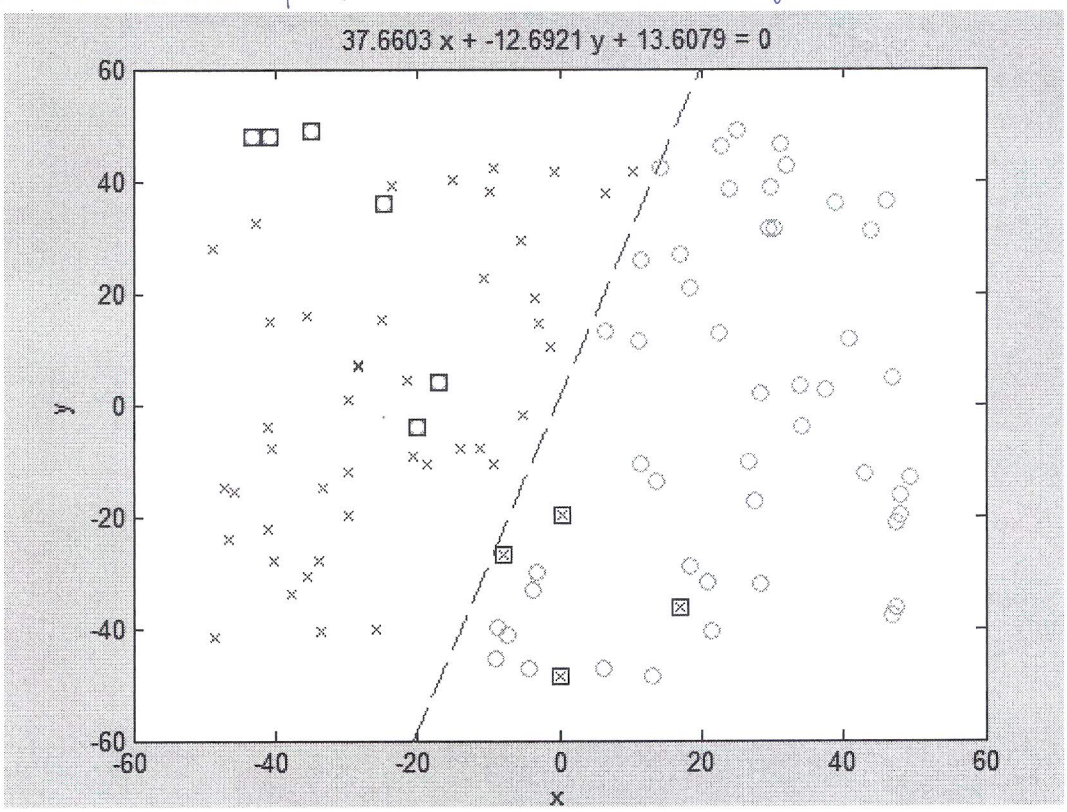
$\Rightarrow h_c(x_{D+2}) = "+1"$

Therefore, there is no any hypothesis "h" that $h(x_v) = \text{sign}(\alpha_v)$ for $v = 1 \sim D+1$ and $h_c(x_{D+2}) = "-1"$. That is to say, the pattern $(\text{sign}(\alpha_1), \text{sign}(\alpha_2), \cdots \text{sign}(\alpha_{D+1}), -1)$ can not be generated.

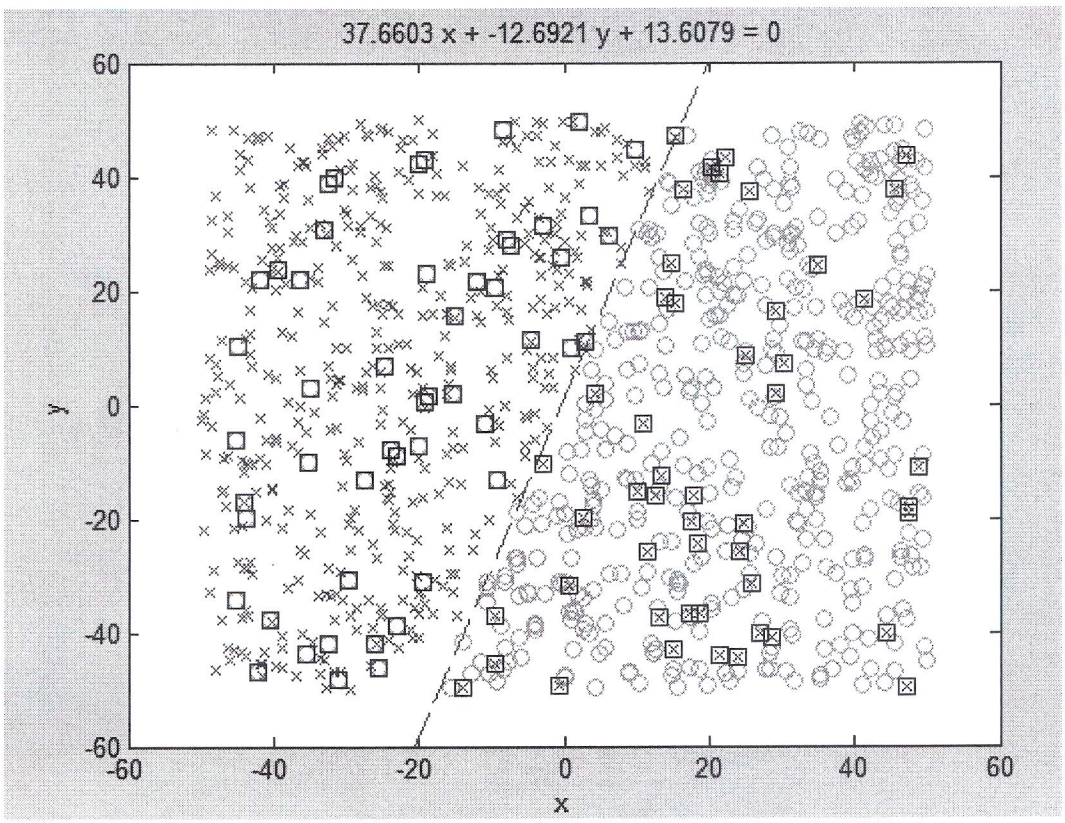
$\Rightarrow d_{VC} \le d+1$

3.4.

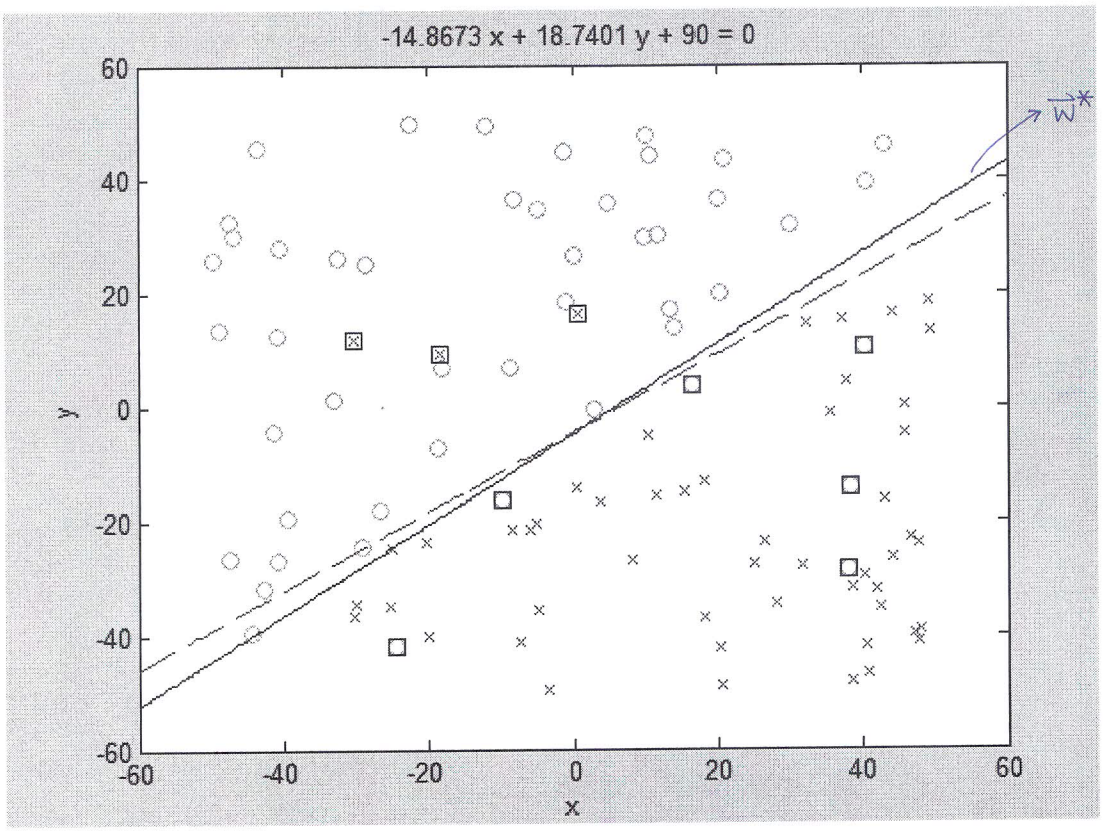
(1) Data set of size 100 which is not linearly separable.
The misclassified points are marked with squares.



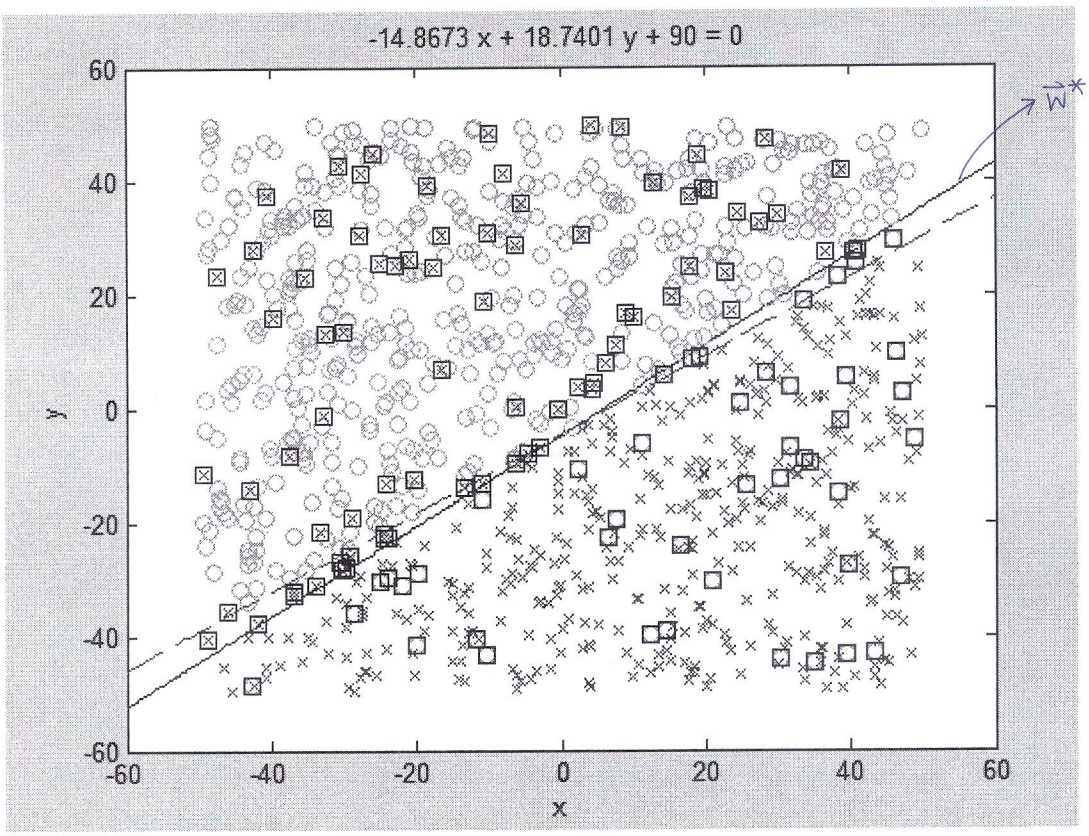
Data set of size 1000 as a test set.



Another experiment of Pocket Algorithm with $\vec{w}^*$.

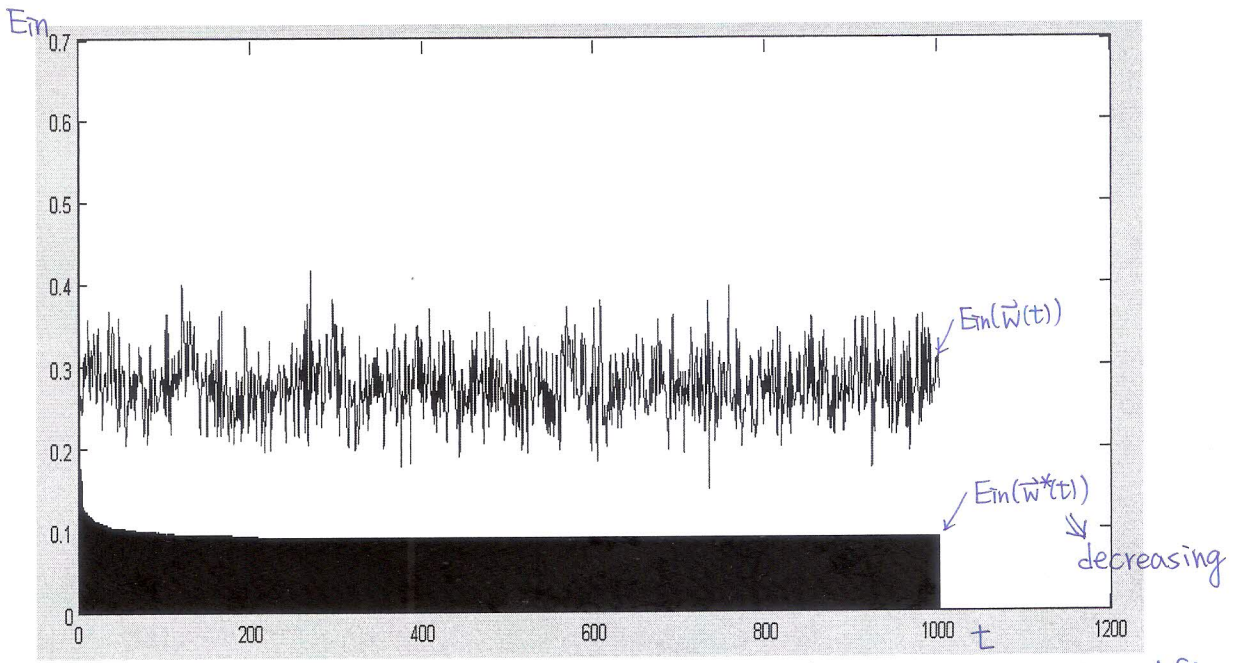


$\vec{w}^*$ from Pocket Algorithm in test set of size 1000.



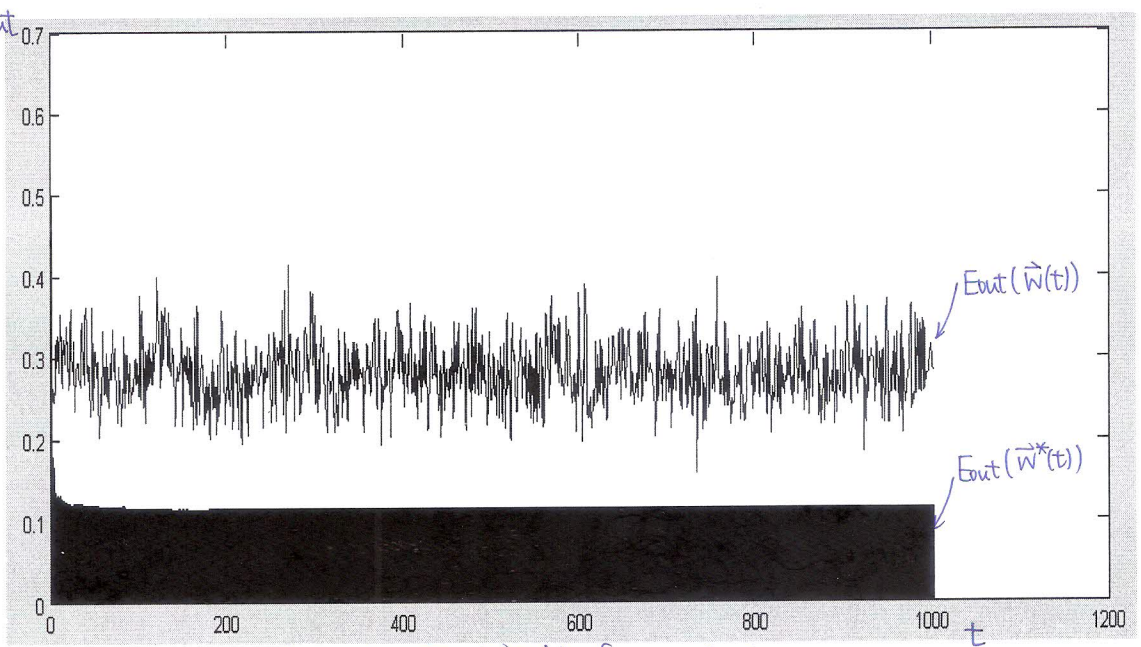
(2) The iterations No. encountered the best $\vec{w}^*$ from 20 experiments

$\Rightarrow$ 58 68 101 291 112 214 318 210 28 208
183 59 95 38 82 157 180 145 202 59



At the beginning, $E_{in}(\vec{w}^*)$ and $E_{in}(\vec{w})$ are quite large (about 0.5). After some iterations, $\vec{w}(t)$ became much better than before, so $E_{in}(\vec{w}^*)$ decreased very soon. And the $E_{in}(\vec{w}^*)$ curve converged to 0.1 eventually. However, every $E_{in}(\vec{w}(t))$ is different from each t of 20 experiments, therefore, $E_{in}(\vec{w})$ seems to be up and down around 0.3.

(3)



$E_{out}(\vec{w}(t))$ is very close to $E_{in}(\vec{w}(t))$ for each t.

$E_{out}(\vec{w}^*)$ acts like $E_{in}(\vec{w}^*)$ a lot, and $E_{out}(\vec{w}^*)$ curve converged to about 0.1. $E_{out}(\vec{w}^*)$ seems to be decreasing in general, however, it is not necessary to decrease all the time, because $\vec{w}^*$ resulted in min. Ein didn't guarantee to have min. Eout.

3.5.

We want to make a len-N-pattern set $\mathbb{I}$ and $\mathbb{I}$: any len-n-subpattern not shattered.

$\Rightarrow |\mathbb{I}| \le B(N, n)$

We let $\mathbb{I} = A_0 \cup A_1 \cup \cdots \cup A_{n-1}$

and A_0 : no-(-1)-pattern $+++\cdots+$ $\quad |A_0| = 1$

A_1 : one-(-1)-pattern $-++\cdots+$, $+-+\cdots+$, $++-\cdots+$, $\vdots$, $+++\cdots-$ $\quad |A_1| = \binom{N}{1} = N$

A_i : i-(-1)-pattern, Choose any i points (-1), the other points (+1) $\quad |A_i| = \binom{N}{i}$

A_{n-1} : $(n-1)$-(-1)-pattern $\quad \cdots\cdots$ $\quad |A_{n-1}| = \binom{N}{n-1}$

$\because A_i \cap A_j = \phi$ for all $i \neq j$

$\therefore |\mathbb{I}| = |A_0| + |A_1| + \cdots + |A_{n-1}| = \sum_{i=0}^{n-1} |A_i|$

Because there is no any len-n-subpattern of $\mathbb{I}$ with all (-1).

Therefore, it guarantee that $\mathbb{I}$: any len-n-subpattern not shattered.

$\Rightarrow B(N, n) \ge |\mathbb{I}| = \sum_{i=0}^{n-1} |A_i| = \sum_{i=0}^{n-1} \binom{N}{i}$

Homework #4

TA in charge: Chao-Kai Chiang

RELEASE DATE: 10/25/2010

DUE DATE: 11/08/2010, 4:00 pm IN CLASS

TA SESSION: 11/04/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

4.1 More on Growth Function and VC Dimension

(1) (5%) Let $\mathcal{H} = \{h_1, h_2, \ldots, h_M\}$ with some finite M. Prove that $d_{\mathsf{VC}}(\mathcal{H}) \leq \log_2 M$.

(2) (5%) For hypothesis sets $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_K$ with finite VC-dimensions $d_{\mathsf{VC}}(\mathcal{H}_k)$, derive and prove the tightest lower bound that you can get on $d_{\mathsf{VC}}\left(\cap_{k=1}^{K} \mathcal{H}_k\right)$.

(3) (5%) For hypothesis sets $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_K$ with finite VC-dimensions $d_{\mathsf{VC}}(\mathcal{H}_k)$, derive and prove the tightest upper bound that you can get on $d_{\mathsf{VC}}\left(\cap_{k=1}^{K} \mathcal{H}_k\right)$.

(4) (5%) For hypothesis sets $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_K$ with finite VC-dimensions $d_{\mathsf{VC}}(\mathcal{H}_k)$, derive and prove the tightest lower bound that you can get on $d_{\mathsf{VC}}\left(\cup_{k=1}^{K} \mathcal{H}_k\right)$.

(5) (5%) For hypothesis sets $\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_K$ with finite VC-dimensions $d_{\mathsf{VC}}(\mathcal{H}_k)$, derive and prove the tightest upper bound that you can get on $d_{\mathsf{VC}}\left(\cup_{k=1}^{K} \mathcal{H}_k\right)$.

4.2 The Hat of Linear Regression

(1) (3%) Do Exercise 3.3-1 of LFD.

(2) (3%) Do Exercise 3.3-2 of LFD.

(3) (3%) Do Exercise 3.3-3 of LFD.

(4) (3%) Do Exercise 3.3-4 of LFD.

(5) (3%) Do Exercise 3.3-5 of LFD.

4.3 The Feature Transforms

(1) (4%) Do Exercise 3.6 of LFD.

(2) (6%) Do Exercise 3.7 of LFD.

(3) (5%) Do Exercise 3.11 of LFD.

4.4 Gradient and Newton Directions

Consider a function

$$E(u,v) = e^u + e^{2v} + e^{uv} + u^2 - 3uv + 4v^2 - 3u - 5v,$$

(1) (3%) Approximate $E(u+\Delta u, v+\Delta v)$ by $\hat{E}_1(\Delta u, \Delta v)$, where $\hat{E}_1$ is the first-order Taylor's expansion of E around $(u,v) = (0,0)$. Suppose $\hat{E}_1(\Delta u, \Delta v) = a_u \Delta u + a_v \Delta v + a$. What are the values of a_u, a_v, and a?

(2) (3%) Minimize $\hat{E}_1$ over all possible $(\Delta u, \Delta v)$ such that $\|(\Delta u, \Delta v)\| = 0.5$. In class, we proved that the optimal column vector $\begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix}$ is parallel to the column vector $-\nabla E(u,v)$, which is called the *negative gradient direction*. Compute the optimal $(\Delta u, \Delta v)$ and the resulting $E(u+\Delta u, v+\Delta v)$.

(3) (3%) Approximate $E(u+\Delta u, v+\Delta v)$ by $\hat{E}_2(\Delta u, \Delta v)$, where $\hat{E}_2$ is the second-order Taylor's expansion of E around $(u,v) = (0,0)$. Suppose

$$\hat{E}_2(\Delta u, \Delta v) = b_{uu}(\Delta u)^2 + b_{vv}(\Delta v)^2 + b_{uv}(\Delta u)(\Delta v) + b_u \Delta u + b_v \Delta v + b.$$

What are the values of b_{uu}, b_{vv}, b_{uv}, b_u, b_v, and b?

(4) (3%) Minimize $\hat{E}_2$ over all possible $(\Delta u, \Delta v)$ (regardless of length). Use the fact that $\nabla^2 E(u,v)$ (the Hessian matrix) is positive definite to prove that the optimal column vector

$$\begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix} = -\left(\nabla^2 E(u,v)\right)^{-1} \nabla E(u,v),$$

which is called the *Newton direction*.

(5) (3%) Numerically compute the following values:

(a) the vector $(\Delta u, \Delta v)$ of length 0.5 along the Newton direction, and the resulting $E(u+\Delta u, v+\Delta v)$.

(b) the vector $(\Delta u, \Delta v)$ of length 0.5 that minimizes $E(u+\Delta u, v+\Delta v)$, and the resulting $E(u+\Delta u, v+\Delta v)$. (*Hint: let* $\Delta u = 0.5 \sin\theta$.)

Compare the values of $E(u+\Delta u, v+\Delta v)$ in (2), (5a), and (5b). Briefly state your findings.

The negative gradient direction and the Newton direction are quite fundamental for designing optimization algorithms. It is important to understand these directions and put them in your toolbox for designing ML algorithms.

4.5 Least-squares Linear Regression (*)

(1) (8%) Implement the least-squares linear regression algorithm taught in class to compute the optimal $(d+1)$-dimensional $\mathbf{w}$ that solves

$$\min_{\mathbf{w}} \sum_{n=1}^{N} \left(y_n - (\mathbf{w} \cdot \mathbf{x}_n)\right)^2.$$

Run the algorithm on the following set for training (each row represents a pair of $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n$ is the "thin" version. The first column is $\mathbf{x}_n[1]$, the second one is $\mathbf{x}_n[2]$, and the third one is y_n):

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_test.dat`

Report the $\mathbf{w}$ you find. Let $g(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$. What is $E_{\text{in}}(g)$ in terms of the 0/1 loss (classification)? How about $E_{\text{out}}(g)$?

Please check the course policy carefully and do not use sophisticated packages in your solution. You ***can*** *use standard matrix multiplication and inversion routines.*

4.6 Gradient Descent for Logistic Regression (*)

Consider the formulation (so-called *logistic regression*)

$$\min_{\mathbf{w}} \quad E(\mathbf{w}), \tag{A1}$$

$$\text{where} \quad E(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} E^{(n)}(\mathbf{w}), \text{ and } E^{(n)}(\mathbf{w}) = \ln\left(1 + \exp\left(-y_n\left(\mathbf{w} \bullet \mathbf{x}_n\right)\right)\right).$$

(1) (3%) Prove that $\frac{1}{\ln 2} E^{(n)}(\mathbf{w})$ is an upper bound of $[\![\text{sign}(\mathbf{w} \bullet \mathbf{x}_n) \neq y_n]\!]$ for any $\mathbf{w}$.

(2) (3%) For a given $(\mathbf{x}_n, y_n)$, derive its gradient $\nabla E^{(n)}(\mathbf{w})$.

(3) (8%) Implement the (fixed-step) stochastic gradient descent algorithm below for (A1).

(a) initialize a $(d+1)$-dimensional vector $\mathbf{w}^{(0)}$, say, $\mathbf{w}^{(0)} \longleftarrow (0, 0, \ldots, 0)$.

(b) for $t = 1, 2, \ldots, T$

- randomly pick one n from $\{1, 2, \ldots, N\}$.
- update

$$\mathbf{w}^{(t)} \longleftarrow \mathbf{w}^{(t-1)} - \eta \cdot \nabla E^{(n)}(\mathbf{w}^{(t-1)}).$$

Assume that

$$g_1^{(t)}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{(t)} \bullet \mathbf{x}\right),$$

where $\mathbf{w}^{(t)}$ are generated from stochastic gradient descent algorithm above. Run the algorithm with $\eta = 0.001$ and $T = 2000$ on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_test.dat`

Plot $E_{\text{in}}\left(g_1^{(t)}\right)$ and $E_{\text{out}}\left(g_1^{(t)}\right)$ as a function of t and briefly state your findings.

(4) (8%) Implement the (fixed-step) gradient descent algorithm below for (A1).

(a) initialize a $(d+1)$-dimensional vector $\mathbf{w}^{(0)}$, say, $\mathbf{w}^{(0)} \longleftarrow (0, 0, \ldots, 0)$.

(b) for $t = 1, 2, \ldots, T$

- update

$$\mathbf{w}^{(t)} \longleftarrow \mathbf{w}^{(t-1)} - \eta \cdot \nabla E(\mathbf{w}^{(t-1)}).$$

Assume that

$$g_2^{(t)}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{(t)} \bullet \mathbf{x}\right),$$

where $\mathbf{w}^{(t)}$ are generated from gradient descent algorithm above. Run the algorithm with $\eta = 0.001$ and $T = 2000$ on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw4_test.dat`

Plot $E_{\text{in}}\left(g_2^{(t)}\right)$ and $E_{\text{out}}\left(g_2^{(t)}\right)$ as a function of t, compare it to your plot for $g_1^{(t)}$, and briefly state your findings.

ML HW#4 R98922081 徐位文

95

4.1

(1) If $dvc(\mathcal{H}) = d$, then $m_{\mathcal{H}}(d) = 2^d$.

$\Rightarrow$ There are at least 2^d hypothese in $\mathcal{H}$.

$\Rightarrow M \geq 2^d$

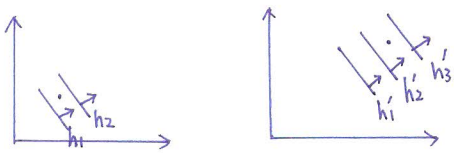
$\Rightarrow \log_2 M \geq d$

$\Rightarrow dvc(\mathcal{H}) \leq \log_2 M$.

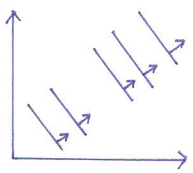
(2) If $\mathcal{H} = \bigcap_{k=1}^{K} \mathcal{H}_k$, then $dvc(\mathcal{H}) \geq 0$.

e.g. $\mathcal{H}_1 = \{h_1, h_2\}$, $\mathcal{H}_2 = \{h_1', h_2', h_3'\}$

and $dvc(\mathcal{H}_1) = 1 = dvc(\mathcal{H}_2)$



$h_1 // h_2 // h_1' // h_2' // h_3'$

$\Rightarrow$ 

There is no any point can be shattered by both $\mathcal{H}_1$ and $\mathcal{H}_2$ in this case.

$\therefore$ We can't guarantee that $dvc(\mathcal{H}_1 \cap \mathcal{H}_2) = 1$ for any two arbitrary hypothesis sets.

(3) If $\mathcal{H} = \bigcap_{k=1}^{K} \mathcal{H}_k$ and $\min_k dvc(\mathcal{H}_k) = d_{min}$, then $dvc(\mathcal{H}) \leq d_{min}$.

$\because$ The best situation is that there is a set of d_{min} points which can be shattered by the hypothesis set that has min. dvc can also be shattered by other rest of hypothesis set, so that $dvc(\bigcap_{k=1}^{K} \mathcal{H}_k) = d_{min}$. In this situation. And it is impossible for the hypothesis set can not be shattered by $\mathcal{H}$. ($\mathcal{H} = \bigcap_{k=1}^{K} \mathcal{H}_k$). $\Rightarrow dvc(\mathcal{H}) \leq d_{min}$

(4) If $\mathcal{H} = \bigcup_{k=1}^{K} \mathcal{H}_k$ and $\max_k dvc(\mathcal{H}_k) = d_{max}$, then $dvc(\mathcal{H}) \geq d_{max}$.

$\because$ If $\mathcal{H}'$ is the hypothesis set that has the max. dvc and $\mathcal{H}' \subseteq \mathcal{H}$ ($\because \mathcal{H} = \bigcup_{k=1}^{K} \mathcal{H}_k$)

then the points set which can be shattered by $\mathcal{H}'$ can also be shattered by $\mathcal{H}$.

$\Rightarrow dvc(\mathcal{H}) \geq d_{max}$

(5) X If $\mathcal{H} = \bigcup_{k=1}^{K} \mathcal{H}_k$ and $|\mathcal{H}| = M$, then $dvc(\mathcal{H}) \leq \log_2 M$. (by (1))

H: finite

4.2

(1) To show that $\hat{y}$ lies in the span of the columns of X (denoted as CS(X)), i.e. $\hat{y} \in CS(X)$, and the error $\vec{y} - \hat{y}$ are orthogonal to every vectors lying in CS(X), i.e. $\vec{y} - \hat{y} \perp \vec{w}$, $\forall \vec{w} \in CS(X)$.

1° $\hat{y} \in CS(X)$

$\because H = X(X^TX)^{-1}X^T$

$\Rightarrow \hat{y} = H\vec{y} = X(X^TX)^{-1}X^T\vec{y} = X[(X^TX)^{-1}X^T\vec{y}] \in CS(X) \Rightarrow \hat{y} \in CS(X)$

2° $(\vec{y}-\hat{y}) \perp \vec{w}$, $\forall \vec{w} \in CS(X)$

$\Leftrightarrow \langle \vec{y}-\hat{y}, \vec{w} \rangle = 0$, $\forall \vec{w} \in CS(X)$

$\forall \vec{w} \in CS(X)$, $\exists \vec{u} + \vec{w} = X\vec{u}$

$\Rightarrow \langle \vec{y}-\hat{y}, \vec{w} \rangle = \vec{w}^T(\vec{y}-\hat{y})$

$= (X\vec{u})^T(\vec{y}-\hat{y})$

$= (X\vec{u})^T\vec{y} - (X\vec{u})^T\hat{y}$

$= \vec{u}^T X^T \vec{y} - \vec{u}^T X^T \hat{y}$ $\because \hat{y} = H\vec{y} = X(X^TX)^{-1}X^T\vec{y}$

$= \vec{u}^T X^T \vec{y} - \vec{u}^T \cancel{X^T X (X^TX)^{-1}} X^T \vec{y}$

$= \vec{u}^T X^T \vec{y} - \vec{u}^T X^T \vec{y}$

$= 0$

(2) $H = X(X^TX)^{-1}X^T$

$\Rightarrow H^2 = [X(X^TX)^{-1}X^T][X(X^TX)^{-1}X^T]$

$= X(X^TX)^{-1}\cancel{X^TX(X^TX)^{-1}}X^T$

$= X(X^TX)^{-1}X^T$

$= H$

(3) $H = X(X^TX)^{-1}X^T$

$\Rightarrow HX = X(\cancel{X^TX})^{-1}\cancel{X^TX} = X$

All column vectors of X lie in the projecting subspace CS(X).

By (2), projecting a projection gives the same thing.

∴ The projection of column vector of X into CS(X) is itself.

(4) $\vec{y} : N\times 1$. $H : N \times N$

$\Rightarrow$ The residual errors : $\vec{y}-\hat{y} = \vec{y} - H\vec{y} = (I-H)\vec{y}$

(5) ∵ H is a projection matrix which projects all vectors into CS(X).

∴ rank(H) = dim[R(H)] = dim[CS(X)] = rank(X) = d+1

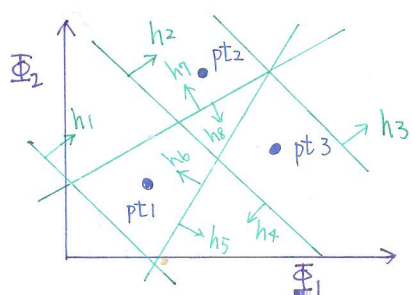
∵ I-H is the subspace that any vector $\vec{v} \in I-H \Rightarrow \vec{v} \perp \vec{w}$, $\forall \vec{w} \in H$

∴ $V = H \oplus (I-H)$ (direct sum)

dim(V) = N , rank(H) = d+1 $\Rightarrow$ rank(I-H) = N-(d+1)

4.3

(1) 1°



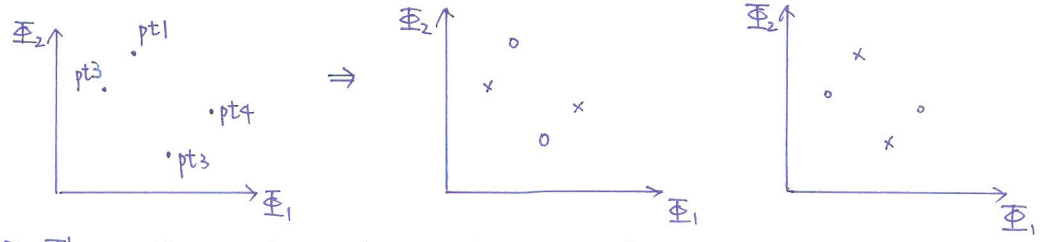
$h_1 \to (+,+,+)$, $h_2 \to (-,+,+)$, $h_3 \to (-,-,-)$

$h_4 \to (+,-,-)$, $h_5 \to (-,-,+)$, $h_6 \to (+,+,-)$

$h_7 \to (-,+,-)$, $h_8 \to (+,-,+)$

$\Rightarrow m_H(3) = 8$

2° ✗

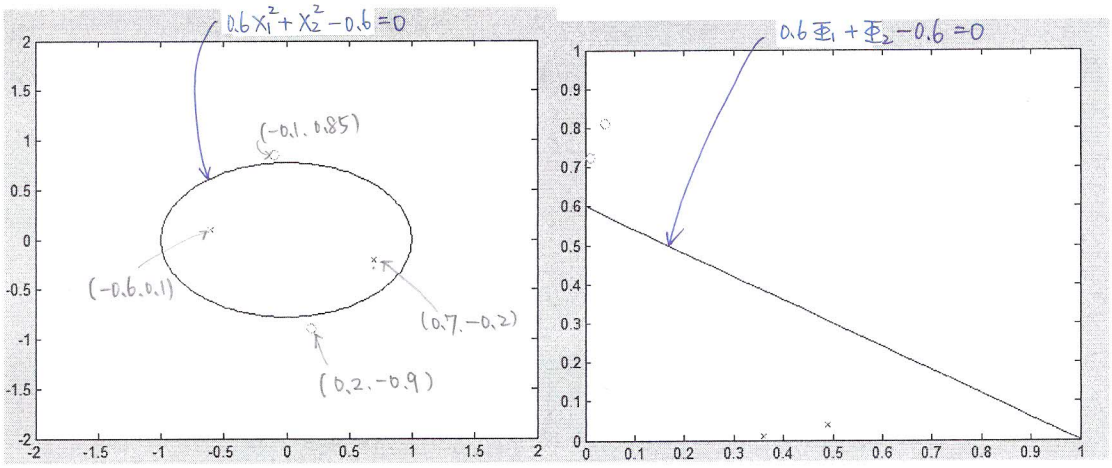


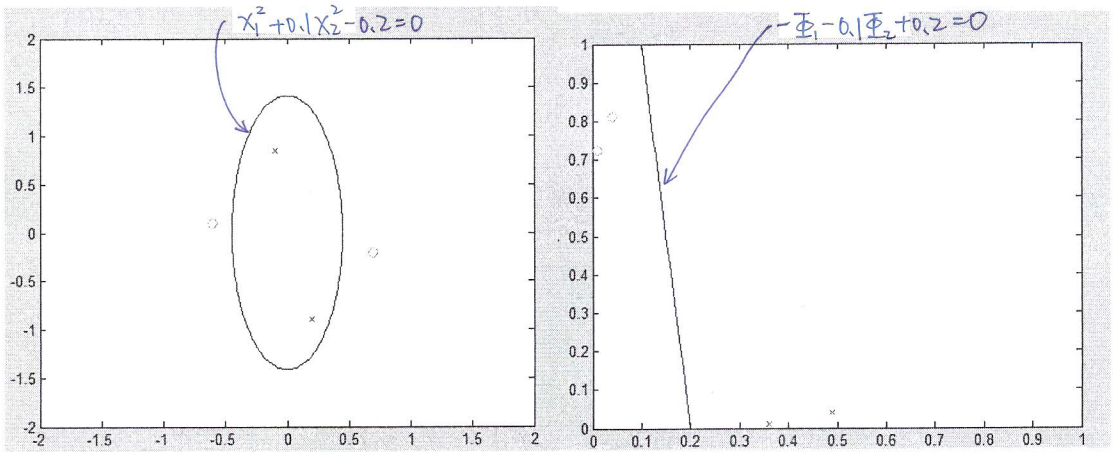
-1

$\Rightarrow$ The patterns $(+,-,+,-)$ and $(-,+,-,+)$ can not be generated by $\mathcal{H}_\Phi$.

$\therefore m_{\mathcal{H}_\Phi}(4) < 16$ for all 4-points.

3° ✓





$\Rightarrow$ This case show that the patterns which can not be generated by $\mathcal{H}$ (left-hand side) can be separated by the hypothesis from $\mathcal{H}_\Phi$ in the transformed feature space $\Phi(x)$. That is to say, the set of 4 points is shattered by $\mathcal{H} \cup \mathcal{H}_\Phi$, i.e. $m_{\mathcal{H} \cup \mathcal{H}_\Phi}(4) = 16$.

(2) ✓ $\Phi_2(\vec{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

1° $\Phi(x) = (8, -6, 1, 1, 0, 0)$

2° $\Phi(x) = (24, -6, -8, 1, 0, 1)$

3° $\Phi(x) = (33, -12, -8, 2, 0, 1)$

4° $\Phi(x) = (-8, -6, 8, 1, 0, -1)$

5° $\Phi(x) = (33, -20, -4, 3, 2, 3)$

6° $\Phi(x) = (-1, 2, 1, 0, 0, 0)$

-2

(3) ✗ No, $d_{VC} = 6$. Because we can use $d_{VC} = 3$ only in the case that we decide on Φ before seeing the data. However, in this case we tried a linear model and found it failed first.

(No, apply the linear model first.)

4.4

(1) First-order Taylor's expansion formula: $f(x,y) \approx f(a,b) + [f_x(a,b)](x-a) + [f_y(a,b)](y-b)$

$$\Rightarrow E(u+\Delta u, v+\Delta v) \approx E(u,v) + E_u(u,v)(\Delta u) + E_v(u,v)(\Delta v)$$

$$= E(u,v) + [e^u + ve^{uv} + 2u - 3v - 3](\Delta u)$$

$$+ [2e^{2v} + ue^{uv} - 3u + 8v - 5](\Delta v)$$

$\Rightarrow$ Expansion of E around $(u,v) = (0,0)$

$$E(u+\Delta u, v+\Delta v) \approx \hat{E}_1(\Delta u, \Delta v) = E(0,0) + E_u(0,0)(\Delta u) + E_v(0,0)(\Delta v)$$

$$= -2(\Delta u) - 3(\Delta v) + 3$$

$\Rightarrow a_u = -2, \quad a_v = -3, \quad a = 3.$

(2) $\nabla E(u,v) = \begin{bmatrix} E_u(u,v) \\ E_v(u,v) \end{bmatrix}$

$$\Rightarrow \min_{\|(\Delta u, \Delta v)\| = 0.5} \hat{E}_1(\Delta u, \Delta v) \Leftrightarrow \min_{\|(\Delta u, \Delta v)\| = 0.5} \nabla E(0,0) \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + \underbrace{E(0,0)}_{\text{const.}}$$

$$\Leftrightarrow \min_{\|(\Delta u, \Delta v)\| = 0.5} \nabla E(0,0) \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} \Rightarrow \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = -0.5 \frac{\nabla E(0,0)}{|\nabla E(0,0)|} = \frac{1}{2\sqrt{13}} \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$\Rightarrow E(u+\Delta u, v+\Delta v) = 2.2509$$

(3) Formula: $f(x,y) \approx f(a,b) + [f_x(a,b)](x-a) + [f_y(a,b)](y-b)$

$$+ \frac{1}{2!}\{[f_{xx}(a,b)](x-a)^2 + 2[f_{xy}(a,b)](x-a)(y-b) + [f_{yy}(a,b)](y-b)^2\}$$

$$\Rightarrow E(u+\Delta u, v+\Delta v) \approx E(u,v) + E_u(u,v)(\Delta u) + E_v(u,v)(\Delta v)$$

$$+ \frac{E_{uu}(u,v)}{2}(\Delta u)^2 + E_{uv}(u,v)(\Delta u)(\Delta v) + \frac{E_{vv}(u,v)}{2}(\Delta v)^2$$

$$= E(u,v) + [e^u + ve^{uv} + 2u - 3v - 3](\Delta u) + [2e^{2v} + ue^{uv} - 3u + 8v - 5](\Delta v)$$

$$+ \frac{1}{2}[e^u + v^2e^{uv} + 2](\Delta u)^2 + [e^{uv} + uve^{uv} - 3](\Delta u)(\Delta v)$$

$$+ \frac{1}{2}[4e^{2v} + u^2e^{uv} + 8](\Delta v)^2$$

$\Rightarrow$ Expansion of E around $(u,v) = (0,0)$

$$E(u+\Delta u, v+\Delta v) \approx \hat{E}_2(\Delta u, \Delta v) = E(0,0) + E_u(0,0)(\Delta u) + E_v(0,0)(\Delta v) + \frac{E_{uu}(0,0)}{2}(\Delta u)^2$$

$$+ E_{uv}(0,0)(\Delta u)(\Delta v) + \frac{E_{vv}(0,0)}{2}(\Delta v)^2$$

$$= \frac{3}{2}(\Delta u)^2 + 6(\Delta v)^2 - 2(\Delta u)(\Delta v) - 2(\Delta u) - 3(\Delta v) + 3$$

$\Rightarrow b_{uu} = \frac{3}{2}, \quad b_{vv} = 6, \quad b_{uv} = -2, \quad b_u = -2, \quad b_v = -3, \quad b = 3$

(4) $\hat{E}_2(\Delta u, \Delta v) = \frac{1}{2}[\Delta u, \Delta v][\nabla^2 E]\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + [\nabla E] \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + b$

$$\Rightarrow \min \hat{E}_2(\Delta u, \Delta v) \Leftrightarrow \min \frac{1}{2}[\Delta u, \Delta v][\nabla^2 E]\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + [\nabla E] \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + b \quad \because b = \text{const.}$$

$$\Leftrightarrow \min \frac{1}{2}[\Delta u, \Delta v][\nabla^2 E]\begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} + [\nabla E] \cdot \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}$$

$$\Leftrightarrow \min \frac{1}{2}\underset{1\times 2}{[\Delta u, \Delta v]}\underset{2\times 2}{[\nabla^2 E]}\underset{2\times 1}{\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}} + [\nabla E]\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$$\Leftrightarrow \min \frac{1}{2}\left([\Delta u, \Delta v][\nabla^2 E]\right)^T\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + [\nabla E]\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$$\Leftrightarrow \min \frac{1}{2}\left([\nabla^2 E]^T[\Delta u, \Delta v]^T\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + [\nabla E]\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$\because \nabla^2 E$: positive definite

$\therefore \nabla^2 E$: symmetric i.e. $[\nabla^2 E]^T = [\nabla^2 E]$

pf.

$\vec{w} = \begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$, $A = \nabla^2 E$, $\vec{b} = \nabla E$

$\hat{E}_2 = \frac{1}{2}\vec{w}^T A\vec{w} + \vec{b}^T\vec{w}$

$$\Leftrightarrow \min \frac{1}{2}\left([\nabla^2 E]\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + [\nabla E]\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$\Rightarrow \min. \hat{E}_2 \Leftrightarrow A\vec{w} + \vec{b} = 0$

$$\Leftrightarrow \min \left(\frac{1}{2}[\nabla^2 E]\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + [\nabla E]\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$\Rightarrow \vec{w} = -A^{-1}\vec{b}$

$= -\nabla^2 E(u,v)^{-1}\nabla E(u,v)$

$\because \nabla^2 E$: positive definite

$\therefore \det(\nabla^2 E) > 0 \Rightarrow \nabla^2 E$: invertible

$$\Leftrightarrow \min \left(\frac{1}{2}\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + [\nabla^2 E]^{-1}[\nabla E]\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$$\Leftrightarrow \min \left([\nabla^2 E]^{-1}[\nabla E]\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} + \frac{1}{2}\underline{\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}}$$

$\underline{\quad}$ → $\|(\Delta u, \Delta v)\|^2 \to$ const.

$$\Leftrightarrow \min \left([\nabla^2 E]^{-1}[\nabla E]\right)\cdot\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$$

$$\Rightarrow \begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} = -\left([\nabla^2 E]^{-1}[\nabla E]\right)$$

(5) 1° $\min_{\|(\Delta u, \Delta v)\|=0.5} \hat{E}_2(\Delta u, \Delta v)$

$$\Rightarrow \begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} = -0.5\,\frac{\begin{bmatrix}3 & -2\\ -2 & 12\end{bmatrix}^{-1}\begin{bmatrix}-2\\ -3\end{bmatrix}}{\left\|\begin{bmatrix}3 & -2\\ -2 & 12\end{bmatrix}^{-1}\begin{bmatrix}-2\\ -3\end{bmatrix}\right\|} = \begin{bmatrix}0.4588\\ 0.1988\end{bmatrix}$$

$\Rightarrow E(u+\Delta u, v+\Delta v) = 1.8905$

2° $\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} = \begin{bmatrix}0.5\sin\theta\\ 0.5\cos\theta\end{bmatrix}$

For $\theta = 1°, 2°, 3°, \cdots, 360°$

$0.5\cdot\sin(61°) = 0.4373$

$0.5\cdot\cos(61°) = 0.2424$

By iterations, when $\theta = 69°$, $E(\Delta u, \Delta v)$ has min. value 1.8685 Compared with others

3° By $\min. \hat{E}_1(\Delta u, \Delta v) \Rightarrow E(\Delta u, \Delta v) = 2.2509$

By $\min. \hat{E}_2(\Delta u, \Delta v) \Rightarrow E(\Delta u, \Delta v) = 1.8905$

$\min. E(\Delta u, \Delta v) \Rightarrow E(\Delta u, \Delta v) = 1.8685$

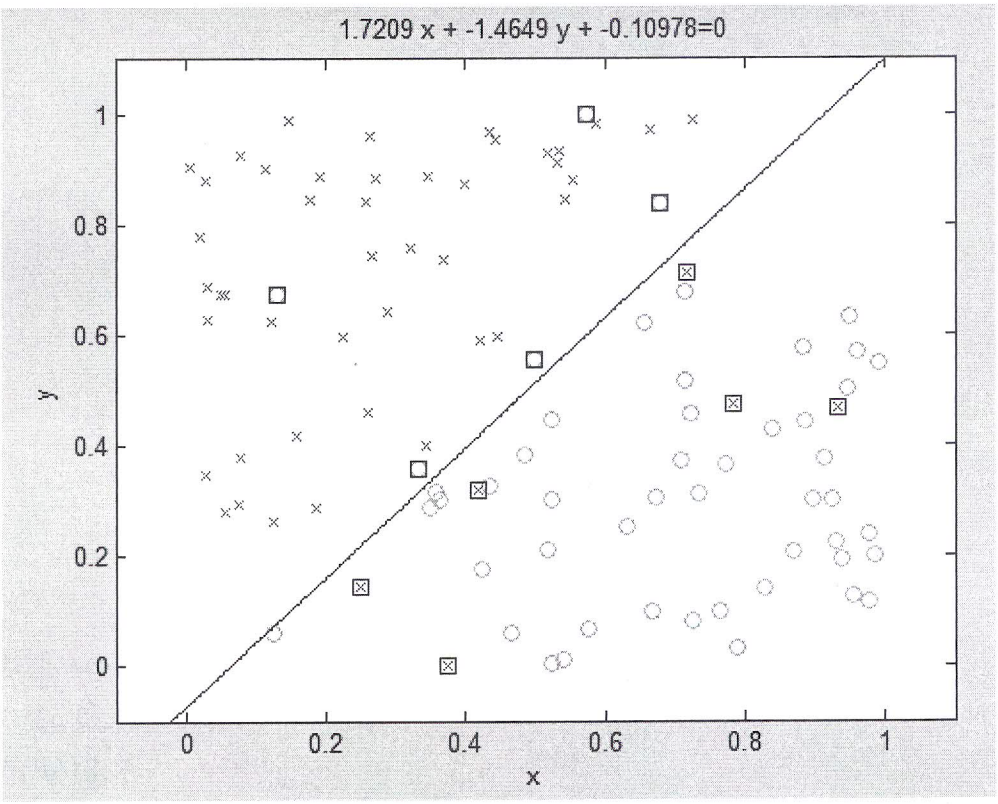
$\Rightarrow$ 1.8685 is the smallest value and the value made by $\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$ by $\min. \hat{E}_2$ is smaller than by $\min. \hat{E}_1$, i.e. $\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix}$ by $\min \hat{E}_2$ is more precise than by $\min. \hat{E}_1$.

4.5

1°

w = [-0.1098 1.7209 -1.4649]

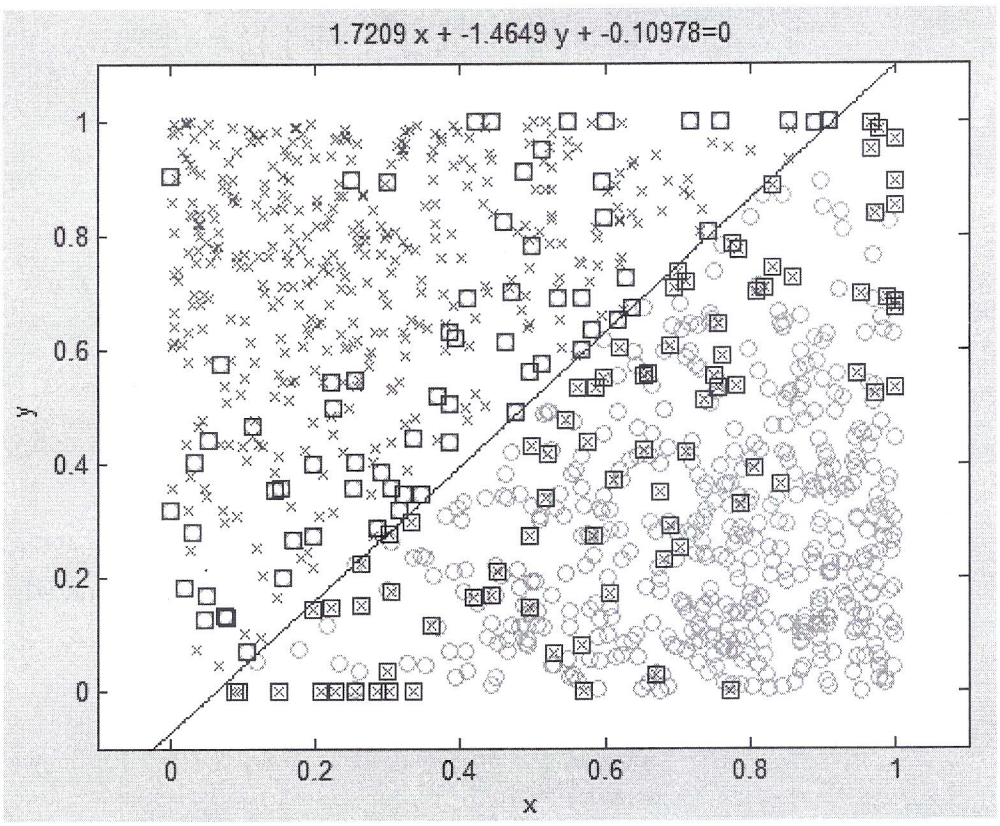
Ein = 0.11



2°

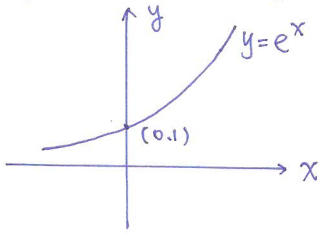
w = [-0.1098 1.7209 -1.4649]

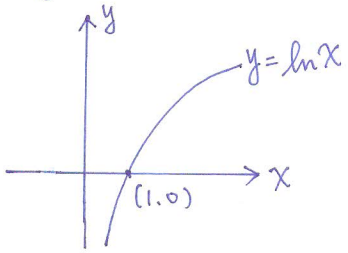
Eout = 0.149



4.6

(1) Some properties about $y=e^x$ and $y=\ln x$





$\Rightarrow e^x > 0$

$e^x > 1 \quad , \forall\ x > 0$

$\Rightarrow \ln x > 0 \quad , \forall\ x > 1$

$a > b \Rightarrow \ln a > \ln b$

1° $y_n = +1$

$\Rightarrow \vec{w} \cdot \vec{x}_n > 0$

$E^{(n)}(\vec{w}) = \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right]$. $\because e^{-y_n(\vec{w}\cdot\vec{x}_n)} > 0$ and $\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right] > 1$

> 0 $\therefore \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right] > 0$

$\Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w}) > 0$

$\Rightarrow \vec{w} \cdot \vec{x}_n < 0$

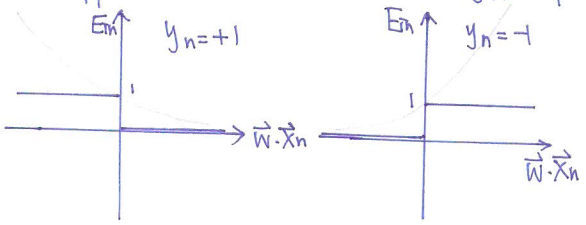
$E^{(n)}(\vec{w}) = \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right]$ $\because -y_n(\vec{w}\cdot\vec{x}_n) > 0 \Rightarrow e^{-y_n(\vec{w}\cdot\vec{x}_n)} > 1$

$\Rightarrow 1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)} > 2$

$> \ln 2$ $\Rightarrow \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right] > \ln 2$

$\Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w}) > 1$

Therefore, $\frac{1}{\ln 2} E^{(n)}(\vec{w}) \geq E_{in}(\vec{w}) \Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w})$ is an upper bound in case of $y_n = +1$



2° $y_n = -1$

$\Rightarrow \vec{w} \cdot \vec{x}_n < 0$

$E^{(n)}(\vec{w}) = \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right] > \ln 1 = 0$

$\Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w}) > 0$

$\Rightarrow \vec{w} \cdot \vec{x}_n > 0$

$E^{(n)}(\vec{w}) = \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right] > \ln 2$

$\Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w}) > 1$

Therefore, $\frac{1}{\ln 2} E^{(n)}(\vec{w}) \geq E_{in}(\vec{w}) \Rightarrow \frac{1}{\ln 2} E^{(n)}(\vec{w})$ is an upper bound in case of $y_n = -1$

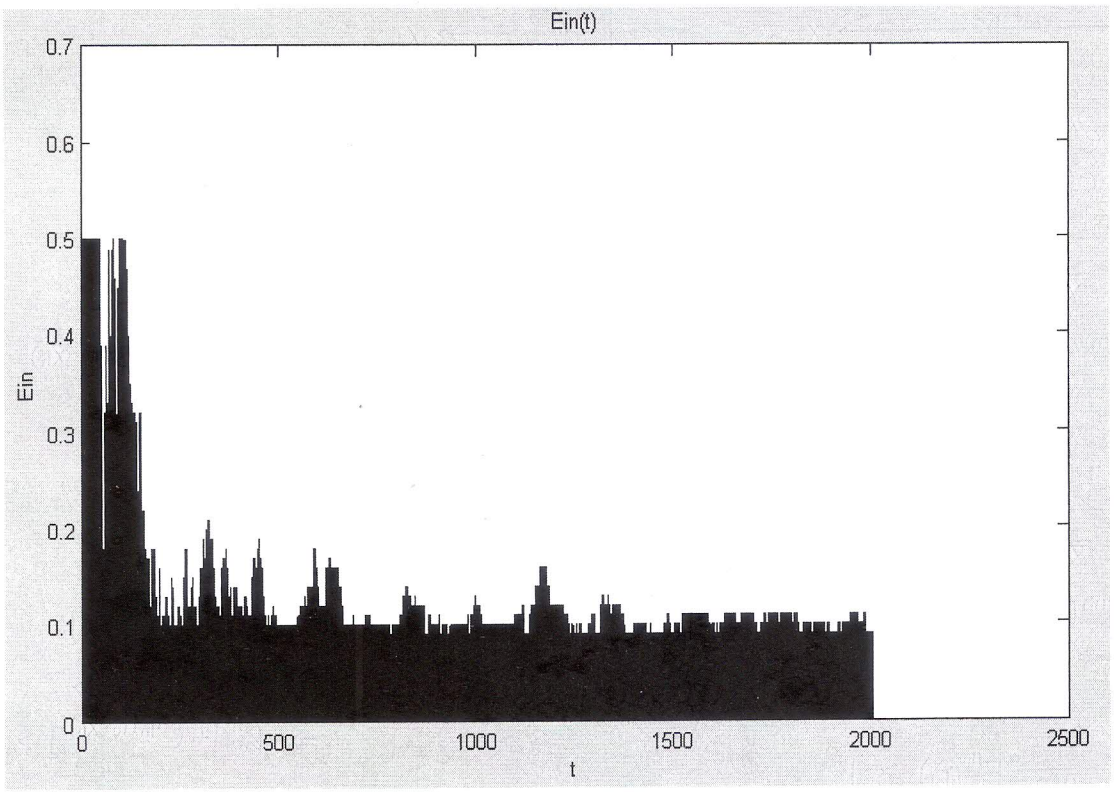
By 1°, 2°, $\frac{1}{\ln 2} E^{(n)}(\vec{w})$ is an upper bound of $[\![\text{sign}(\vec{w}\cdot\vec{x}_n) \neq y_n]\!]$ for any $\vec{w}$.

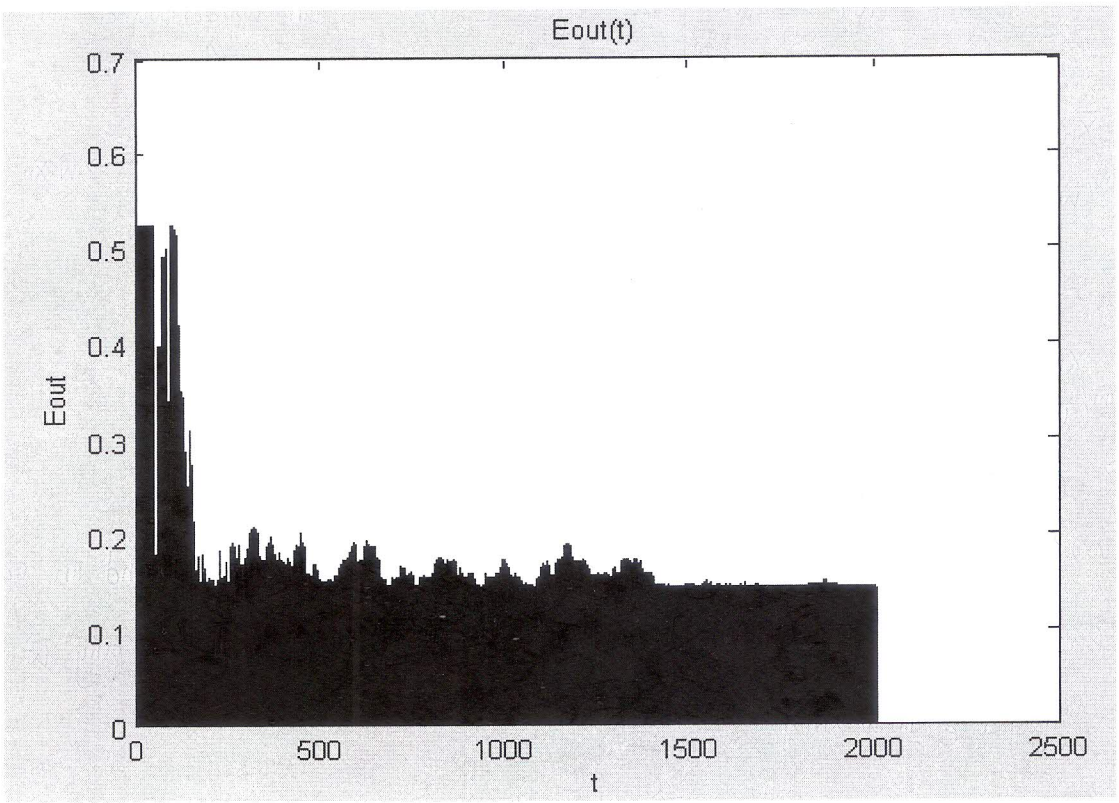
(2) $E^{(n)}(\vec{w}) = \ln\left[1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right]$

$\vec{w} = [w_0, w_1, w_2, \cdots, w_d]^T$, $\vec{x}_n = [x_0, x_1, x_2, \cdots, x_d]^T$

$$\nabla E^{(n)}(\vec{w}) = \begin{bmatrix} \frac{\partial}{\partial w_0} E^{(n)}(\vec{w}) \\ \frac{\partial}{\partial w_1} E^{(n)}(\vec{w}) \\ \vdots \\ \frac{\partial}{\partial w_d} E^{(n)}(\vec{w}) \end{bmatrix} = \left[\frac{1}{1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}}\right]\left[e^{-y_n(\vec{w}\cdot\vec{x}_n)}\right]\begin{bmatrix} -y_n x_1 \\ -y_n x_2 \\ \vdots \\ -y_n x_d \end{bmatrix} = \frac{-y_n \cdot e^{-y_n(\vec{w}\cdot\vec{x}_n)}}{1 + e^{-y_n(\vec{w}\cdot\vec{x}_n)}} \vec{x}_n$$

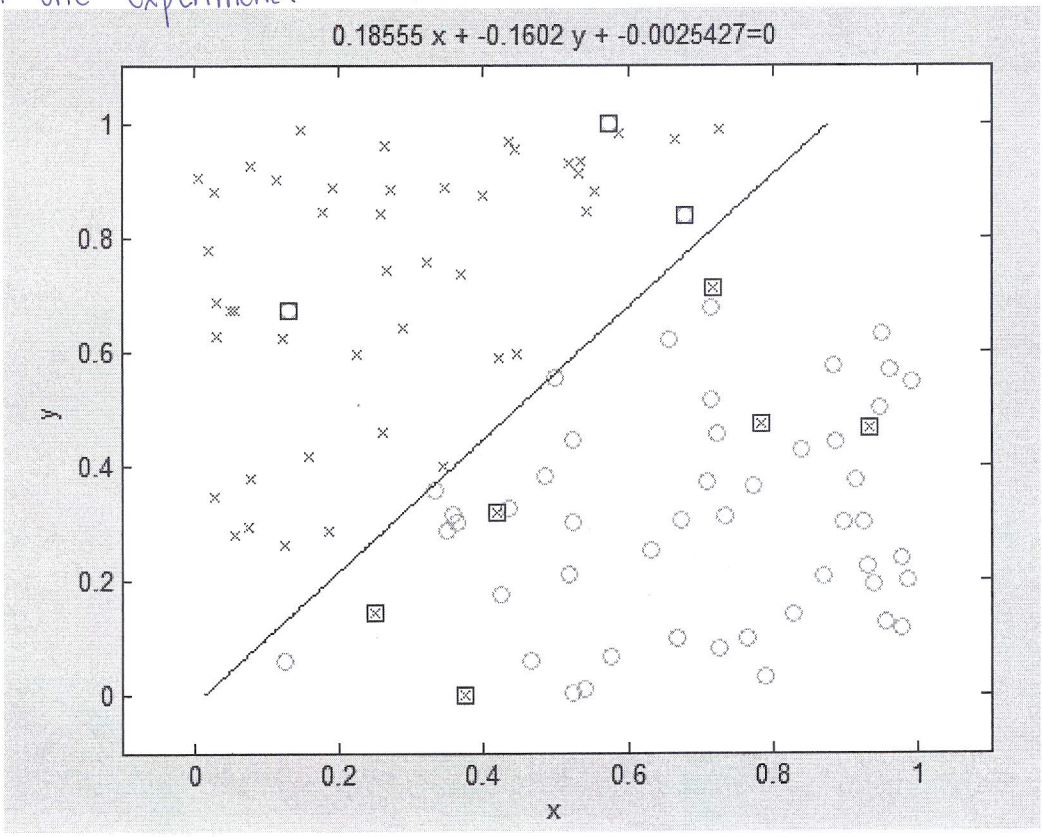
(3) In most of cases using the stochastic gradient descent algorithm, Ein is quite large before about 100 iterations. And after more iterations to update the hypothesis, Ein seems to tend to approach 0.1 with little oscillations.

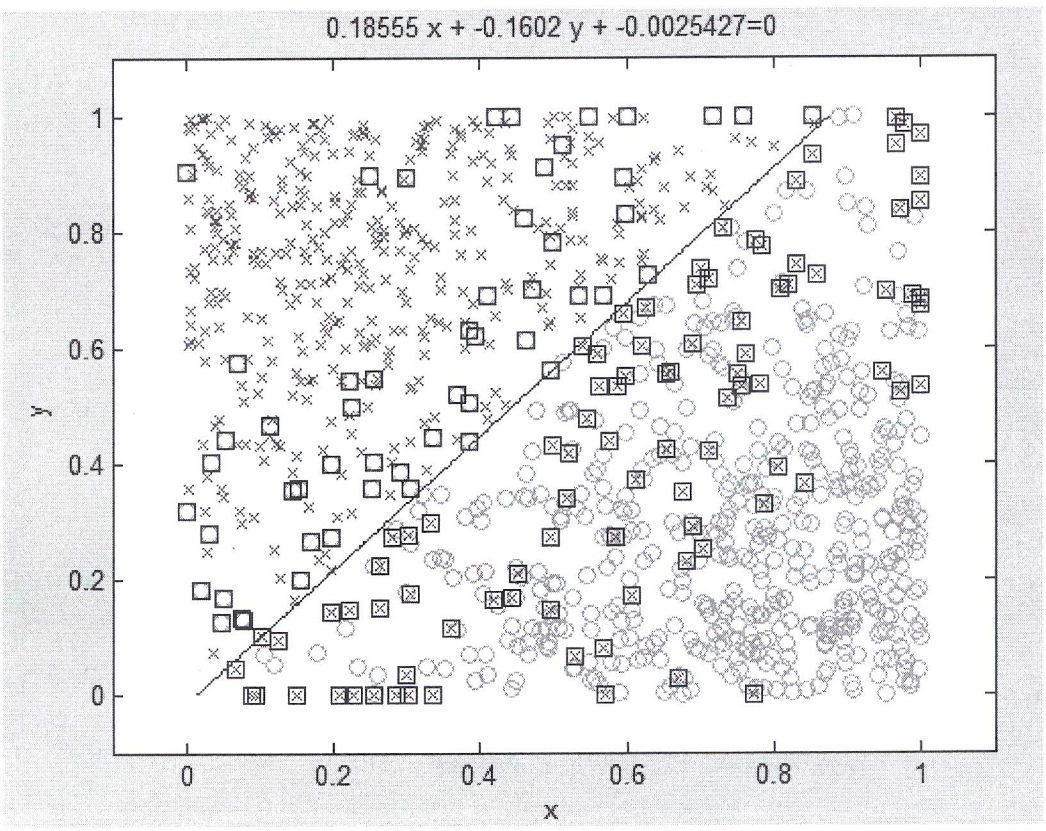




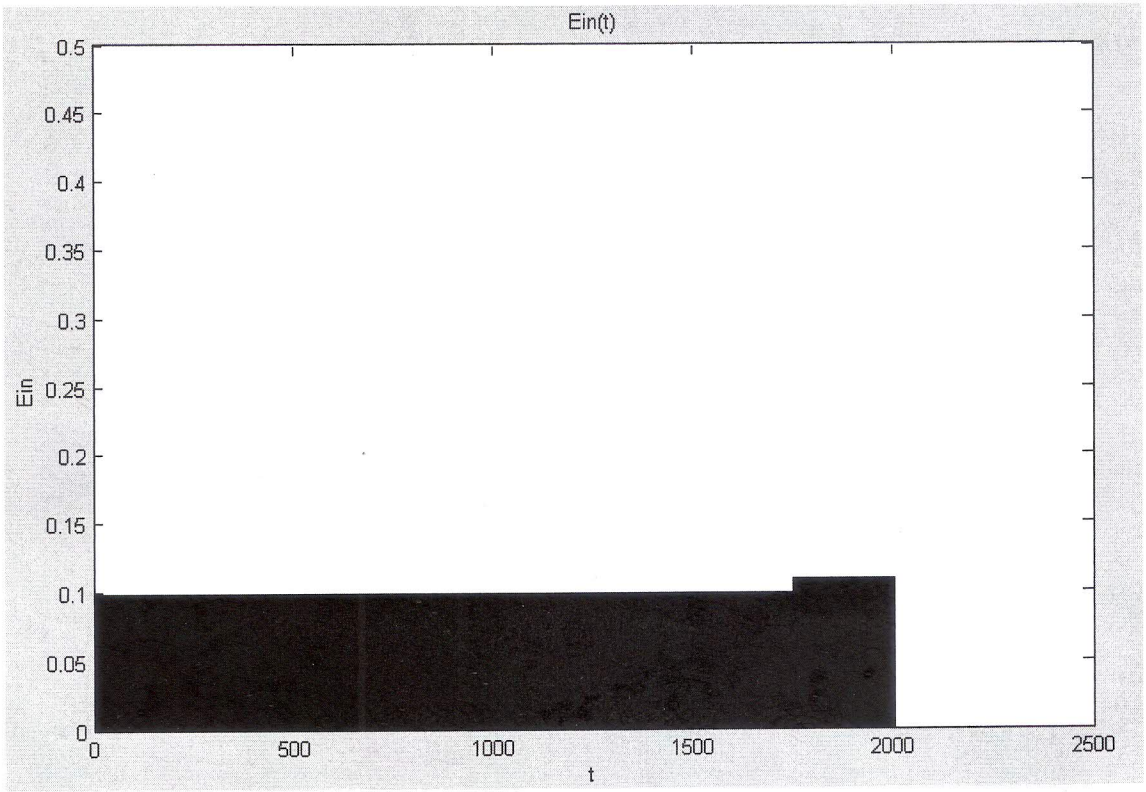
Eout is quite similar to Ein. Eout is quite large in the begining and it becomes more stable to be around 0.15 after more iterations.

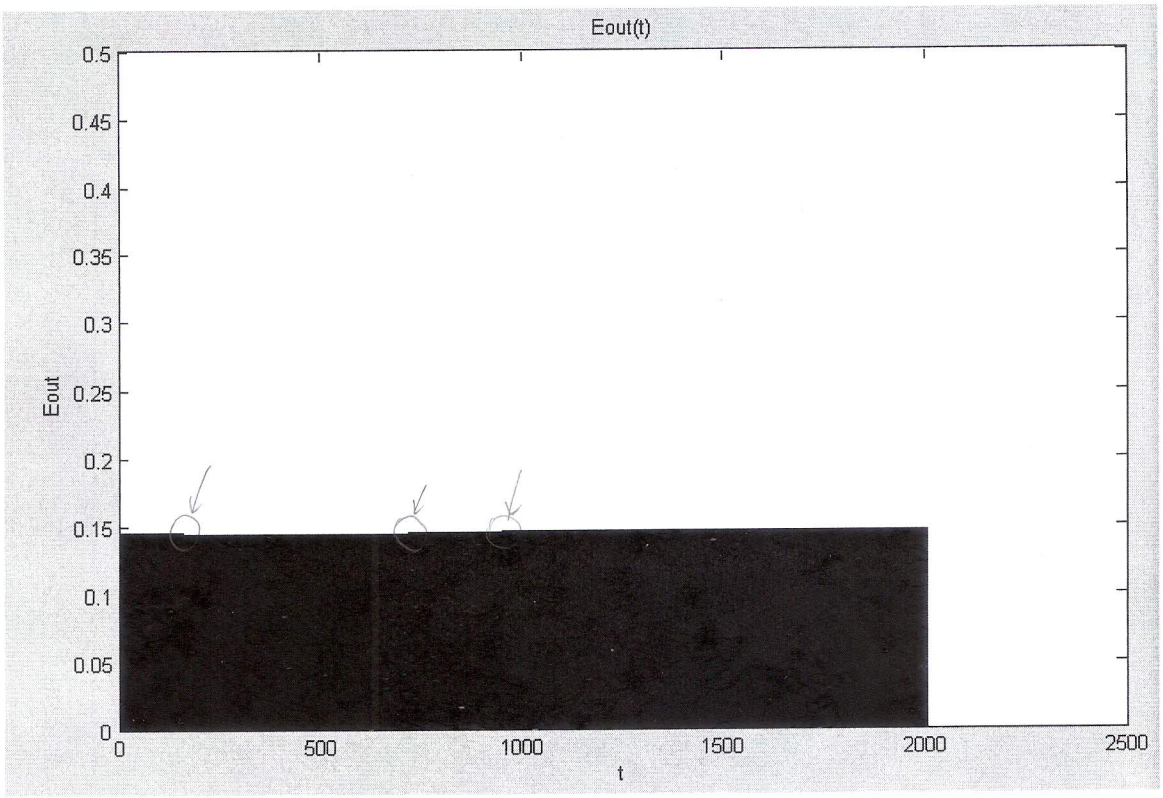
⇒ The following is a final hypothesis performing in training data and testing data from one experiment.





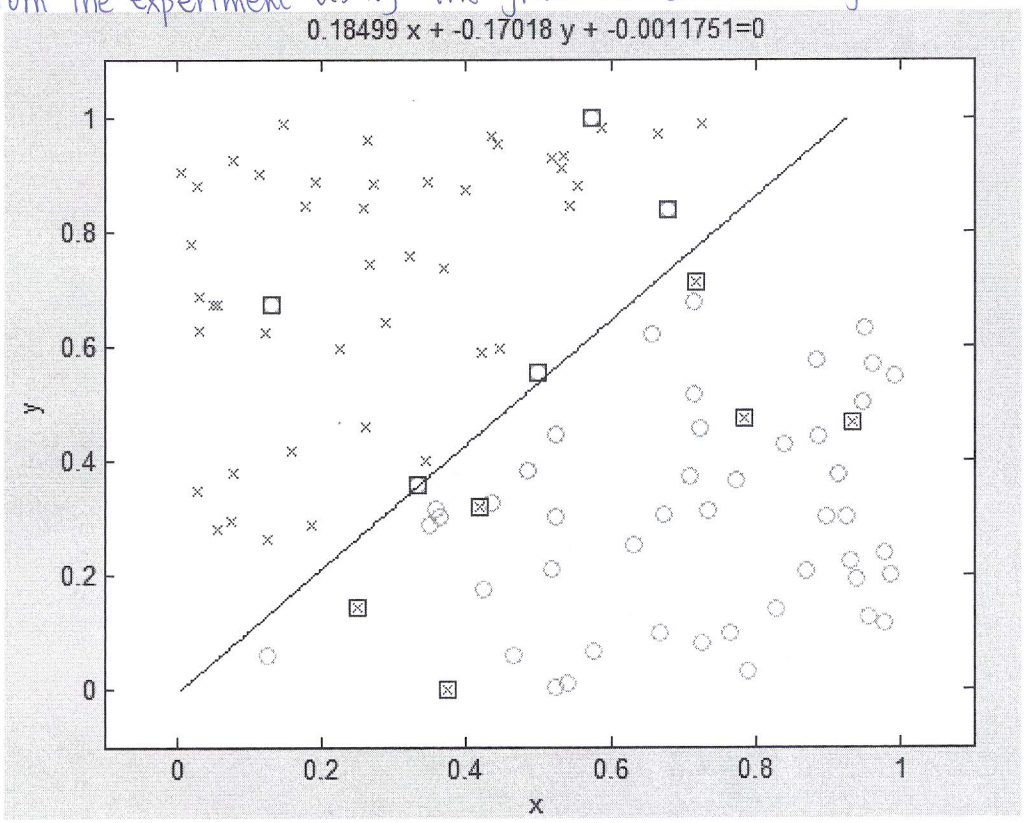
(4) In the experiment using the gradient descent algorithm, Ein is down to 0.1 after the first update. It's different from the Ein using the stochastic algorithm. The Ein using the stochastic one is quite large and needs more iterations to descend its value.

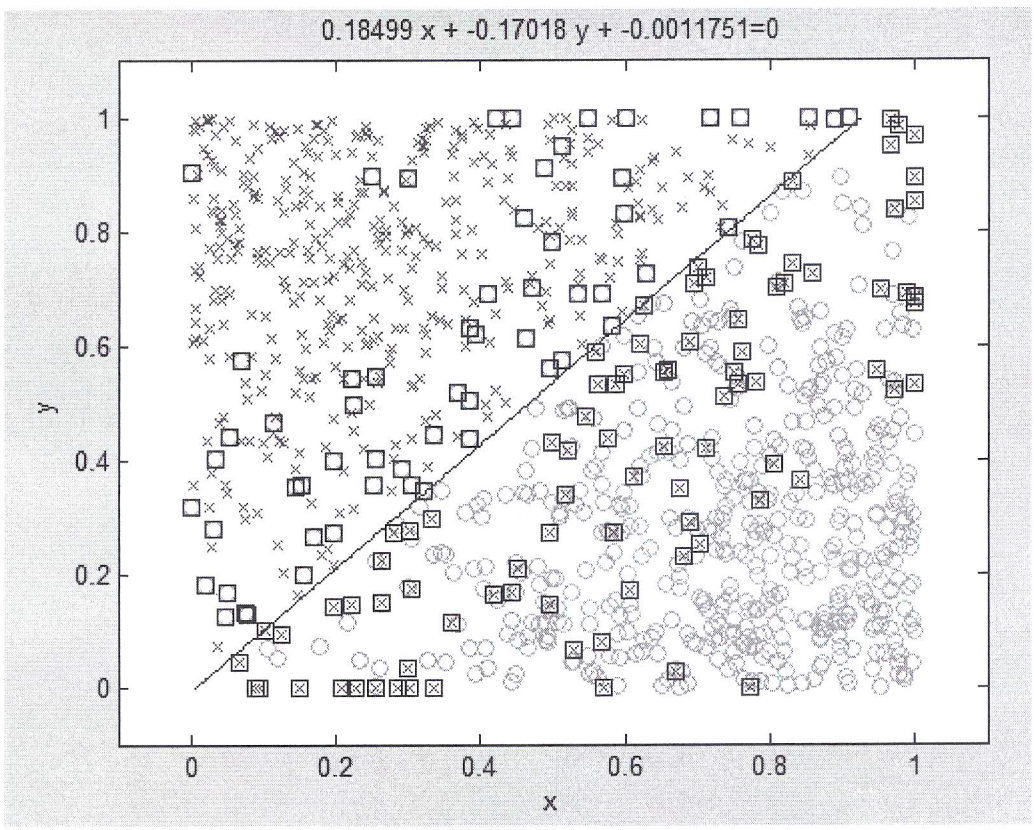




The Eout using the gradient descent algorithm is below 0.15 in the first iteration. Eout seems to be very stable, it only changed three times in 2000 iterations.

⇒ The following is the final hypothesis performing in training data and testing data from the experiment using the gradient descent algorithm.





Homework #5

TA in charge: Yu-Cheng Chou

RELEASE DATE: 11/08/2010

DUE DATE: 11/22/2010, 4:00 pm IN CLASS

TA SESSION: 11/18/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

5.1 Low-Order Transforms: Decision Stump

(1) (5%) Do the first part in Exercise 3.9 of LFD (prove the d_{VC} of a single transform).

(2) (5%) Do the second part in Exercise 3.9 of LFD (prove the d_{VC} of the union).

5.2 Data-dependent Transforms

A Transformer thinks the following procedures would work well in getting a low E_{out} from any two-dimensional data sets. Please point out if there are any potential caveats in the procedures:

(1) (5%) Use the feature transform

$$\Phi(\mathbf{x}) = \begin{cases} (\underbrace{0, \cdots, 0}_{n-1}, 1, 0, \cdots) & \text{if } \mathbf{x} = \mathbf{x}_n \\ (0, \cdots, 0, 0, 0, \cdots) & \text{otherwise .} \end{cases}$$

before running PLA.

(2) (5%) Use the feature transform $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_N(\mathbf{x}))$ with

$$\phi_n(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2\sigma^2}\right)$$

and some very small σ^2 before running PLA.
(Note: You can use the fact that if $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ are all different, the matrix $\mathrm{A} = [a_{mn}]$ with $a_{mn} = \phi_m(\mathbf{x}_n)$ is always invertible.)

5.3 Regularization and Weight Decay

Consider the regularized linear regression formulation

$$\min_{\mathbf{w}} \quad \lambda \mathbf{w} \cdot \mathbf{w} + \frac{1}{N} \sum_{n=1}^{N} \Big(y_n - \big(\mathbf{w} \cdot \mathbf{x}_n \big) \Big)^2.$$

with some $\lambda > 0$.

(1) (5%) Let $\mathbf{w}_{\text{lin}}$ be the optimal solution for the plain-vanilla linear regression and $\mathbf{w}_{\text{reg}}(\lambda)$ be the optimal solution for the formulation above. Prove that $\|\mathbf{w}_{\text{reg}}(\lambda)\| \leq \|\mathbf{w}_{\text{lin}}\|$ for any $\lambda > 0$.
(Note: This is one origin of the name "weight decay.")

Next, consider a more general formulation of regularized learning:

$$\min_{\mathbf{w}} \quad \tilde{E}(\mathbf{w}) = \lambda \mathbf{w} \cdot \mathbf{w} + E_{\text{in}}(\mathbf{w}).$$

with some $\lambda \geq 0$, where $\lambda = 0$ corresponds to not using regularization.

(2) (5%) Let $\mathbf{w}_{\text{reg}}(\lambda)$ be the optimal solution for the formulation above. Prove that $\|\mathbf{w}_{\text{reg}}(\lambda)\|$ is a non-increasing function of λ for $\lambda \geq 0$.

(3) (5%) Assume that E_{in} is differentiable and use gradient descent to minimize $\tilde{E}$:

$$\mathbf{w}^{new} \leftarrow \mathbf{w}^{old} - \eta \nabla \tilde{E}(\mathbf{w}).$$

Show that the update rule above is the same as

$$\mathbf{w}^{new} \leftarrow (1 - 2\eta\lambda)\mathbf{w}^{old} - \eta \nabla E_{\text{in}}(\mathbf{w}).$$

(Note: This is another origin of the name "weight decay": $\mathbf{w}^{old}$ decays before being updated by the gradient of E_{in}.)

5.4 Regularization and Virtual Examples

(1) (5%) Consider the regularized linear regression formulation in Problem 5.3. Let

$$\tilde{\mathbf{x}}_i = (\underbrace{0, \cdots, 0}_{i-1}, \sqrt{N\lambda}, \underbrace{0, \cdots, 0}_{d-i+1})$$

and $\tilde{y}_i = 0$. Prove that solving the formulation is equivalent to applying the plain-vanilla linear regression on $\{(\mathbf{x}_n, y_n)\}_{n=1}^{N} \cup \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i=1}^{d+1}$.
(Note: The pairs $(\tilde{\mathbf{x}}_i, \tilde{y}_i)$ are called virtual examples. The results above shows that regularization can be viewed as using "additional" examples to guide the learning process.)

(2) (5%) Recall that when $\mathrm{X}^T\mathrm{X}$ is invertible, the solution of the plain-vanilla linear regression is $(\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{Y}$. Explain how you can use the result from (1) to easily derive the solution of the regularized linear regression formulation.

5.5 Non-Falsifiability

(1) (3%) Do Exercise 5.2(b) of LFD.

(2) (3%) Do Exercise 5.2(c) of LFD.

(3) (3%) Do Exercise 5.2(d) of LFD.

(4) (3%) Do Exercise 5.2(e) of LFD.

(5) (3%) Do Exercise 5.2(f) of LFD.

5.6 Regularized Linear Regression (*)

(1) (10%) Implement the linear regression algorithm in Problem 4.5. Run the algorithm on the following data set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_6_train.dat`

and the following set for testing

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_6_test.dat`

Let

$$g(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x}).$$

What is $E_{\text{in}}(g)$ in terms of the 0/1 loss (classification)? How about $E_{\text{out}}(g)$?

Plot the training examples $(\mathbf{x}_n, y_n)$ and the decision boundary $\mathbf{w} \cdot \mathbf{x} = 0$ in the same figure. Use different symbols to distinguish examples with different y_n. Briefly state your findings.

Please check the course policy carefully and do not use sophisticated packages in your solution. You ***can*** *use standard matrix multiplication and inversion routines.*

(2) (10%) Split the given training examples in

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_6_train.dat`

to 120 "base" examples (the first 120) and 80 "validation" ones (the last 80).

Ideally, you should randomly do the 120/80 *split. Because the given examples are already randomly permuted, however, we would use a fixed split for the purpose of this problem.*

Implement an algorithm that solves the regularized linear regression formulation in Problem 5.3. Run the algorithm on the 120 base examples using $\log_{10} \lambda = \{2, 1, 0, -1, \ldots, -8, -9, -10\}$. Let

$$g_\lambda(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{reg}}(\lambda) \cdot \mathbf{x}).$$

Validate g_λ with the 80 validation examples and test it with the test examples in

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_6_test.dat`

Plot $E_{\text{base}}(g_\lambda)$, $E_{\text{val}}(g_\lambda)$, $E_{\text{out}}(g_\lambda)$ on the same figure as a function of $\log_{10} \lambda$, where the base training error E_{base} is E_{in} evaluated on only the 120 base examples. Briefly state your findings.

5.7 Large-Margin Perceptron Classification (*)

(1) (10%) Implement the perceptron learning algorithm in Problem 1.3. Run the algorithm on the following data set for training (until E_{in} reaches 0):

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_7_train.dat`

and the following set for testing

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_7_test.dat`

Let $\mathbf{w} = (w_0, \mathbf{u})$ as the solution from PLA and $g(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$. Record the following two items:

- the "thickness" of $\mathbf{w}$: $\min_n \left\{ y_n (\frac{\mathbf{w}}{\|\mathbf{u}\|} \cdot \mathbf{x}_n) \right\}$
- the out-of-sample error E_{out} of g

Repeat the experiment over 100 runs. Plot a histogram of the thickness and another histogram of the out-of-sample error. Briefly state your findings.

(2) (10%) Implement the large-margin perceptron formulation below:

$$\begin{aligned} \min_{\mathbf{u},b} \quad & \frac{1}{2}\mathbf{u} \cdot \mathbf{u} \\ \text{subject to} \quad & y_n (\mathbf{u} \cdot \mathbf{x}_n + b) \geq 1 \text{ for } n = 1, 2, \ldots, N. \end{aligned}$$

Run the algorithm on the following data set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_7_train.dat`

and the following set for testing

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw5_7_test.dat`

Let $\mathbf{w} = (b, \mathbf{u})$ as the solution from the formulation and $g(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$. Report the following two items:

- the "thickness" of $\mathbf{w}$: $\min_n \left\{ y_n (\frac{\mathbf{w}}{\|\mathbf{u}\|} \cdot \mathbf{x}_n) \right\}$
- the out-of-sample error E_{out} of g

Compare the numbers with the histograms that you get from PLA. Briefly state your findings.

(*Note: You can use any general-purpose packages for quadratic programming to solve this problem, but you* ***cannot*** *use any SVM-specific packages.*)

ML HW #5 R98922081 徐伍文

94

5.1.

(1) From Lemma 2.4 and 2.5 : The VC dimension of the perceptron is exactly d+1

∴ For feature transform which maps input vector in $\mathbb{R}^d$ to $\mathbb{R}^1$

$\Rightarrow$ $d_{vc}(\mathcal{H}_k) = 1+1 = 2$

(2) We can find out that PLA works like "rays" or "decision stump" when it's in one dimension. Therefore, when input number is N, we can get that $m_{\mathcal{H}}(N) = 2N$. If there is a number D such that when $N = D$, it's impossible for this low-dimensional feature transforms method to shatter, then it means $(2D)d < 2^D$. And we can choose $D > 2d$ that makes the inequality above satisfied. That is to say, Once input number exceeds double of its dimension, the low-dimensional feature transforms couldn't shatter. So we have to choose our input data number $N \le 2d$.

$\Rightarrow$ $d_{vc}(\bigcup_{k=1}^{d} \mathcal{H}_k) \le \log_2(|\bigcup_{k=1}^{d} \mathcal{H}_k|) = \log_2(2(N)d)$, $\because N \le 2d$

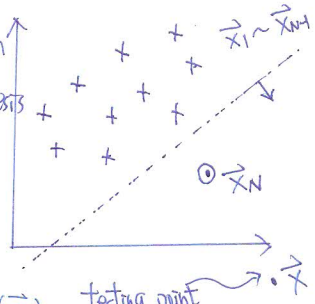
$\le \log(2(2d)d) = \log 4d^2 = 2(\log_2 d + 1)$

5.2

(1) In this transform, any testing data maps to $\vec{0}$ unless the testing point happens to be the same with one of the training data. And for some hypothesis which works in this N-dimension PLA will classify these testing points which are mapped to $\vec{0}$ into the same class even though they are not with the same label in fact.

(2) The feature transform function is an gaussian with some very small σ^{-2}. If we assume that σ^2 is so small and these N training points are with enough long distance for each other so that $\phi_m(\vec{x}_n) = \exp\left(-\frac{\|\vec{x}_n - \vec{x}_m\|^2}{2\sigma^2}\right) \longrightarrow 0$, then $A \sim I$ and $\vec{w}^T A = \vec{y}^T$

$\Rightarrow \vec{w}^T = A^{-1}\vec{y}^T \approx \vec{y}^T \Rightarrow \vec{w} \approx \vec{y}$. And for some testing point $\vec{x}$: $h(\vec{x}) = \text{sign}(\vec{w} \cdot \Phi(\vec{x}))$. It may have some problems that if the training data is biased that points with some label are much more than the other one. This label points may dominate the result of $\text{sign}(\vec{w} \cdot \Phi(\vec{x}))$. For example, if $\vec{x}_1 \sim \vec{x}_{N-1}$ in the training data which are labeled with "-1" and only $\vec{x}_N$ is labeled with "+1", then the hypothesis in N-dimension $\vec{w}$ approximates to be $\vec{y}$, that is $\vec{w} \approx [-1, -1, \cdots, -1, 1]^T$.



For some testing point $\vec{x}$ which is in the same side with $\vec{x}_N$ but with long distance with $\vec{x}_N$. Even though $\phi_N(\vec{x})$ is greater than $\phi_1(\vec{x}) \sim \phi_{N-1}(\vec{x})$, for $\Phi(\vec{x}) = (\phi_1(\vec{x}), \phi_2(\vec{x}), \cdots, \phi_{N-1}(\vec{x}), \phi_N(\vec{x}))$, $\Phi(\vec{x}) \cdot \vec{w} = \phi_N(\vec{x}) - \phi_1(\vec{x}) - \phi_2(\vec{x}) - \cdots - \phi_{N-1}(\vec{x})$ may be negative because $\phi_N(\vec{x})$ is not big enough and there are too many negative ones.

5.3

(1) $\vec{w}_{lin} = (X^TX)^{-1}X^T\vec{y}$ ($X: N\times(d+1)$, $X^TX: (d+1)\times(d+1)$, $\vec{y}: N\times 1$, $\vec{X}^T\vec{y}: (d+1)\times 1$)

Let $A = X^TX$

$\because A^T = (X^TX)^T = X^TX = A$

$\Rightarrow$ A: symmetric $\Leftrightarrow \exists P$: orthogonal matrix (i.e. $P^TP = I$, $P^{-1} = P^T$)

$\nexists A = PDP^T$

$$= [\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{d+1}] \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \bigcirc \\ & & \ddots & \\ & \bigcirc & & \alpha_{d+1} \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_{d+1} \end{bmatrix}$$

where α_i is the eigenvalue of A

and $\vec{v}_i$ is the eigenvector with respect to α_i

$$\Rightarrow A^{-1} = (PDP^T)^{-1} = (P^T)^{-1}D^{-1}P^{-1} = (P^{-1})^TD^{-1}P^{-1} = (P^T)^TD^{-1}P^T = PD^{-1}P^T ,$$

$$= [\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{d+1}] \begin{bmatrix} \frac{1}{\alpha_1} & & & \\ & \frac{1}{\alpha_2} & & \bigcirc \\ & & \ddots & \\ & \bigcirc & & \frac{1}{\alpha_{d+1}} \end{bmatrix} \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \\ \vdots \\ \vec{v}_{d+1} \end{bmatrix}$$

$$\Rightarrow [A^{-1}]^T[A^{-1}] = [PD^{-1}P^T]^T[PD^{-1}P^T] = PD^{-1}\cancel{P^TP}D^{-1}P^T = P[D^{-1}]^2P^T$$

Let $\vec{z} = X^T\vec{y}$, $\vec{z}: (d+1)\times 1$

$$\Rightarrow \|\vec{w}_{lin}\|^2 = [(X^TX)^{-1}X^T\vec{y}]^T[(X^TX)^{-1}X^T\vec{y}]$$

$$= [A^{-1}\vec{z}]^T[A^{-1}\vec{z}]$$

$$= \vec{z}^T[A^{-1}]^T[A^{-1}]\vec{z}$$

$$= \vec{z}^TP[D^{-1}]^2P^T\vec{z} = \sum_{i=1}^{d+1} \frac{1}{(\alpha_i)^2}(\vec{z}^T\vec{v}_i)^2$$

$\vec{w}_{reg}(\lambda) = (X^TX + \lambda I)^{-1}X^T\vec{y}$, for $\lambda > 0$

Let $A = X^TX$, A: symmetric $\Leftrightarrow \exists P$: orthogonal matrix $\nexists A = PDP^T$

$B = X^TX + \lambda I$, B: symmetric and $B = PD'P^T$, $P = [\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{d+1}]$

And $D = \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \bigcirc \\ & & \ddots & \\ & \bigcirc & & \alpha_{d+1} \end{bmatrix}$, $D' = \begin{bmatrix} \beta_1 & & & \\ & \beta_2 & & \bigcirc \\ & & \ddots & \\ & \bigcirc & & \beta_{d+1} \end{bmatrix} = \begin{bmatrix} \alpha_1+\lambda & & & \\ & \alpha_2+\lambda & & \bigcirc \\ & & \ddots & \\ & \bigcirc & & \alpha_{d+1}+\lambda \end{bmatrix}$

Where α_i is the eigenvalue of A

β_i is the eigenvalue of B and $\beta_i = \alpha_i + \lambda$

$\vec{v}_i$ is the eigenvector with respect to α_i and β_i

$$\therefore B^{-1} = (PD'P^T)^{-1} = P[D']^{-1}P^T$$

$$= P\begin{bmatrix} \frac{1}{\beta_1} & & \\ & \frac{1}{\beta_2} & \\ & & \ddots \\ & & & \frac{1}{\beta_{d+1}} \end{bmatrix} P^T = P\begin{bmatrix} \frac{1}{\alpha_1+\lambda} & & \\ & \frac{1}{\alpha_2+\lambda} & \\ & & \ddots \\ & & & \frac{1}{\alpha_{d+1}+\lambda} \end{bmatrix} P^T$$

$$\Rightarrow [B^{-1}]^T[B^{-1}] = P[D']^{-1}P^TP[D']^{-1}P^T = P[D'^{-1}]^2P^T$$

Let $\vec{z} = X^T\vec{y}$

$$\Rightarrow \|\vec{w}_{reg}(\lambda)\|^2 = [(X^TX + \lambda I)^{-1}X^T\vec{y}]^T[(X^TX + \lambda I)^{-1}X^T\vec{y}]$$

$$= [B^{-1}\vec{z}]^T[B^{-1}\vec{z}]$$

$$= \vec{z}^T[B^{-1}]^T[B^{-1}]\vec{z}$$

$$= \vec{z}^TP[D'^{-1}]^2P^T\vec{z} = \sum_{i=1}^{d+1} \frac{1}{(\alpha_i+\lambda)^2}(\vec{z}^T\vec{v})^2$$

$\because A = X^TX$: invertible $\therefore$ A : positive definite $\Rightarrow \alpha_i > 0$ for all $i = 1 \sim d+1$

$$\Rightarrow \|\vec{w}_{reg}(\lambda)\|^2 = \sum_{i=1}^{d+1} \frac{1}{(\alpha_i+\lambda)^2}(\vec{z}^T\vec{v}_i)^2 \quad . \ (\lambda \geq 0 \ , \ (\vec{z}^T\vec{v}_i)^2 \geq 0)$$

$$\leq \sum_{i=1}^{d+1} \frac{1}{(\alpha_i)^2}(\vec{z}^T\vec{v}_i)^2 = \|\vec{w}_{lin}\|^2$$

Therefore, $\|\vec{w}_{reg}(\lambda)\| \leq \|\vec{w}_{lin}\|$

(2) From above, $\|\vec{w}_{reg}(\lambda)\| = [\sum_{i=1}^{d+1} \frac{1}{(\alpha_i+\lambda)^2}(\vec{z}^T\vec{v}_i)^2]^{\frac{1}{2}} \quad . (\alpha_i > 0 \ , \forall i)$

For $\lambda_1 < \lambda_2 \Rightarrow \|\vec{w}_{reg}(\lambda_1)\| = [\sum_{i=1}^{d+1} \frac{1}{(\alpha_i+\lambda_1)^2}(\vec{z}^T\vec{v}_i)^2]^{\frac{1}{2}} \quad . (\alpha_i > 0 . \forall i)$

$$\geq [\sum_{i=1}^{d+1} \frac{1}{(\alpha_i+\lambda_2)^2}(\vec{z}^T\vec{v}_i)^2]^{\frac{1}{2}} = \|\vec{w}_{reg}(\lambda_2)\|^2$$

Therefore, $\|\vec{w}_{reg}(\lambda)\|$ is a non-increasing function of λ

(3) $\tilde{E}(\vec{w}) = \lambda\vec{w}\cdot\vec{w} + E_{in}(\vec{w})$

$$\Rightarrow \nabla\tilde{E}(\vec{w}) = 2\lambda\vec{w} + \nabla E_{in}(\vec{w})$$

$$\therefore \vec{w}^{new} \leftarrow \vec{w}^{old} - \eta\nabla\tilde{E}(\vec{w})$$

$$\Rightarrow \vec{w}^{new} \leftarrow \vec{w}^{old} - \eta[2\lambda\vec{w}^{old} + \nabla E_{in}(\vec{w})]$$

$$\Rightarrow \vec{w}^{new} \leftarrow (1-2\eta\lambda)\vec{w}^{old} + \eta\cdot\nabla E_{in}(\vec{w})$$

5.4

(1) Let $\vec{w} = (w_0, w_1, w_2, \cdots, w_d)$

then for plain-vanilla linear regression :

$$\min_{\vec{w}} \frac{1}{N+d+1}\left[\sum_{n=1}^{N}(y_n - (\vec{w}\cdot\vec{x}_n))^2 + \sum_{i=1}^{d+1}(y_i - (\vec{w}\cdot\tilde{x}_i))^2\right]$$

$$\Rightarrow \min_{\vec{w}} \frac{1}{N}\sum_{n=1}^{N}(y_n - (\vec{w}\cdot\vec{x}_n))^2 + \frac{1}{N}\sum_{i=1}^{d+1}(0 - (\sqrt{N\lambda}\cdot w_{i-1}))^2$$

$$\Rightarrow \min_{\vec{w}} \frac{1}{N}\sum_{n=1}^{N}(y_n - (\vec{w}\cdot\vec{x}_n))^2 + \frac{N\lambda}{N}\left(\sum_{i=1}^{d+1} w_{i-1}^2\right)$$

$$\Rightarrow \min_{\vec{w}} \frac{1}{N}\sum_{n=1}^{N}(y_n - (\vec{w}\cdot\vec{x}_n))^2 + \lambda\vec{w}\cdot\vec{w} \longrightarrow \text{regularized linear regression}$$

(2) In plain-vanilla linear regression : $X = \begin{bmatrix}\vec{x}_1\\ \vec{x}_2\\ \vdots\\ \vec{x}_N\end{bmatrix} : N\times(d+1)$ and each $\vec{x}_i : 1\times(d+1)$

Let $X = [\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{d+1}] : N\times(d+1)$ where $\vec{v} : N\times 1$

then $X^TX = \begin{bmatrix}\vec{v}_1^T\\ \vec{v}_2^T\\ \vdots\\ \vec{v}_{d+1}^T\end{bmatrix}[\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_{d+1}] = \begin{bmatrix}\vec{v}_1^T\vec{v}_1 & \vec{v}_1^T\vec{v}_2 & \cdots & \vec{v}_1^T\vec{v}_{d+1}\\ \vec{v}_2^T\vec{v}_1 & \vec{v}_2^T\vec{v}_2 & \cdots & \vec{v}_2^T\vec{v}_{d+1}\\ \vdots & \vdots & \ddots & \vdots\\ \vec{v}_{d+1}^T\vec{v}_1 & \vec{v}_{d+1}^T\vec{v}_2 & \cdots & \vec{v}_{d+1}^T\vec{v}_{d+1}\end{bmatrix} : (d+1)\times(d+1)$

Now, we add the virtual examples into the matrix, and renamed it as A

$$\Rightarrow A = \begin{bmatrix}\vec{x}_1\\ \vdots\\ \vec{x}_N\\ \tilde{x}_1\\ \vdots\\ \tilde{x}_{d+1}\end{bmatrix} : (N+d+1)\times(d+1) \quad \text{and let } \vec{u}_i = [\vec{v}_i; \tilde{x}_i^T] \Rightarrow A = [\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_{d+1}]$$

then $A^TA = \begin{bmatrix}\vec{u}_1^T\\ \vec{u}_2^T\\ \vdots\\ \vec{u}_{d+1}^T\end{bmatrix}[\vec{u}_1\ \vec{u}_2\ \cdots\ \vec{u}_{d+1}] = \begin{bmatrix}\vec{u}_1^T\vec{u}_1 & \vec{u}_1^T\vec{u}_2 & \cdots & \vec{u}_1^T\vec{u}_{d+1}\\ \vec{u}_2^T\vec{u}_1 & \vec{u}_2^T\vec{u}_2 & \cdots & \vec{u}_2^T\vec{u}_{d+1}\\ \vdots & \vdots & \ddots & \vdots\\ \vec{u}_{d+1}^T\vec{u}_1 & \vec{u}_{d+1}^T\vec{u}_2 & \cdots & \vec{u}_{d+1}^T\vec{u}_{d+1}\end{bmatrix}$

$$= \begin{bmatrix}\vec{v}_1^T\vec{v}_1+N\lambda & \vec{v}_1^T\vec{v}_2 & \cdots & \vec{v}_1^T\vec{v}_{d+1}\\ \vec{v}_2^T\vec{v}_1 & \vec{v}_2^T\vec{v}_2+N\lambda & \cdots & \vec{v}_2^T\vec{v}_{d+1}\\ \vdots & \vdots & \ddots & \vdots\\ \vec{v}_{d+1}^T\vec{v}_1 & \vec{v}_{d+1}^T\vec{v}_2 & \cdots & \vec{v}_{d+1}^T\vec{v}_{d+1}+N\lambda\end{bmatrix} = X^TX + N\lambda I$$

And $\because \tilde{y}_i = 0$ for all $i = 1\sim(d+1)$ $\therefore A^T\vec{y}' = X^T\vec{y}$, where $\vec{y}' = [\vec{y}^T; \underbrace{0; \cdots; 0}_{d+1}]^T$

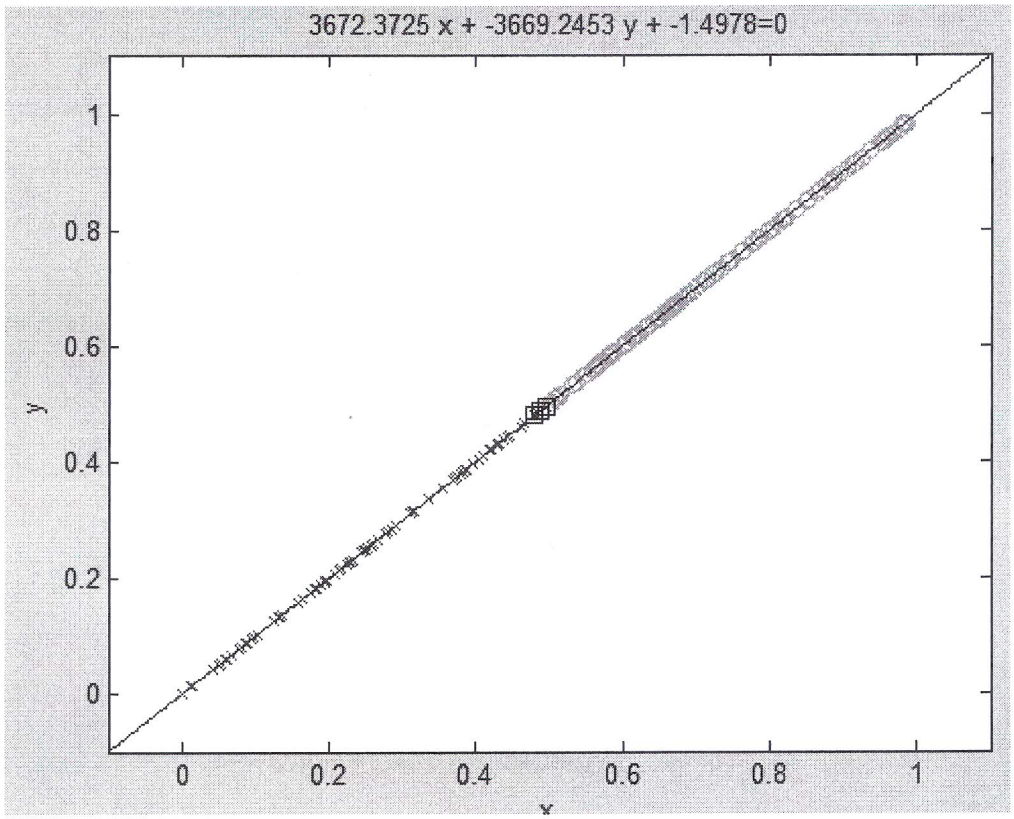
Therefore $\vec{w} = (A^TA)^{-1}A^T\vec{y}' = (X^TX + N\lambda I)^{-1}X^T\vec{y}$

5.5 is in next 4 pages

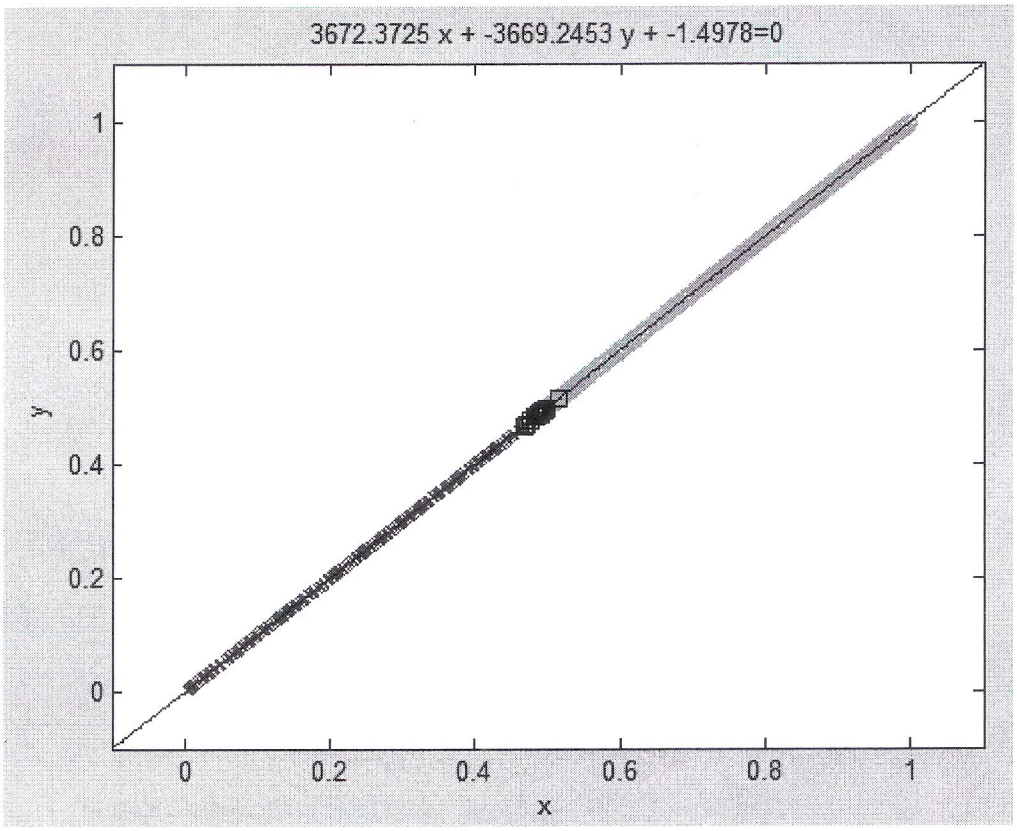
5.6

(1) We can find out that each example is very close to the line of hypothesis. It may have the problem of overfitting.

Ein = 0.015

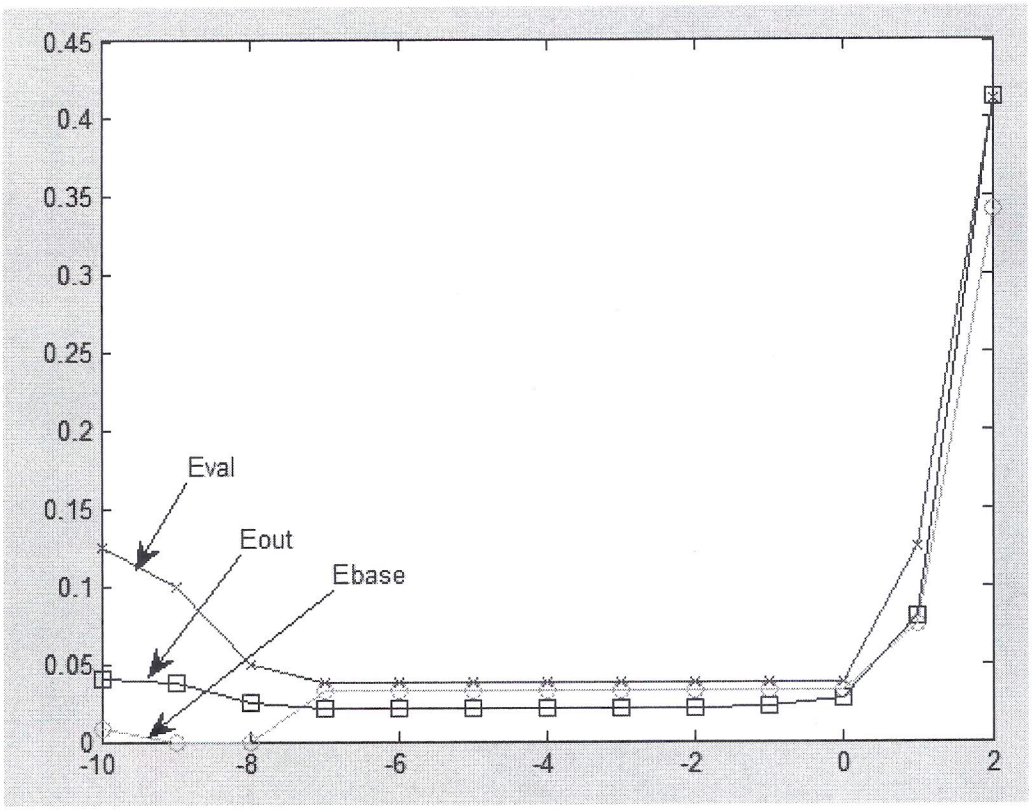


Eout = 0.0200



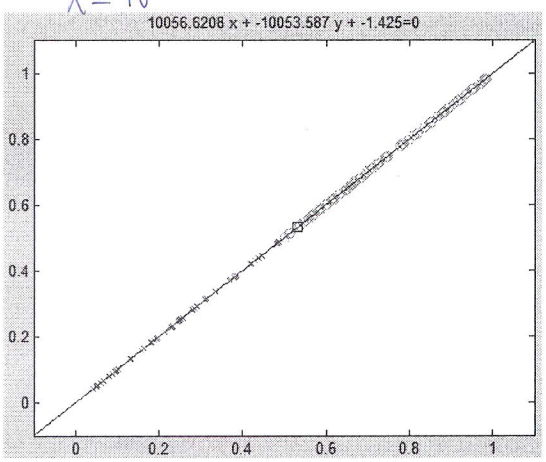
(2) In this problem, we can know that if we can let our validation examples good enough to be close to Eout or have the same tendency with Eout. We can choose a proper λ for regularization to get a better / smaller Eout by observing Eval.

$\log_{10}\lambda$	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2
Ebase	0.00833	0	0	0.03333	0.03333	0.03333	0.03333	0.03333	0.03333	0.03333	0.03333	0.07500	0.34166
Eval	0.12500	0.10000	0.05000	0.03750	0.03750	0.03750	0.03750	0.03750	0.03750	0.03750	0.03750	0.12500	0.41250
Eout	0.04000	0.03800	0.02500	0.02100	0.02100	0.02100	0.02100	0.02100	0.02100	0.02200	0.02800	0.08000	0.41400

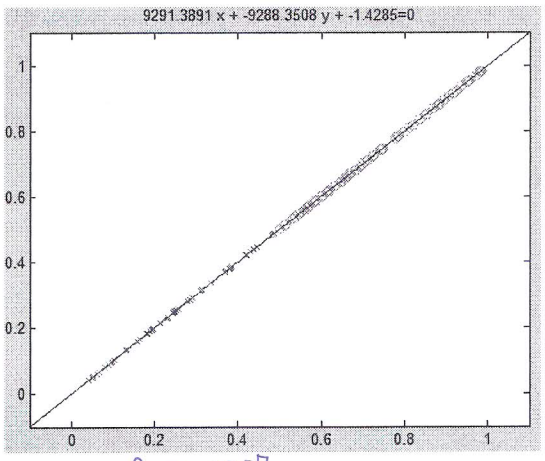


From figure above, we can notice that, when we apply $\lambda = 10^{-10}, 10^{-9}, 10^{-8}$, we can have small Ein, however, Eval and Eout are not as small as their Ein. This may be the problem of overfitting. Therefore, we try to do some more regularization, then apply $\lambda = 10^{-7}, 10^{-6}, \cdots, 10^{1}, 1$. We find out that even though we may have larger Ein than before and the hypothesis seems not to fit the training data as well as previous one, we can have better results (small Eval and Ein) in the testing data. But if we keep increasing λ ($\lambda = 10, 100$ in this case), we may get not only bad Ein (the hypothesis fits the training data badly) but also bad Eval and Eout. This may fall in the situation of underfitting.

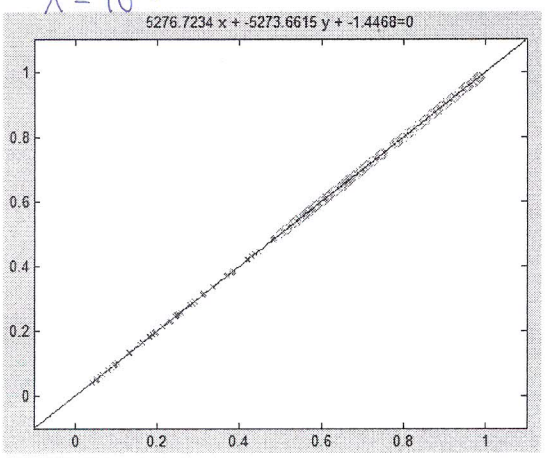
$\lambda = 10^{-10}$



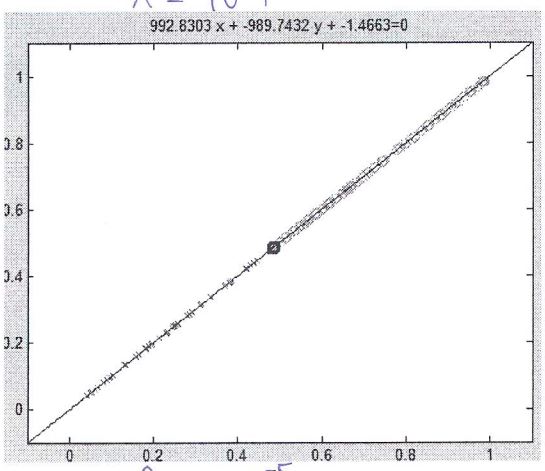
$\lambda = 10^{-9}$



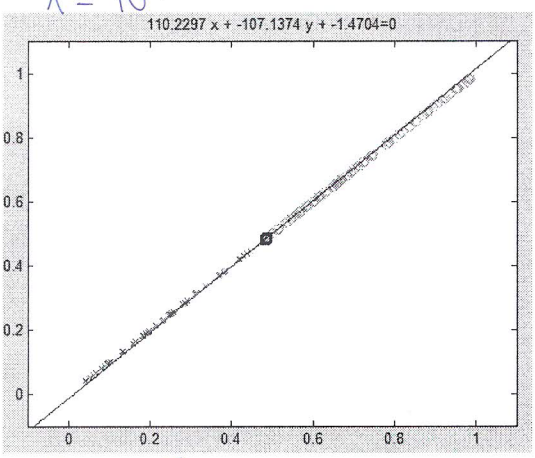
$\lambda = 10^{-8}$



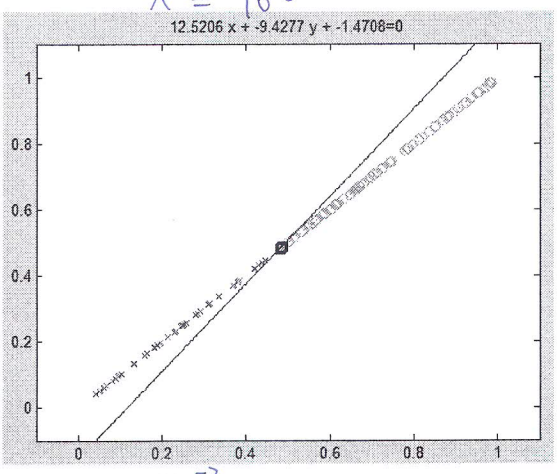
$\lambda = 10^{-7}$



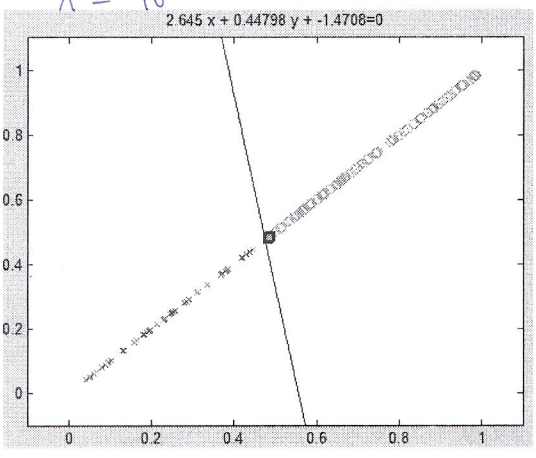
$\lambda = 10^{-6}$



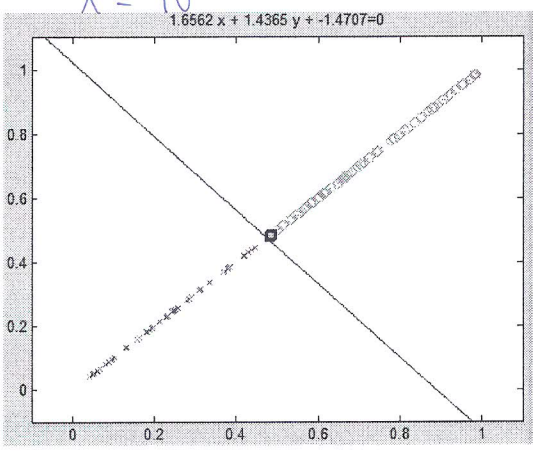
$\lambda = 10^{-5}$



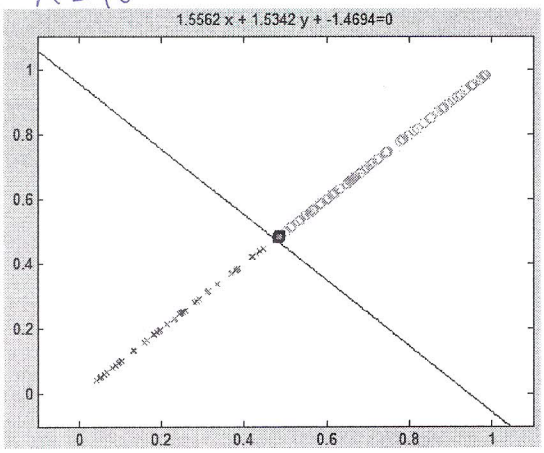
$\lambda = 10^{-4}$



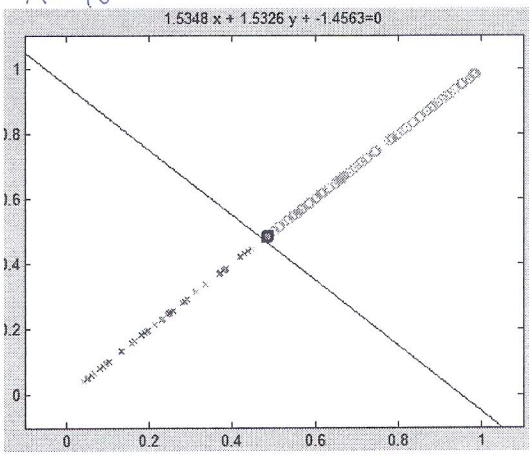
$\lambda = 10^{-3}$



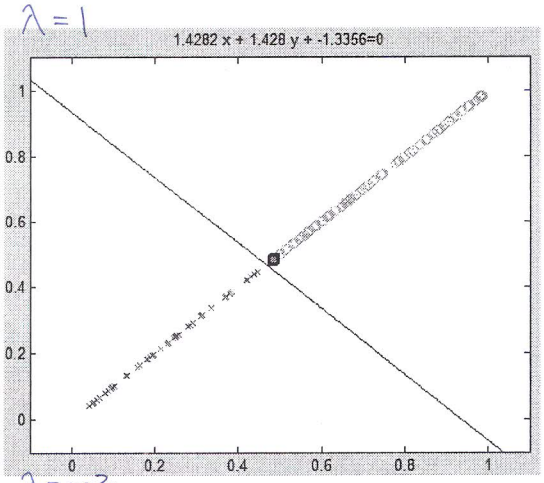
$\lambda = 10^{-2}$



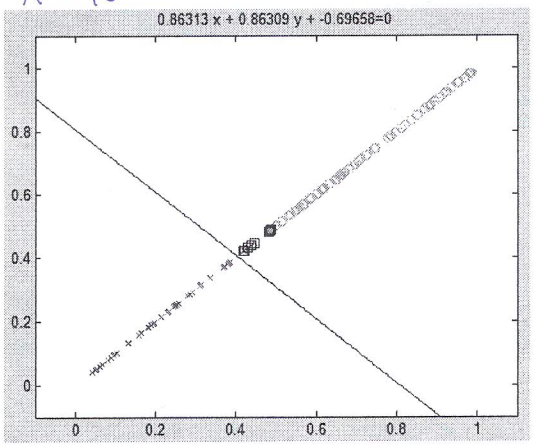
$\lambda = 10^{-1}$



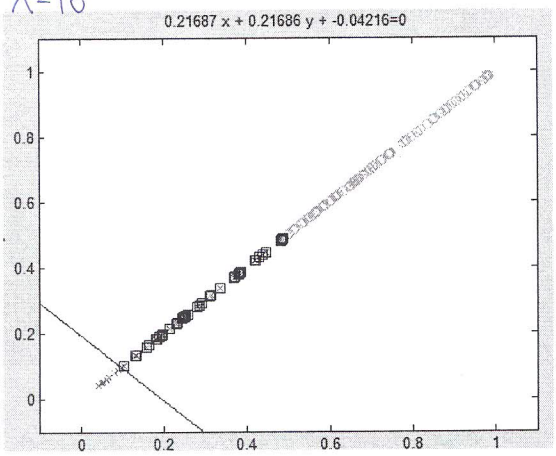
$\lambda = 1$



$\lambda = 10$



$\lambda = 10^{2}$



5.5

(1) 32

(2) 16

(3) $32+16+8+4+2 = 62$

(4) $50 - 62 \times 0.5 = 19$

(5) $p[\text{falsification}] \geq 1 - \frac{m_{\mathcal{H}}(N)}{2^N}$ → 32. ~~1~~ (only one hits the correct result)

-2.

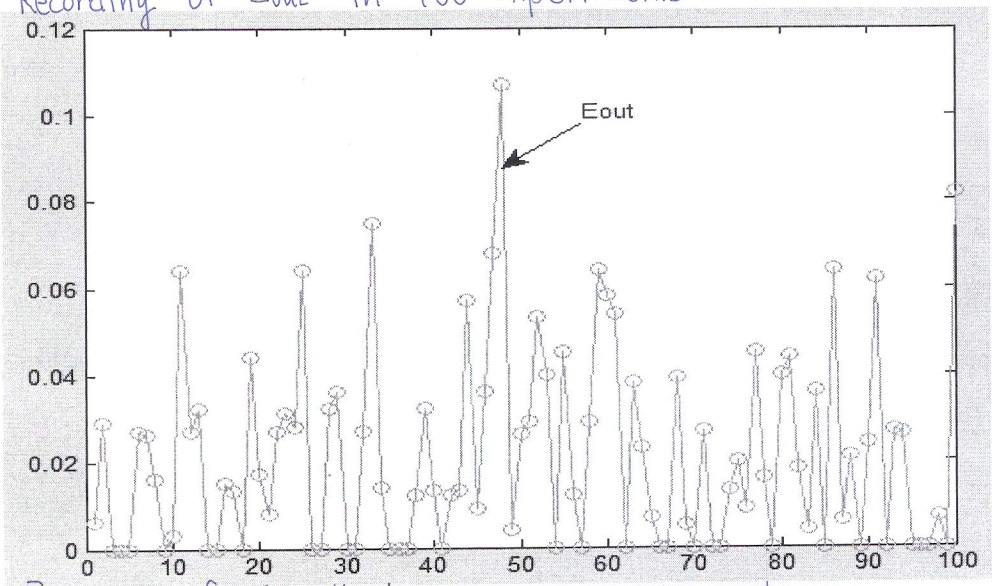
$\rightarrow 2^N = 2^5 = 32$

credibility $= \frac{1}{32}$. $p[\text{falsification}] \geq \frac{31}{32}$

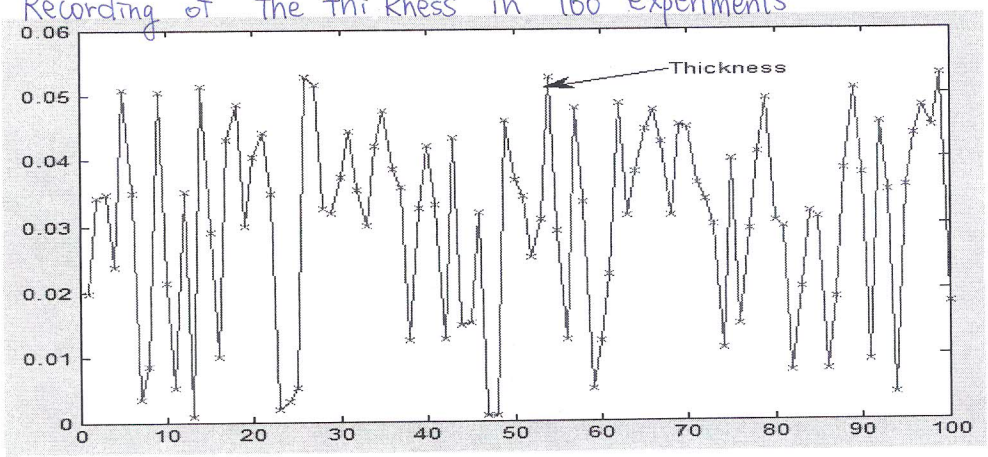
5.7

(1)

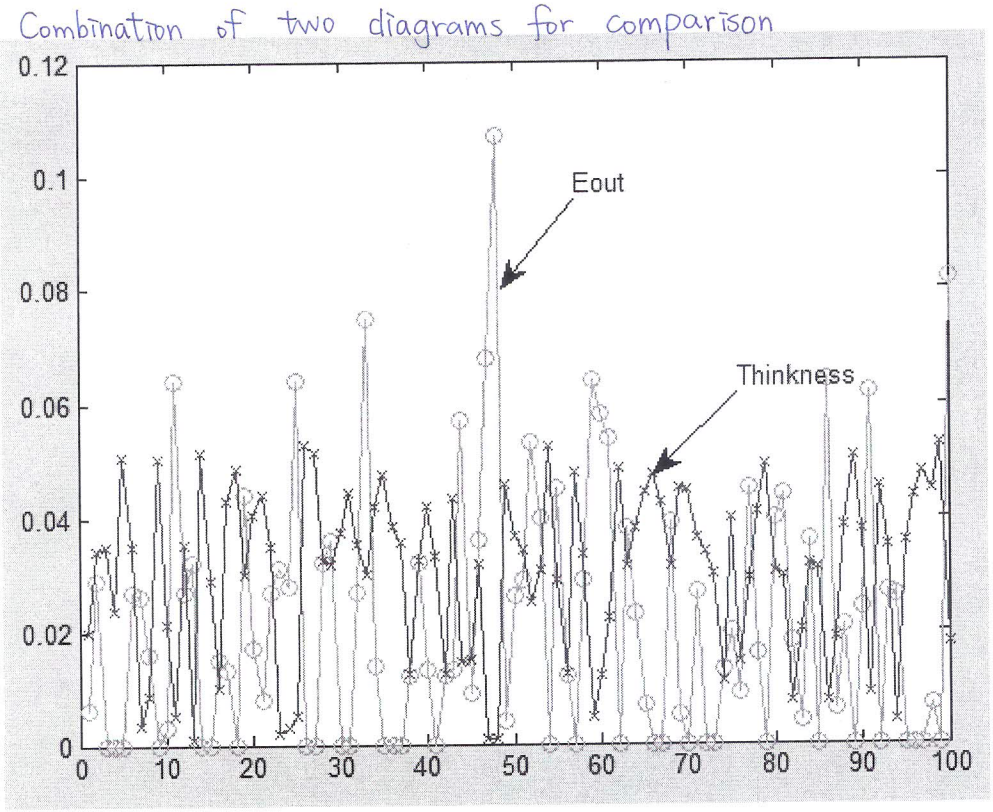
Recording of Eout in 100 experiments.



Recording of the thickness in 100 experments



Combination of two diagrams for comparison

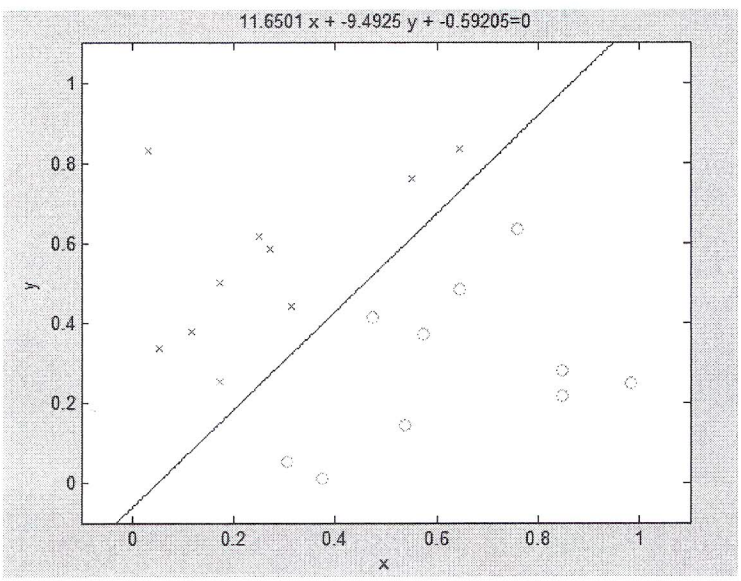


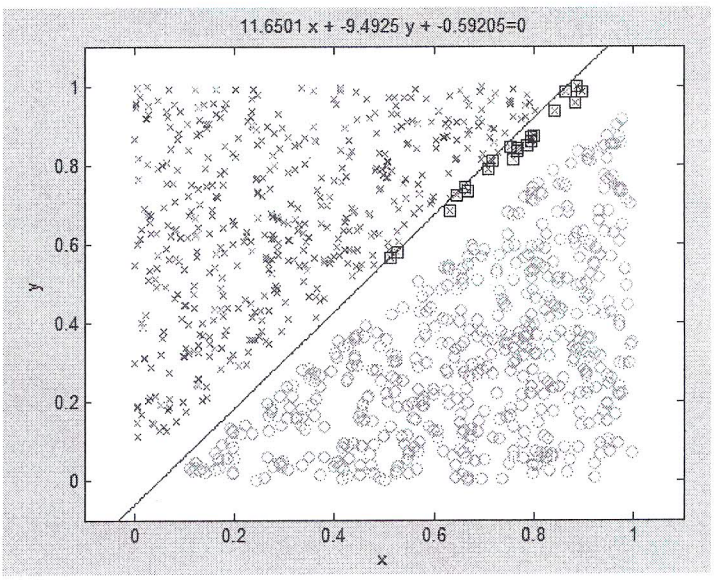
Comparing those two diagrams, we can notice some experiments which have large Eout are "more likely" to have small values of thickness in general. But it doesn't mean that large thickness can always result in a small Eout, not so absolute. It is more likely to be in that way because if you have a large thickness, you may have a larger "space" to fit the testing data.

(2) The largest thickness we can get from the training data is 0.0665, and the hypothesis having the largest thickness results in Eout = 0.021. Compared with the previous diagrams, we can conclude that, even though the hypothesis have the largest thickness doesn't guarantee its Eout is the smallest one.

Eout = 0.021

L = 0.0665





Homework #6

TA in charge: Chia-Hsuan Wang

RELEASE DATE: 11/29/2010

DUE DATE: 12/13/2010, 4:00 pm IN CLASS

TA SESSION: 12/09/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

6.1 Transforms: Explicit versus Implicit

Consider the following training set:

$$\mathbf{x}_1 = (1,0), y_1 = -1 \qquad \mathbf{x}_2 = (0,1), y_2 = -1 \qquad \mathbf{x}_3 = (0,-1), y_3 = -1$$
$$\mathbf{x}_4 = (-1,0), y_4 = +1 \qquad \mathbf{x}_5 = (0,2), y_5 = +1 \qquad \mathbf{x}_6 = (0,-2), y_6 = +1$$
$$\mathbf{x}_7 = (-2,0), y_7 = +1$$

(1) Use following nonlinear transformation of the input space $\mathcal{X}$ into another two-dimensional space Φ:

$$\phi_1(\mathbf{x}) = (\mathbf{x}[2])^2 - 2\mathbf{x}[1] - 1 \qquad \phi_2(\mathbf{x}) = (\mathbf{x}[1])^2 - 2\mathbf{x}[2] + 1$$

(a) (5%) Transform the training set into the Φ space.

(b) (5%) Write down the equation of the optimal separating "hyperplane" in the Φ space. Then, plot the transformed training points on the Φ plane as well as the boundary between the $+1$ and -1 regions, and mark the support vectors.

(c) (5%) Write down the equation of the corresponding nonlinear curve in the $\mathcal{X}$ space. Then, plot the original training points on the $\mathcal{X}$ plane as well as the boundary between the $+1$ and -1 regions, and mark the support vectors.

(2) Consider the same training set, but instead of explicitly transforming the input space $\mathcal{X}$, apply the (hard-margin) SVM algorithm with the kernel function

$$K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^2,$$

which corresponds to a second-order polynomial transformation.

(a) (5%) Set up the optimization problem using $(\alpha_1, \cdots, \alpha_7)$ and numerically solve for them (you can use any package you want). What is the optimal $\boldsymbol{\alpha}$?

(b) (5%) Write down the equation of the corresponding nonlinear curve in the $\mathcal{X}$ space. Then, plot the original training points on the $\mathcal{X}$ plane as well as the boundary between the $+1$ and -1 regions, and mark the support vectors.

(3) (5%) Should the two nonlinear curves (and support vectors) found in (1) and (2) be the same? Why or why not? Make a comparison and briefly describe your findings.

6.2 A Leave-One-Out Bound of Support Vector Machine

Consider the soft-margin SVM

$$(D) \quad \min_{\boldsymbol{\alpha}} \quad E_N(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n$$
$$\text{subject to} \quad \sum_{n=1}^{N} y_n\alpha_n = 0$$
$$0 \le \alpha_n \le C$$

and a soft-margin SVM without the N-th example

$$(D_{-N}) \quad \min_{\boldsymbol{\beta}} \quad E_{N-1}(\boldsymbol{\beta}) = \frac{1}{2}\sum_{n=1}^{N-1}\sum_{m=1}^{N-1}\beta_n\beta_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m) - \sum_{n=1}^{N-1}\beta_n$$
$$\text{subject to} \quad \sum_{n=1}^{N-1} y_n\beta_n = 0$$
$$0 \le \beta_n \le C$$

(1) (5%) Assume that $\boldsymbol{\alpha}^*$ is an optimal solution for (D) with $\alpha_N^* = 0$. Let $\hat{\boldsymbol{\beta}} = \left(\alpha_1^*, \alpha_2^*, \ldots, \alpha_{N-1}^*\right)$. Argue that $\hat{\boldsymbol{\beta}}$ is a feasible vector for (D_{-N}). That is, check that $\hat{\boldsymbol{\beta}}$ satisfies all constraints of (D_{-N}).

(2) (5%) Assume that $\boldsymbol{\beta}^*$ is an optimal solution for (D_{-N}). That is, $E_{N-1}(\boldsymbol{\beta}) \ge E_{N-1}(\boldsymbol{\beta}^*)$ for all feasible vectors $\boldsymbol{\beta}$. Prove that the $\hat{\boldsymbol{\beta}}$ above satisfies

$$E_{N-1}(\hat{\boldsymbol{\beta}}) = E_{N-1}(\boldsymbol{\beta}^*).$$

In other words, $\hat{\boldsymbol{\beta}}$ is also optimal for (D_{-N}).

(3) (5%) Recall that #SV = (# of nonzero α_n^*). With the results in (1) and (2), prove that the leave-one-out cross-validation error of SVM is upper bounded by the percentage of support vectors. You can use the fact that

$$\alpha_n^* = 0 \Longrightarrow y_n\left(\mathbf{w}^* \cdot \phi(\mathbf{x}_n) + b^*\right) \ge 1.$$

6.3 Radius of Transformed Vectors via the Kernel

Recall that for support vector machines, d_{VC} is upper bounded by $\frac{R^2}{\rho^2}$, where ρ is the margin and R is the radius of the minimum hypersphere that $\mathcal{X}$ resides in. In general, R should come from our knowledge on the learning problem, but we can *estimate* it by looking at the minimum hypersphere that the training examples resides in. In particular, we want to seek for the optimal R for

$$(P) \quad \min_{R\in\mathbb{R},\mathbf{c}\in\mathbb{R}^d} R^2 \text{ subject to } \|\mathbf{x}_n - \mathbf{c}\|^2 \le R^2 \text{ for } n = 1, 2, \cdots, N.$$

(1) (5%) Let λ_n be the Lagrange multipliers for the n-th constraint above. Following the derivation of the dual SVM in class, write down (P) as an equivalent optimization problem

$$\min_{R\in\mathbb{R},\mathbf{c}\in\mathbb{R}^d} \quad \max_{\lambda_n\ge 0} \quad L(R,\mathbf{c},\boldsymbol{\lambda}).$$

What is $L(R,\mathbf{c},\boldsymbol{\lambda})$?

(2) (5%) Using (assuming) strong duality, the solution to (P) would be the same as the Lagrange dual problem

$$\max_{\lambda_n\ge 0} \quad \min_{R\in\mathbb{R},\mathbf{c}\in\mathbb{R}^d} \quad L(R,\mathbf{c},\boldsymbol{\lambda}).$$

Use the optimality conditions of the inner optimization problem to simplify the Lagrange dual problem, and obtain a dual problem that involves only λ_n.

(3) (5%) Explain what would happen when we use $\phi(\mathbf{x}_n)$ instead of $\mathbf{x}_n$, and write down the optimization problem that uses $K(\mathbf{x}_n, \mathbf{x}_m)$ to replace $\phi(\mathbf{x}_n) \bullet \phi(\mathbf{x}_m)$—that is, the kernel trick.

(4) (Bonus 5%) Show how you would get the optimal *value* of R^2 after solving the dual optimization problem in (3).

6.4 Experiments with Nonlinear Support Vector Machine (*)

Write a program to implement the nonlinear Support Vector Machine by solving

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n \\ \text{subjectto} \quad & \sum_{n=1}^{N} y_n\alpha_n = 0 \\ & 0 \le \alpha_n \le C \end{aligned}$$

Run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw6_4_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw6_4_test.dat`

(1) (10%) Use the polynomial kernel $(1 + \mathbf{x} \cdot \mathbf{x}')^d$ with $d = 3, 6, 9$, and $C = 0.001, 1, 1000$. Let $g^{(1)}_{d,C}$ be the classifier obtained when using (d, C) as the parameter. For each (d, C) combination, show $E_{\text{in}}\left(g^{(1)}_{d,C}\right)$, $E_{\text{out}}\left(g^{(1)}_{d,C}\right)$, and $\frac{\#SV}{N}$. Briefly describe your findings.

(2) (10%) Use the Gaussian-RBF kernel $\exp\left(\frac{-(\mathbf{x}-\mathbf{x}')^2}{2\sigma^2}\right)$ with $\sigma = 0.125, 0.5, 2$ and $C = 0.001, 1, 1000$. Let $g^{(2)}_{\sigma,C}$ be the classifier obtained when using (σ, C) as the parameter. For each (σ, C) combination, show $E_{\text{in}}\left(g^{(2)}_{\sigma,C}\right)$, $E_{\text{out}}\left(g^{(2)}_{\sigma,C}\right)$, and $\frac{\#SV}{N}$. Briefly describe your findings.

(*Note: For this problem, you CAN use any package you want. A recommended choice is LIBSVM developed by Prof. Chih-Jen Lin in our department*)

6.5 Experiments with Bootstrap Aggregation (*)

(1) (10%) Recall that you proved the d_{VC} of decision stumps in Homework 5.1. Now, let us implement a decision stump learning algorithm A_{ds}. In particular, set

$$h_{s,i,\theta}(\mathbf{x}) = \text{sign}\Big(s \cdot \mathbf{x}[i] - \theta\Big),$$

where $s \in \{-1, +1\}$, $i \in \{1, 2, \ldots, d\}$, and $\theta \in \mathbb{R}$. Given a weighted training set $Z = \{(\mathbf{x}_n, y_n, u_n)\}_{n=1}^{N}$,

$$A_{ds}(Z) = \underset{h_{s,i,\theta}}{\text{argmin}} \sum_{n=1}^{N} u_n \cdot [\![y_n \neq h_{s,i,\theta}(\mathbf{x}_n)]\!] .$$

Run the algorithm on the following set for training (with $u_n = \frac{1}{N}$ for all N):

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw6_5_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/hw6_5_test.dat`

Let g be the classifier returned from A_{ds}. Report $E_{\text{in}}(g)$ and $E_{\text{out}}(g)$. Briefly state your findings.

(2) (10%) Implement the bootstrap aggregation (bagging) algorithm with decision stumps (i.e., use A_{ds} as A_b below):

- Input: $Z = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$.
- for $t = 1, 2, \ldots, T$,
  - generate $Z^{(t)}$ from Z by bootstrapping—uniformly sampling N examples from Z with replacement
  - let $h_t = A_b(Z^{(t)})$ and $\alpha_t = 1$.
- Output: an ensemble $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

Use a total of $T = 100$ iterations. Let $H_t(\mathbf{x}) = \text{sign}\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x})\right)$. Plot $E_{\text{in}}(H_t)$ and $E_{\text{out}}(H_t)$ as functions of t on the same figure. Briefly state your findings.

(3) (Bonus 5%) Prove that you can implement an A_{ds} that runs in time $O(N \log N)$ instead of the brute-force implementation that takes $O(N^2)$.

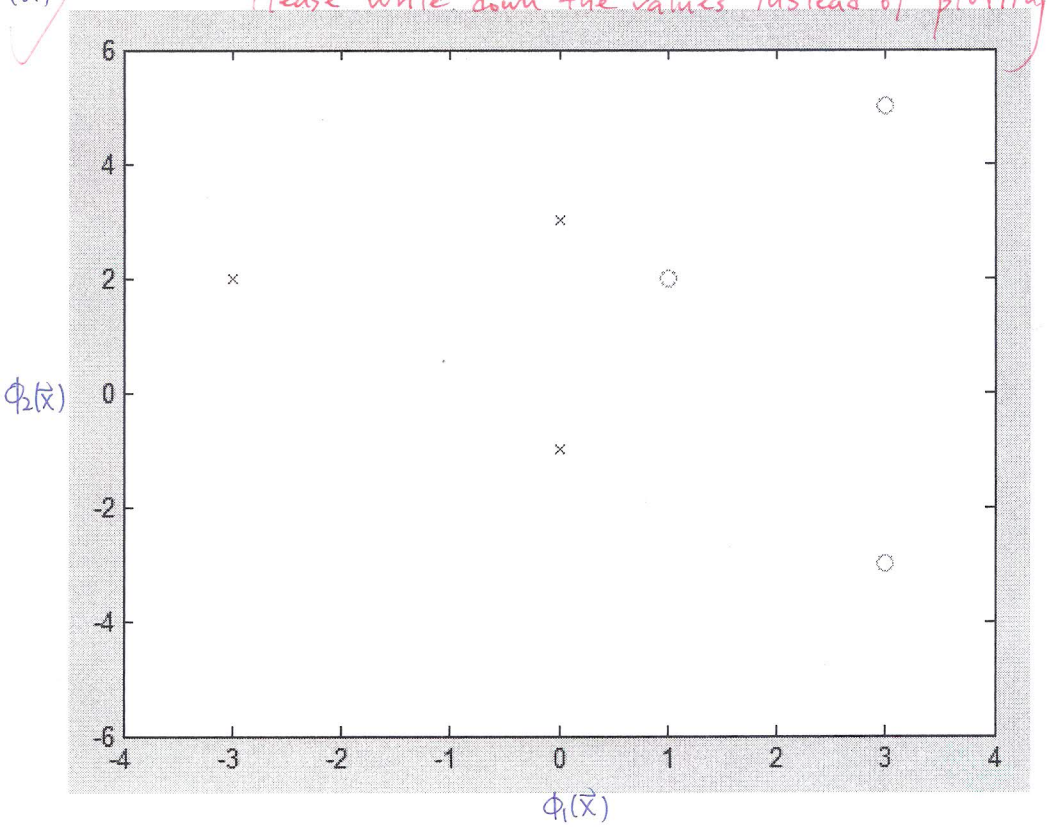
ML HW #6 R98922081 徐位文

93 + 3

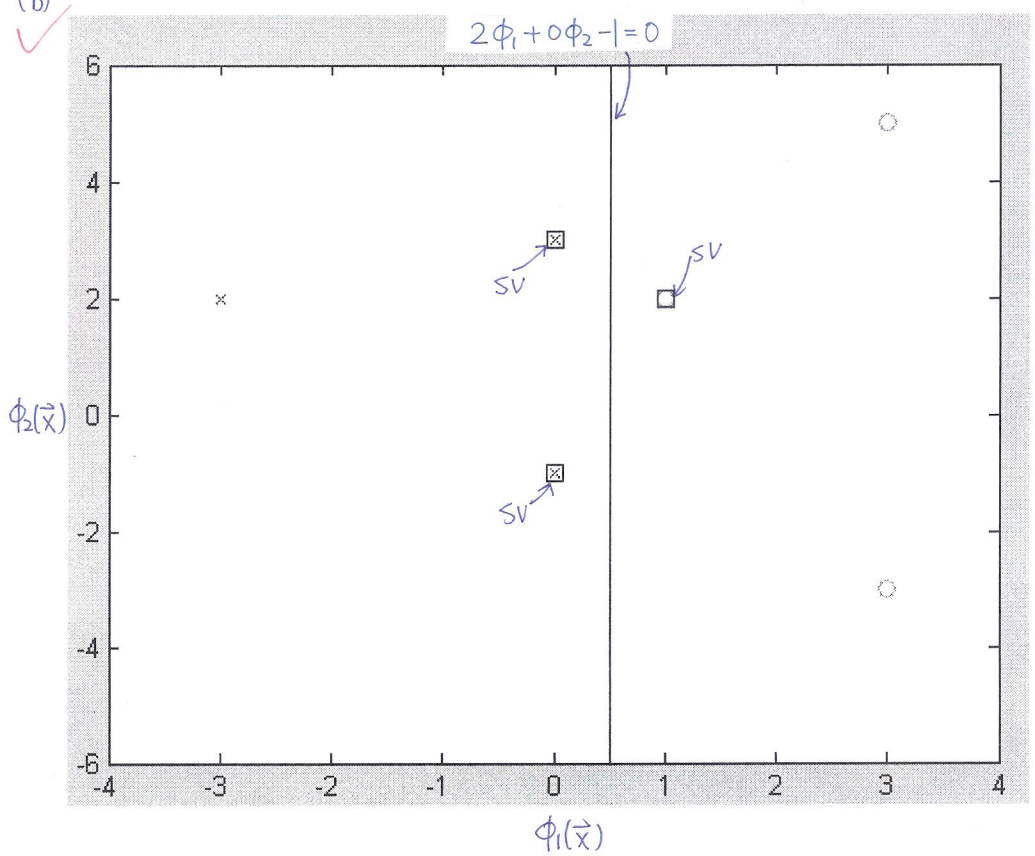
6.1 30

(1)

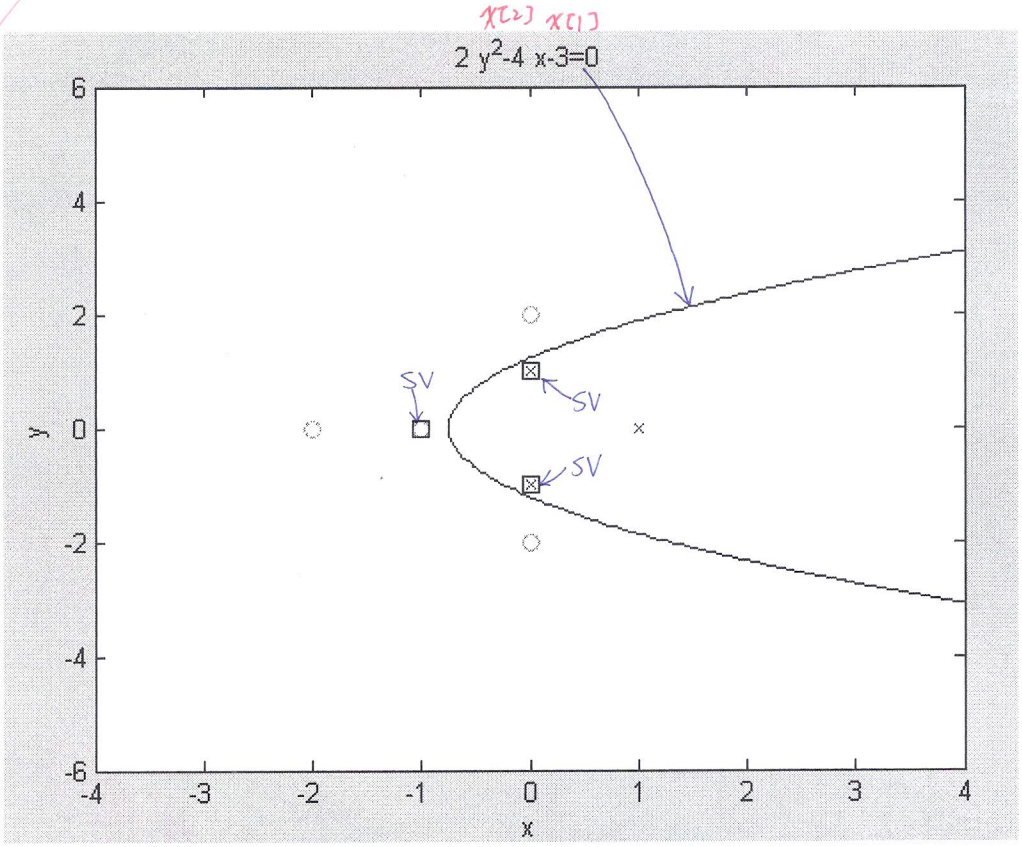
(a) Please write down the values instead of plotting them.



(b)



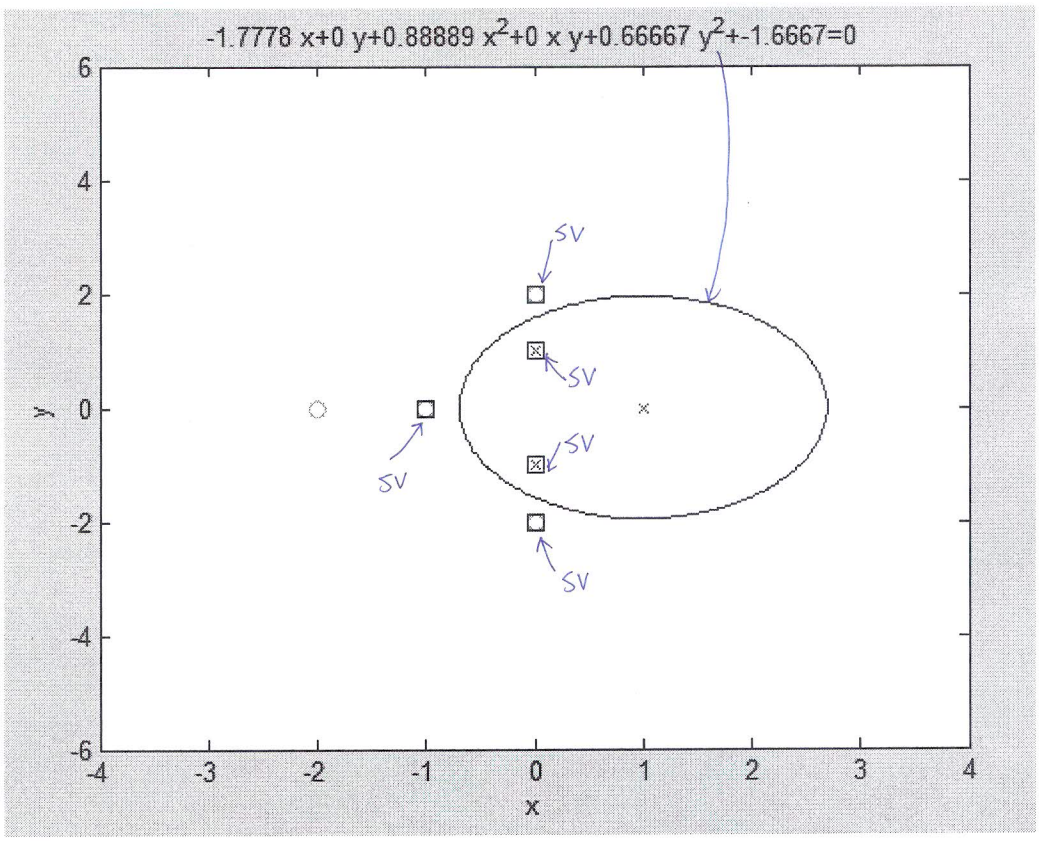
(c)



(2)

(a) Alpha = [0 1.2222 0.1852 0.8889 0.5185 0 0]

(b)



(3) Two nonlinear curves are different. The curve in (1) is a parabola and in (2) is an ellipse. And there are 3 support vectors in (1) and 5 support vectors in (2). Two curves are different because we use different transformations. Each hyperplane in its transforming domain is affacted by its corresponding support vectors. When we transformed it back to X domain, curves are determined by its corresponding transformation.

6.2.11

(1) $\because \vec{\alpha}^*$ is an optimal solution for (D)

$\therefore \vec{\alpha}^*$ satitfies the constraints of (D)

that is, $\sum_{n=1}^{N} y_n \alpha_n^* = 0 \Rightarrow \sum_{n=1}^{N-1} y_n \alpha_n^* + y_N \alpha_N^* = 0 \quad \because \alpha_N^* = 0$

$\Rightarrow \sum_{n=1}^{N-1} y_n \alpha_n^* = 0$ and $\hat{\beta} = (\alpha_1^*, \alpha_2^*, \cdots, \alpha_{N-1}^*)$

Therefore, for constraints $\sum_{n=1}^{N-1} y_n \beta_n = 0$ in (D-N), $\hat{\beta}$ holds.

And for $\vec{\alpha}^*$: $0 \le \alpha_n^* \le C$ for $n = 1 \sim N$

$\Rightarrow 0 \le \alpha_n^* \le C$ for $n = 1 \sim N-1 \Rightarrow$ for constraints $0 \le \beta_n \le C$, $\hat{\beta}$ holds

(2) $\vec{\alpha}^* = \arg\min_{\vec{\alpha}} E_N(\vec{\alpha}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(\vec{x}_n, \vec{x}_m) - \sum_{n=1}^{N} \alpha_n$

$\Rightarrow E_N(\vec{\alpha}^*) = \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n^* \alpha_m^* y_n y_m K(\vec{x}_n, \vec{x}_m) - \sum_{n=1}^{N} \alpha_n^* \quad \because \alpha_N^* = 0$

$= \frac{1}{2}\sum_{n=1}^{N-1}\sum_{m=1}^{N-1} \alpha_n^* \alpha_m^* y_n y_m K(\vec{x}_n, \vec{x}_m) - \sum_{n=1}^{N-1} \alpha_n^* \quad \because \hat{\beta} = (\alpha_1^*, \alpha_2^*, \cdots, \alpha_{N-1}^*)$

$= E_{N-1}(\hat{\beta})$

$\because \alpha_N^* = 0 \quad \therefore E_N(\vec{\alpha}^*) = E_{N-1}(\vec{\beta}^*)$ why?

$\Rightarrow E_{N-1}(\hat{\beta}) = E_{N-1}(\vec{\beta}^*)$

1° $E_{N-1}(\vec{\beta}^*) \le E_{N-1}(\hat{\beta})$

2° $E_{N-1}(\hat{\beta}) = E_N(\vec{\alpha}^*) \le E_N(\vec{\alpha}') = E_{N-1}(\vec{\beta}^*)$

$\vec{\alpha}' = [\beta_1^*, \beta_2^*, \cdots, \beta_{N-1}^*, \alpha_N^*]^T$

$\Rightarrow$ by 1°, 2° $E_{N-1}(\hat{\beta}) = E_{N-1}(\vec{\beta}^*)$

(3) From (1) and (2), we know that if we remove an example $\vec{x}_n$ which corresponding $\alpha_n^* = 0$ from the training data, then the hypothesis trained by $N-1$ examples will be the same as by N examples. In leave-one-out cross-validation, those example points which corresponding $\alpha_n^* = 0$, then $y_n(\vec{w}^* \cdot \phi(\vec{x}_n) + b^*) \ge 1$, will always right as a testing point for the hypothesis trained by other $N-1$ examples. That is to say, the number of α_n^* which is nonzero is the most number of errors for leave-one-out, cross-validation. $\because$ # SV = # of nonzero α_n^* $\therefore$ the percentage of SV is the upper bound error rate of SVM.

6.3 1✓

(1) ✓ $\min_{R\in\mathbb{R},\,\vec{c}\in\mathbb{R}^d} R^2 \quad \text{s.t. } \|\vec{x}_n-\vec{c}\|^2 \le R^2 \text{ for } n=1\sim N$

$\Leftrightarrow \min_{R\in\mathbb{R},\,\vec{c}\in\mathbb{R}^d}\ \max_{\lambda_n\ge 0}\ R^2+\sum_{n=1}^{N}\lambda_n\left[\|\vec{x}_n-\vec{c}\|^2-R^2\right]$

$\Rightarrow L(R,\vec{c},\vec{\lambda}) = R^2+\sum_{n=1}^{N}\lambda_n\left[\|\vec{x}_n-\vec{c}\|^2-R^2\right]$

(2) $\max_{\lambda_n\ge 0}\ \min_{R\in\mathbb{R},\,\vec{c}\in\mathbb{R}^d} R^2+\sum_{n=1}^{N}\lambda_n\left[\|\vec{x}_n-\vec{c}\|^2-R^2\right]$

$\frac{\partial L(R,\vec{c},\vec{\lambda})}{\partial c_i}=0 \Rightarrow \vec{c}=\sum_{n=1}^{N}\lambda_n\vec{x}_n$ -1

$\frac{\partial L(R,\vec{c},\vec{\lambda})}{\partial R}=0 \Rightarrow \sum_{n=1}^{N}\lambda_n=1$ what about $R=0$? $\Sigma\lambda_n=0$? $\Sigma\lambda_n\neq 1$

$\Rightarrow \max_{\lambda_n\ge 0} R^2+\sum_{n=1}^{N}\lambda_n\left[\|\vec{x}_n-\vec{c}\|^2-R^2\right]$

$\Rightarrow \max_{\lambda_n\ge 0} R^2\left[1-\sum_{n=1}^{N}\lambda_n\right]+\sum_{n=1}^{N}\lambda_n\vec{x}_n\cdot\vec{x}_n-2\vec{c}\cdot\sum_{n=1}^{N}\lambda_n\vec{x}_n+\vec{c}\cdot\vec{c}\sum_{n=1}^{N}\lambda_n$ (the sum $\sum\lambda_n$ $\to 1$)

$\Rightarrow \max_{\lambda_n\ge 0} \sum_{n=1}^{N}\lambda_n\vec{x}_n\cdot\vec{x}_n-2\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]^T\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]+\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]^T\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]$

$\Rightarrow \max_{\lambda_n\ge 0} -\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]^T\left[\sum_{n=1}^{N}\lambda_n\vec{x}_n\right]+\sum_{n=1}^{N}\lambda_n\|\vec{x}_n\|^2$

$\Rightarrow \min_{\lambda_n\ge 0} \sum_{n=1}^{N}\sum_{m=1}^{N}\lambda_n\lambda_m\vec{x}_n\cdot\vec{x}_m-\sum_{n=1}^{N}\lambda_n\|\vec{x}_n\|^2$ subject to $\Sigma\lambda_n=1$

(3) $\min_{\lambda_n\ge 0} \sum_{n=1}^{N}\sum_{m=1}^{N}\lambda_n\lambda_m\,\underline{\Phi(\vec{x}_n)\cdot\Phi(\vec{x}_m)}-\sum_{n=1}^{N}\lambda_n\|\Phi(\vec{x}_n)\|^2$

$\Rightarrow \min_{\lambda_n\ge 0} \sum_{n=1}^{N}\sum_{m=1}^{N}\lambda_n\lambda_m\,\underline{K(\vec{x}_n,\vec{x}_m)}-\sum_{n=1}^{N}\lambda_n\underline{[K(\vec{x}_n,\vec{x}_n)]}$

Why bother using ϕ? ->

(4) We can get λ_n^* from above equation and choose an example $\vec{x}_m$ which corresponding $\lambda_m^*>0$, and we know $\vec{c}=\sum_{n=1}^{N}\lambda_n\Phi(\vec{x}_n)$, then the optimal value of $R_\phi^2=\|\Phi(\vec{x}_m)-\vec{c}\|^2$ — why? — x

+3 where $\|\Phi(\vec{x}_m)-\vec{c}\|^2=(\Phi(\vec{x}_m)-\vec{c})\cdot(\Phi(\vec{x}_m)-\vec{c})$

$=\Phi(\vec{x}_m)\cdot\Phi(\vec{x}_m)-2\Phi(\vec{x}_m)\cdot\vec{c}+\vec{c}\cdot\vec{c}$

$=K(\vec{x}_m,\vec{x}_m)-2\sum_{n=1}^{N}\lambda_n^*K(\vec{x}_m,\vec{x}_n)+\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i^*\lambda_j^*K(\vec{x}_i,\vec{x}_j)$

6.4 20

(1) $k(\vec{x}, \vec{x}') = (1 + \vec{x} \cdot \vec{x}')^d$

	C=0.001	C=1	C=1000
d=3	Ein = 0.4200 Eout = 0.5140 SV rate = 0.8500	Ein = 0.1300 Eout = 0.1830 SV rate = 0.4800	Ein = 0.1200 Eout = 0.1150 SV rate = 0.3600
d=6	Ein = 0.4200 Eout = 0.5140 SV rate = 0.8400	Ein = 0.1100 Eout = 0.1770 SV rate = 0.4400	Ein = 0.0600 Eout = 0.1090 SV rate = 0.2600
d=9	Ein = 0.2400 Eout = 0.2370 SV rate = 0.6800	Ein = 0.0700 Eout = 0.1030 SV rate = 0.2900	Ein = 0.0700 Eout = 0.1480 SV rate = 0.2900

1° In general, in the same condition of C, the bigger d is, the more complicated the hypothesis will be, so we can have lower Ein but have risks of overfitting.

2° In general, in the same condition of d, the bigger C is, the lower Ein we will get. Because bigger C will reduce the violations while training. overfitting risk also

3° In general, the higher SV rate is, the higher Ein and Eout we will have.
bounded by $\frac{\#SV}{N}$

(2) $K(\vec{x}, \vec{x}') = \exp(-\|\vec{x} - \vec{x}'\|^2 / 2\sigma^2)$

	C=0.001	C=1	C=1000
σ=0.125	Ein = 0.4200 Eout = 0.5140 SV rate = 0.8700	Ein = 0.0300 Eout = 0.1160 SV rate = 0.4900	Ein = 0 Eout = 0.1720 SV rate = 0.2600
σ=0.5	Ein = 0.4200 Eout = 0.5140 SV rate = 0.8400	Ein = 0.1200 Eout = 0.1870 SV rate = 0.5300	Ein = 0.0600 Eout = 0.1070 SV rate = 0.2800
σ=2	Ein = 0.4200 Eout = 0.5140 SV rate = 0.8400	Ein = 0.2200 Eout = 0.2750 SV rate = 0.8300	Ein = 0.1300 Eout = 0.1830 SV rate = 0.4500

1° In general, in the same condition of σ, the bigger C is, the lower Ein we can have, but still may have problems of overfitting. (e.g. σ=0.125, C=1 and C=1000)

2° In general, in the same condition of C, the smaller σ is, the lower Ein we can get. Because the smaller σ of the Gaussian function will have a better discriminant ability. And it may have overfitting problem. (e.g. C=1000, σ=0.125 and σ=0.5)

3° In general, the higher SV rate, the higer Ein and Eout we will have.

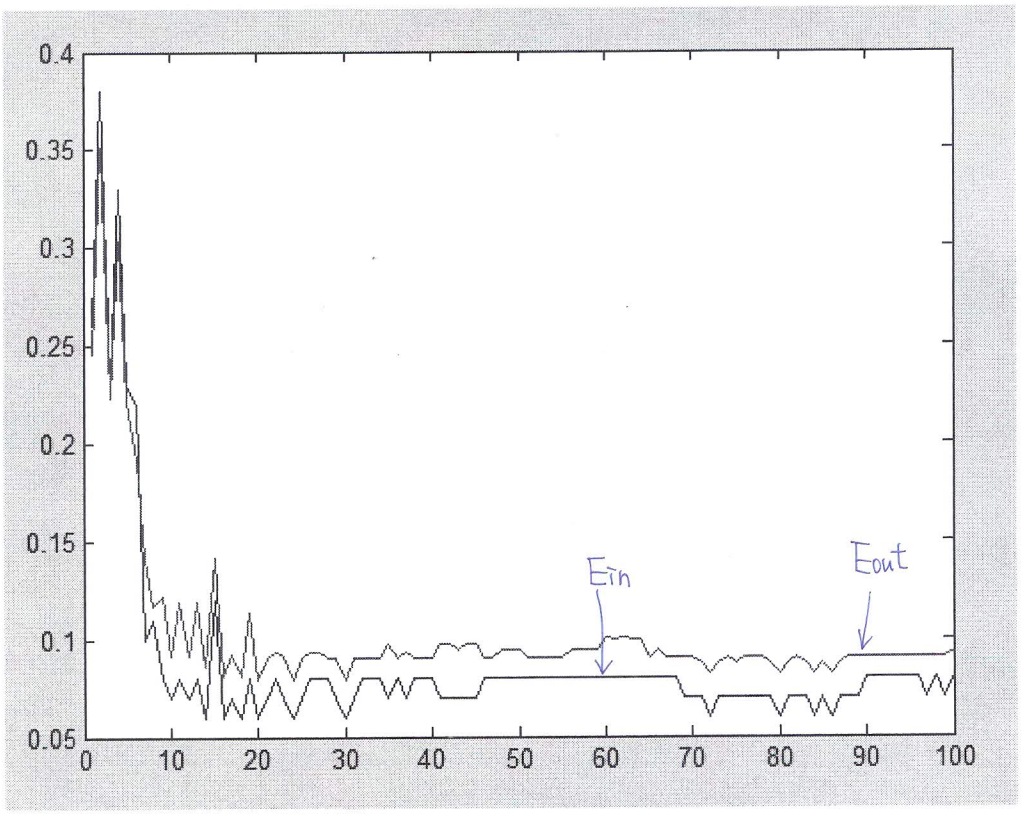
6.5 70

(1)

Ein = 0.2300 Eout = 0.2560

In this case, the smallest Ein happened when d=1, s=1, and we choosed θ=0.6451. The hypothesis we made results in Eout = 0.2560.

(2)



After bagging algorithm with more iterations, Ein and Eout seem to have the same tendency to go down and converge. We can see that both Ein and Eout after bagging become obviously lower than without bagging.

(3) For certain dimension i, s (+1/-1), we can sort $\vec{x}_n[i]$, $n=1\sim N$ from training data first, and then use binarary search (How?) instead of sequential search. So the complexity would be $\underbrace{O(N\log N)}_{\text{sorting}} + \underbrace{O(N\log N)}_{\text{find }\theta\text{ (binary search)}} = O(N\log N)$

$+\underbrace{O(N)}_{\text{try } N \text{ examples}}$

Get 2.5 pts if you explain to TA clearly and I believe that you're right.

Homework #7

TA in charge: Yu-Xun Ruan and Chen-Wei Hung

RELEASE DATE: 12/13/2010

DUE DATE: 12/27/2010, 4:00 pm IN CLASS

TA SESSION: 12/23/2010, 6:00 pm IN R110

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

7.1 Kernels and Transforms

(1) (5%) Let all the input vectors be real values ($\mathbf{x} \in \mathcal{X} \in \mathbb{R}$) with $|\mathbf{x}| < 1$. Consider a kernel function $K(\mathbf{x}, \mathbf{x}') = \frac{1}{1-\mathbf{x} \bullet \mathbf{x}'}$. Prove that K is a valid kernel by deriving a transform function $\boldsymbol{\phi}(\mathbf{x})$ such that

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \bullet \boldsymbol{\phi}(\mathbf{x}').$$

(2) (5%) For two valid kernels $K_1(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_1(\mathbf{x}) \bullet \boldsymbol{\phi}_1(\mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_2(\mathbf{x}) \bullet \boldsymbol{\phi}_2(\mathbf{x}')$, consider a kernel function $K(\mathbf{x}, \mathbf{x}') = \gamma_1 K_1(\mathbf{x}, \mathbf{x}') + \gamma_2 K_2(\mathbf{x}, \mathbf{x}')$ with $\gamma_1 > 0$ and $\gamma_2 > 0$. Prove that K is a valid kernel by deriving a transform function $\boldsymbol{\phi}(\mathbf{x})$ such that

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \bullet \boldsymbol{\phi}(\mathbf{x}').$$

(The result shows that a conic combination of valid kernels is still a valid kernel.)

(3) (5%) For two valid kernels $K_1(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_1(\mathbf{x}) \bullet \boldsymbol{\phi}_1(\mathbf{x}')$ and $K_2(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_2(\mathbf{x}) \bullet \boldsymbol{\phi}_2(\mathbf{x}')$, consider a kernel function $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') \cdot K_2(\mathbf{x}, \mathbf{x}')$. Prove that K is a valid kernel by deriving a transform function $\boldsymbol{\phi}(\mathbf{x})$ such that

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x}) \bullet \boldsymbol{\phi}(\mathbf{x}').$$

(The result shows that a multiplication of valid kernels is still a valid kernel.)

(4) (5%) For a given data set $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}$ in which all $\mathbf{x}_n$ are different. Consider the feature transform in Problem 5.2(1):

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{cases} (\underbrace{0, \cdots, 0}_{n-1}, 1, 0, \cdots) & \text{if } \mathbf{x} = \mathbf{x}_n \\ (0, \cdots, 0, 0, 0, \cdots) & \text{otherwise .} \end{cases}$$

What is the kernel function that corresponds to this transform? When using the kernel function, what does the matrix Q in the dual problem of SVM look like?

(5) (5%) Prove that the function $K(\mathbf{x}, \mathbf{x}') = -(\mathbf{x} \bullet \mathbf{x}')^2 + 2\mathbf{x} \bullet \mathbf{x}'$ is NOT a valid kernel function.

7.2 Kernel from Decision Stumps

When talking about non-uniform voting in aggregation, we mentioned that $\boldsymbol{\alpha}$ can be viewed as a weight vector learned from any linear algorithm coupled with the following transform:

$$\boldsymbol{\phi}(\mathbf{x}) = \Big(h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_T(\mathbf{x})\Big).$$

When studying kernel methods, we mentioned that the kernel is simply a computational short-cut for the inner product $\boldsymbol{\phi}(\mathbf{x}) \bullet \boldsymbol{\phi}(\mathbf{x}')$. In this problem, we mix the two topics together using the decision stumps as our $h(\mathbf{x})$.

(1) (5%) Assume that the input vectors contain only bounded integers. That is, $\mathcal{X} \subseteq (\mathbb{Z} \cap [-B, B])^d$. Consider decision stumps

$$h_{s,i,\theta}(\mathbf{x}) = \text{sign}\Big(s \cdot \mathbf{x}[i] - \theta\Big).$$

Two decision stumps $h^{(1)}$ and $h^{(2)}$ are defined as the *same* if $h^{(1)}(\mathbf{x}) = h^{(2)}(\mathbf{x})$ for every $\mathbf{x} \in \mathcal{X}$. Two decision stumps are different if they are not the same. Argue that there are only finitely-many different decision stumps for $\mathcal{X}$ and list all of them for the case of $d = 3$ and $B = 2$.

(2) (5%) Let $\mathcal{H} = \{$ all different decision stumps for $\mathcal{X}$ $\}$. Since $\mathcal{H}$ is finite, we can enumerate each hypothesis $h \in \mathcal{H}$ by some index t. Define

$$\boldsymbol{\phi}_{ds}(\mathbf{x}) = \Big(h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_t(\mathbf{x}), \cdots, h_{|\mathcal{H}|}(\mathbf{x})\Big).$$

Derive a simple equation that evaluates $K_{ds}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_{ds}(\mathbf{x}) \bullet \boldsymbol{\phi}_{ds}(\mathbf{x}')$ efficiently.

(3) (Bonus 5%) Argue that for any kernel function $K(\mathbf{x}, \mathbf{x}')$, using $K(\mathbf{x}, \mathbf{x}') + c$ for any constant c is equivalent to using $K(\mathbf{x}, \mathbf{x}')$ in the dual of (hard- or soft-margin) SVM. In other words, prove that the resulting classifier is exactly the same. Use your argument to obtain a kernel of the form $K_{ds}(\mathbf{x}, \mathbf{x}') + c$ that is even easier to evaluate.

The result can be easily extended to the case when $\mathcal{X}$ is an arbitrary box in $\mathbb{R}^d$ as well.

7.3 Power of Adaptive Boosting

The adaptive boosting (AdaBoost) algorithm, as shown in the class slides, is as follows:

- For input $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, set $u_n = \frac{1}{N}$ for all n.
- For $t = 1, 2, \cdots, T$,
  - Learn a simple hypothesis h_t such that h_t solves

$$h_t = \underset{h \in \mathcal{H}}{\text{argmin}} \sum_{n=1}^{N} u_n \cdot [\![y_n \neq h(\mathbf{x}_n)]\!].$$

with the help of some base learner A_b that learns from $h \in \mathcal{H}$.

  - Compute the weighted error $\epsilon_t = \dfrac{\sum_{n=1}^N u_n \cdot [\![y_n \neq h_t(\mathbf{x}_n)]\!]}{\sum_{n=1}^N u_n}$ and the confidence

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

  - Change the example weights: $u_n = u_n \cdot \exp\Big(-\alpha_t y_n h_t(\mathbf{x}_n)\Big)$.
- Output: combined function $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

In this problem, we will prove that AdaBoost can reach $E_{\text{in}}(H) = 0$ if T is large enough and every hypothesis h_t satisfies $\epsilon_t \leq \epsilon < \frac{1}{2}$.

(1) (5%) Let $U^{(t-1)} = \sum_{n=1}^{N} u_n$ at the beginning of the t-th iteration. According to the AdaBoost algorithm above, for $t \geq 1$, prove that

$$U^{(t)} = \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x}_n)\right).$$

(2) (5%) By the result in (1), prove that $E_{\text{in}}(H) \leq U^{(T)}$.

(3) (5%) According to the AdaBoost algorithm above, for $t \geq 1$, prove that $U^{(t)} = U^{(t-1)} \cdot 2\sqrt{\epsilon_t(1-\epsilon_t)}$.

(4) (5%) Using $0 \leq \epsilon_t \leq \epsilon < \frac{1}{2}$, for $t \geq 1$, prove that $\sqrt{\epsilon_t(1-\epsilon_t)} \leq \sqrt{\epsilon(1-\epsilon)}$.

(5) (5%) Using $\epsilon < \frac{1}{2}$, prove that $\sqrt{\epsilon(1-\epsilon)} \leq \frac{1}{2}\exp\left(-2(\frac{1}{2}-\epsilon)^2\right)$.

(6) (5%) Using the results above, prove that $U^{(T)} \leq \exp\left(-2T(\frac{1}{2}-\epsilon)^2\right)$.

(7) (5%) Using the results above, argue that after $T = O(\log N)$ iterations, $E_{\text{in}}(H) = 0$.

7.4 Optimization View of Adaptive Boosting

Assume that $\mathcal{H} = \{h_\ell\}_{\ell=1}^{L}$ is a finite set of hypotheses. Consider an error function for an ensemble $H(\mathbf{x})$ of the form $\text{sign}\left(\sum_{\ell=1}^{L} \beta_\ell h_\ell(\mathbf{x})\right)$:

$$E_{\text{exp}}(H) = \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \sum_{\ell=1}^{L} \beta_\ell h_\ell(\mathbf{x}_n)\right).$$

Using your results in Problem 7.4(3), we can easily show that $E_{\text{exp}}(H)$ is an upper bound of $E_{\text{in}}(H)$. We now prove that AdaBoost is equivalent to a particular way of minimizing $E_{\text{exp}}(H)$.

(1) (5%) Assume that we hope to update from $\boldsymbol{\beta}^{old}$ to $\boldsymbol{\beta}^{new}$ by changing only component i of $\boldsymbol{\beta}^{old}$. That is, for a given vector $\boldsymbol{\beta}^{old} = (\beta_1^{old}, \beta_2^{old}, \cdots, \beta_{L-1}^{old}, \beta_L^{old})$, we want to set

$$\beta_i^{new} = \beta_i^{old} + \Delta_i.$$

such that

$$\Delta_i = \underset{\Delta}{\text{argmin}} \ \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \left(\sum_{\ell=1}^{L} \beta_\ell^{old} h_\ell(\mathbf{x}_n)\right) - y_n\big(\Delta \cdot h_i(\mathbf{x}_n)\big)\right).$$

Let $u_n = \frac{1}{N}\exp\left(-y_n \sum_{\ell=1}^{L} \beta_\ell^{old} h_\ell(\mathbf{x}_n)\right)$ and $\epsilon_i = \frac{\sum_{n=1}^{N} u_n \cdot [\![y_n \neq h_i(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n}$. What is the optimal Δ_i in terms of ϵ_i?

(The result shows that the α_t in AdaBoost is the steepest descent choice for E_{exp} after getting h_t.)

(2) (5%) Suppose now that we want to pick the best i that greedily makes $E_{\text{exp}}(H)$ the smallest. That is, for a given vector $(\beta_1, \beta_2, \cdots, \beta_{L-1}, \beta_L)$, we want to solve

$$\min_{\Delta, i} \ \frac{1}{N} \sum_{n=1}^{N} \exp\left(-y_n \left(\sum_{\ell=1}^{L} \beta_\ell^{old} h_\ell(\mathbf{x}_n)\right) - y_n\big(\Delta \cdot h_i(\mathbf{x}_n)\big)\right).$$

If all $h \in \mathcal{H}$ satisfies

$$\frac{\sum_{n=1}^{N} u_n \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]}{\sum_{n=1}^{N} u_n} \leq \frac{1}{2},$$

show that the optimal

$$h_i = \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{n=1}^{N} u_n \cdot [\![y_n \neq h(\mathbf{x}_n)]\!] .$$

(The result shows that the h_t in AdaBoost is the optimal coordinate choice for E_{exp}.)

All the results can be extended to the case when $\mathcal{H}$ is of an infinite size as well. This problem shows that AdaBoost is optimizing a particular error function E_{exp} slowly (by coordinate).

7.5 Experiments with Adaptive Boosting (*)

(1) (10%) Implement the AdaBoost algorithm with decision stumps (i.e., use A_{ds} as A_b). Run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw7_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw7_test.dat`

Use a total of $T = 300$ iterations. Let $H_t(\mathbf{x}) = \text{sign}\left(\sum_{\tau=1}^{t} \alpha_\tau h_\tau(\mathbf{x})\right)$. Plot $E_{\text{in}}(H_t)$, $E_{\text{out}}(H_t)$ and $U^{(t)}$ (see the definition above) as functions of t on the same figure. Briefly state your findings.

7.6 Experiments with Unpruned Decision Tree (*)

(1) (10%) Implement the following simple decision tree algorithm (without pruning):

`function DecisionTree(`$\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$`) returns a tree`

- When every example in $\mathcal{D}$ shares the same y_n, return a constant hypothesis $h(\mathbf{x}) = y_n$.
- Learn a decision-stump branch function $b(\mathbf{x})$ that separates $\mathcal{D}$ to 2 parts:

$$b(\mathbf{x}) = \operatorname*{argmax}_{h_{s,i,\theta}} \left(\text{impurity}(\mathcal{D}) - \sum_{c=1}^{2} \frac{|\mathcal{D}^{(c)}|}{|\mathcal{D}|} \text{impurity}(\mathcal{D}^{(c)}) \right)$$

  where $\mathcal{D}^{(c)} = \{(\mathbf{x}_n, y_n) \colon b(\mathbf{x}_n) = c\}$ and the impurity is measured by Gini index.
- Obtain $H_c^{(L-1)} =$ `DecisionTree(`$\mathcal{D}^{(c)}$`)`
- Return

$$H^{(L)}(\mathbf{x}) = \sum_{c=1}^{2} [\![b(\mathbf{x}) = c]\!] \cdot H_c^{(L-1)}(\mathbf{x})$$

Run the algorithm on the following set for training:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw7_train.dat`

and the following set for testing:

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/data/hw7_test.dat`

Plot the training examples $(\mathbf{x}_n, y_n)$ and the decision boundary (where the prediction changes) in the same figure. In addition, show your binary tree $H^{(L)}$ (by graph or by writing down the if-then-else). Report $E_{\text{in}}(H^{(L)})$ and $E_{\text{out}}(H^{(L)})$. Briefly state your findings.

ML HW#7 R98922081 徐伍文

7.1

(1) $\vec{x} \in \mathbb{R}^{\infty} \in \mathbb{R}$, $|\vec{x}|<1$

Let $x = \vec{x}$, $x \in \mathbb{R}$

$\Rightarrow \phi(\vec{x}) = [1, x, x^2, x^3, \cdots]$

(2) $\gamma_1 > 0$, $\gamma_2 > 0$

$\Rightarrow \phi(\vec{x}) = [\sqrt{\gamma_1}\,\phi_1(\vec{x}) \quad \sqrt{\gamma_2}\,\phi_2(\vec{x})]$

$\phi(\vec{x})$ is composed of elements from $\phi_1(\vec{x})$ with multiplying $\sqrt{\gamma_1}$ and elements from $\phi_2(\vec{x})$ with multiplying $\sqrt{\gamma_2}$.

(3) $\phi_1(\vec{x}) = [\phi_{11}(\vec{x}), \phi_{12}(\vec{x}), \cdots, \phi_{1d_1}(\vec{x})]^{1\times d_1}$

$\phi_2(\vec{x}) = [\phi_{21}(\vec{x}), \phi_{22}(\vec{x}), \cdots, \phi_{2d_2}(\vec{x})]^{1\times d_2}$

$\Rightarrow \phi(\vec{x}) = [\phi_{11}(\vec{x})\phi_{21}(\vec{x}), \phi_{11}(\vec{x})\phi_{22}(\vec{x}), \cdots, \phi_{11}(\vec{x})\phi_{2d_2}(\vec{x}),$
$\phi_{12}(\vec{x})\phi_{21}(\vec{x}), \phi_{12}(\vec{x})\phi_{22}(\vec{x}), \cdots, \phi_{12}(\vec{x})\phi_{2d_2}(\vec{x}),$
$\cdots\cdots\cdots\cdots$
$\phi_{1d_1}(\vec{x})\phi_{21}(\vec{x}), \phi_{1d_1}(\vec{x})\phi_{22}(\vec{x}), \cdots, \phi_{1d_1}(\vec{x})\phi_{2d_2}(\vec{x})]^{1\times(d_1\times d_2)}$

$= \text{reshape}(\phi_1(\vec{x})^T\phi_2(\vec{x}), [1, d_1\times d_2])$

Element of $\phi(\vec{x})$ is the multiplication of element from $\phi_1(\vec{x})$ and element from $\phi_2(\vec{x})$ pairwisely.

(4) $K(\vec{x}, \vec{x}') = \begin{cases} 1, & \text{if } \vec{x} = \vec{x}' = \vec{x}_n \text{ for some } \vec{x}_n \in D \\ 0, & \text{otherwise} \end{cases}$

$$Q = I_N = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}_{N\times N}$$

(5) $K(\vec{x}, \vec{x}')$: valid kernel $\Leftrightarrow K(\vec{x}, \vec{x}') = \Phi(\vec{x})\cdot\Phi(\vec{x}')$

$\Rightarrow K(\vec{x}, \vec{x}) = \Phi(\vec{x})\cdot\Phi(\vec{x}) = \|\Phi(\vec{x})\|^2 \geq 0$

For $\vec{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, $K(\vec{x}, \vec{x}) = -(\vec{x}\cdot\vec{x})^2 - 2\vec{x}\cdot\vec{x}$

$= -9 + 6 = -3 < 0$ ✗

Therefore, $K(\vec{x}, \vec{x}') = -(\vec{x}\cdot\vec{x}')^2 + 2(\vec{x}\cdot\vec{x}')$ is NOT a valid kernel.

7.2.

(1) # input vectors = finite

For example: $\vec{x}_1 \sim \vec{x}_N$, then there are "at most" $2d(N-1)+2$ different decision stumps for $\mathcal{X}$, i.e. finitely-many.

($s=+1$ or $s=-1$; # dimensions; all "+1" or all "−1"; most number of θ)

For $\mathcal{X} \subseteq \{\mathbb{Z} \cap [-2,2]\}^3$, $h_{s,i,\theta}(\vec{x}) = \text{sign}(s \cdot \vec{x}[i] - \theta) = f(s,i,\theta,\vec{x})$

$h^{(1)} = f(1,1,-2.5,\vec{x})$ i.e. all "+1". $h^{(2)} = f(1,1,2.5,\vec{x})$, i.e. all "−1"

$h^{(3)} = f(1,1,1.5,\vec{x})$, $h^{(4)} = f(1,1,0.5,\vec{x})$, $h^{(5)} = f(1,1,-0.5,\vec{x})$,

$h^{(6)} = f(1,1,-1.5,\vec{x})$, $h^{(7)} = f(-1,1,-1.5,\vec{x})$, $h^{(8)} = f(-1,1,-0.5,\vec{x})$

$h^{(9)} = f(-1,1,0.5,\vec{x})$, $h^{(10)} = f(-1,1,1.5,\vec{x})$, $h^{(11)} = f(1,2,1.5,\vec{x})$,

$h^{(12)} = f(1,2,0.5,\vec{x})$, $h^{(13)} = f(1,2,-0.5,\vec{x})$, $h^{(14)} = f(1,2,-1.5,\vec{x})$

$h^{(15)} = f(-1,2,-1.5,\vec{x})$, $h^{(16)} = f(-1,2,-0.5,\vec{x})$, $h^{(17)} = f(-1,2,0.5,\vec{x})$

$h^{(18)} = f(-1,2,1.5,\vec{x})$, $h^{(19)} = f(1,3,1.5,\vec{x})$, $h^{(20)} = f(1,3,0.5,\vec{x})$

$h^{(21)} = f(1,3,-0.5,\vec{x})$, $h^{(22)} = f(1,3,-1.5,\vec{x})$, $h^{(23)} = f(-1,3,-1.5,\vec{x})$

$h^{(24)} = f(-1,3,-0.5,\vec{x})$, $h^{(25)} = f(-1,3,0.5,\vec{x})$, $h^{(26)} = f(-1,3,1.5,\vec{x})$

(2)
$$\begin{aligned}
K_{ds}(\vec{x},\vec{x}') &= \phi_{ds}(\vec{x}) \cdot \phi_{ds}(\vec{x}') \\
&= h_1(\vec{x})h_1(\vec{x}') + h_2(\vec{x})h_2(\vec{x}') + \cdots + h_{|H|}(\vec{x})h_{|H|}(\vec{x}') \\
&= \sum_i [\![h_i(\vec{x}) == h_i(\vec{x}')]\!] - \sum_j [\![h_j(\vec{x}) \neq h_j(\vec{x}')]\!] \quad -5 \\
&= (|H| - \sum_j [\![h_j(\vec{x}) \neq h_j(\vec{x}')]\!]) - \sum_j [\![h_j(\vec{x}) \neq h_j(\vec{x}')]\!] \\
&= |H| - 2\sum_j [\![h_j(\vec{x}) \neq h_j(\vec{x}')]\!] \\
&= |H| - 2\left(2 \times \sum_i |\vec{x}_i - \vec{x}_i'|\right) \quad +5
\end{aligned}$$

We may choose $C = -|H|$

(3) In SVM, pick some $\alpha_n > 0$ to compute b, i.e. $y_n\left(\sum_{m=1}^{N} y_m \alpha_m K(\vec{x}_m,\vec{x}_n) + b\right) = 1$

If we replace $K(\vec{x}_m,\vec{x}_n)$ with $K(\vec{x}_m,\vec{x}_n) + C$ for a constant C,

then we pick some $\alpha_n > 0$: $y_n\left(\sum_{m=1}^{N} y_m \alpha_m (K(\vec{x}_m,\vec{x}_n) + C) + b'\right) = 1$.

$$\Rightarrow y_n\left(\sum_{m=1}^{N} y_m \alpha_m K(\vec{x}_m,\vec{x}_n) + C\sum_{m=1}^{N} y_m \alpha_m + b'\right) = 1$$

$$\Rightarrow b' = b - C\sum_{m=1}^{N} y_m \alpha_m$$

$$g(\vec{x}) = \text{sign}\left(\sum_{m=1}^{N} y_m \alpha_m K(\vec{x}_m,\vec{x}) + b\right) \quad \left(b = b' + C\sum_{m=1}^{N} y_m \alpha_m\right)$$

$$= \text{sign}\left(\sum_{m=1}^{N} y_m \alpha_m K(\vec{x}_m,\vec{x}) + b' + C\sum_{m=1}^{N} y_m \alpha_m\right)$$

$$= \text{sign}\left(\sum_{m=1}^{N} y_m \alpha_m (K(\vec{x}_m,\vec{x}) + C) + b'\right) = g'(\vec{x})$$

7.3

(1) $u_n^{(0)} = \frac{1}{N}$

$u_n^{(1)} = u_n^{(0)} \cdot \exp(-\alpha_1 y_n h_1(\vec{x}_n)) = \frac{1}{N} \exp(-\alpha_1 y_n h_1(\vec{x}_n))$

$u_n^{(2)} = u_n^{(1)} \cdot \exp(-\alpha_2 y_n h_2(\vec{x}_n)) = \frac{1}{N} \exp(-\alpha_1 y_n h_1(\vec{x}_n)) \cdot \exp(-\alpha_2 y_n h_2(\vec{x}_n))$

$= \frac{1}{N} \exp[(-\alpha_1 y_n h_1(\vec{x}_n)) + (-\alpha_2 y_n h_2(\vec{x}_n))]$

.....

$u_n^{(t)} = \frac{1}{N} \exp[(-\alpha_1 y_n h_1(\vec{x}_n)) + (-\alpha_2 y_n h_2(\vec{x}_n)) + \cdots + (-\alpha_t y_n h_t(\vec{x}_n))]$

$= \frac{1}{N} \exp[-y_n \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\vec{x}_n)]$

$U^{(t)} = \sum_{n=1}^{N} u_n^{(t)} = \sum_{n=1}^{N} \frac{1}{N} \exp[-y_n \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\vec{x}_n)]$

$= \frac{1}{N} \sum_{n=1}^{N} \exp[-y_n \sum_{\tau=1}^{t} \alpha_\tau h_\tau(\vec{x}_n)]$ ✓

(2) $H(\vec{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\vec{x}))$

$\Rightarrow E_{in}(H) = \sum_{n=1}^{N} [\![y_n \neq H(\vec{x}_n)]\!] / N$

$U^{(T)} = \frac{1}{N} \sum_{n=1}^{N} \exp[-y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n)]$

For $y_n \neq H(\vec{x}_n)$ $\begin{cases} [\![y_n \neq H(\vec{x}_n)]\!] = 1 \\ \exp[-y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n)] \geq 1 \end{cases}$ $\because -y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n) \geq 0$

For $y_n = H(\vec{x}_n)$ $\begin{cases} [\![y_n \neq H(\vec{x}_n)]\!] = 0 \\ \exp[-y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n)] > 0 \end{cases}$

$\Rightarrow$ In each case : $[\![y_n \neq H(\vec{x}_n)]\!] \leq \exp[-y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n)]$

$\Rightarrow \frac{1}{N} \sum_{n=1}^{N} [\![y_n \neq H(\vec{x}_n)]\!] \leq \frac{1}{N} \sum_{n=1}^{N} \exp[-y_n \sum_{t=1}^{T} \alpha_t h_t(\vec{x}_n)]$

$\Rightarrow E_{in}(H) \leq U^{(T)}$ ✓

(3) $U^{(t)} = \sum_{n=1}^{N} u_n^{(t)}$ $\quad (u_n^{(t)} = u_n^{(t-1)} \cdot \exp(-\alpha_t y_n h_t(\vec{x}_n)))$

$= \sum_{n=1}^{N} u_n^{(t-1)} \cdot \exp(-\alpha_t y_n h_t(\vec{x}_n))$

$= e^{\alpha_t} \cdot \underbrace{\sum_{n=1}^{N} u_n [\![y_n \neq h_t(\vec{x}_n)]\!]}_{\epsilon_t \cdot \sum_{n=1}^{N} u_n^{(t-1)} = \epsilon_t \cdot U^{(t-1)}} + e^{-\alpha_t} \cdot \underbrace{\sum_{n=1}^{N} u_n [\![y_n = h_t(\vec{x}_n)]\!]}_{\sum_{n=1}^{N} u_n^{(t-1)} - \epsilon_t \cdot \sum_{n=1}^{N} u_n^{(t-1)} = (1-\epsilon_t) \cdot \sum_{n=1}^{N} u_n^{(t-1)} = (1-\epsilon_t) \cdot U^{(t-1)}}$

$= \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} [\epsilon_t \cdot U^{(t-1)}] + \sqrt{\frac{\epsilon_t}{1-\epsilon_t}} [(1-\epsilon_t) \cdot U^{(t-1)}]$

$= \sqrt{\epsilon_t(1-\epsilon_t)} \cdot U^{(t-1)} + \sqrt{\epsilon_t(1-\epsilon_t)} \cdot U^{(t-1)} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \cdot U^{(t-1)}$ ✓

(4) $0 \le \epsilon_t \le \epsilon < \frac{1}{2}$

$$\Rightarrow \epsilon(1-\epsilon) - \epsilon_t(1-\epsilon_t) = \epsilon - \epsilon^2 - \epsilon_t + \epsilon_t^2$$

$$= (\epsilon - \epsilon_t) - (\epsilon^2 - \epsilon_t^2)$$

$$= (\epsilon - \epsilon_t) - (\epsilon - \epsilon_t)(\epsilon + \epsilon_t)$$

$$= \underbrace{(\epsilon - \epsilon_t)}_{\epsilon \ge \epsilon_t \Rightarrow \epsilon - \epsilon_t \ge 0} \underbrace{[1-(\epsilon+\epsilon_t)]}_{\frac{1}{2} > \epsilon \ge \epsilon_t \Rightarrow \epsilon+\epsilon_t < 1 \Rightarrow 1-(\epsilon+\epsilon_t) > 0} \ge 0$$

$$\Rightarrow \epsilon(1-\epsilon) \ge \epsilon_t(1-\epsilon_t)$$

$$\Rightarrow \sqrt{\epsilon(1-\epsilon)} \ge \sqrt{\epsilon_t(1-\epsilon_t)} \quad \checkmark$$

(5) $\because e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

$$\Rightarrow \frac{1}{2} e^{-2(\frac{1}{2}-\epsilon)^2} \ge \frac{1}{2}[1 - 2(\tfrac{1}{2}-\epsilon)^2] \quad \because 0 \le \epsilon < \tfrac{1}{2}$$

$$= \frac{1}{2} - (\tfrac{1}{2}-\epsilon)^2 \quad \therefore \frac{[-2(\frac{1}{2}-\epsilon)^2]^2}{2!} + \frac{[-2(\frac{1}{2}-\epsilon)^2]^3}{3!} > 0$$

$$= \frac{1}{2} - \frac{1}{4} + \epsilon - \epsilon^2 \quad \frac{[-2(\frac{1}{2}-\epsilon)^2]^4}{4!} + \frac{[-2(\frac{1}{2}-\epsilon)^2]^5}{5!} > 0$$

$$= \frac{1}{4} + \epsilon(1-\epsilon) \quad \cdots\cdots\cdots$$

Let $t = \sqrt{\epsilon(1-\epsilon)}$

$$\Rightarrow [\tfrac{1}{4} + \epsilon(1-\epsilon)] - \sqrt{\epsilon(1-\epsilon)} = t^2 - t + \tfrac{1}{4}$$

$$= (t - \tfrac{1}{2})^2 \ge 0$$

$$\Rightarrow \frac{1}{4} + \epsilon(1-\epsilon) \ge \sqrt{\epsilon(1-\epsilon)}$$

$$\Rightarrow \frac{1}{2} e^{-2(\frac{1}{2}-\epsilon)^2} \ge \sqrt{\epsilon(1-\epsilon)} \quad \checkmark$$

(6) From (2) : $U^{(T)} = 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot U^{(T-1)}$

$$= 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot (2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdot U^{(T-2)})$$

$$= \cdots\cdots$$

$$= 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot 2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdots\cdots 2\sqrt{\epsilon_1(1-\epsilon_1)} \cdot U^{(0)}, \quad U^{(0)} = \sum_{n=1}^{N} u_n^{(0)} = \sum_{n=1}^{N} \frac{1}{N} = 1$$

$$= 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot 2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdots\cdots 2\sqrt{\epsilon_1(1-\epsilon_1)}$$

From (3),(4) : $2\sqrt{\epsilon_t(1-\epsilon_t)} \le 2\sqrt{\epsilon(1-\epsilon)} \le \exp[-2(\frac{1}{2}-\epsilon)^2]$, for $t \ge 1$, $0 \le \epsilon_t \le \epsilon < \frac{1}{2}$

$$\Rightarrow U^{(T)} = 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot 2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdots\cdots 2\sqrt{\epsilon_1(1-\epsilon_1)}$$

$$\le e^{-2(\frac{1}{2}-\epsilon)^2} \cdot e^{-2(\frac{1}{2}-\epsilon)^2} \cdot \cdots\cdots e^{-2(\frac{1}{2}-\epsilon)^2} = [e^{-2(\frac{1}{2}-\epsilon)^2}]^T = e^{-2T(\frac{1}{2}-\epsilon)^2}$$

$$\Rightarrow U^{(T)} \le \exp[-2T(\tfrac{1}{2}-\epsilon)^2] \quad \checkmark$$

(7) $T = O(\log N) \Rightarrow$ We can replace T with $C \cdot \ln N$, C: constant, $C > 0$

$$E_{in}(H) \le U^{(T)} \le \exp[-2(\tfrac{1}{2}-\epsilon)^2 \cdot C \cdot \ln N]$$

$$= \exp[\ln N^{-2(\frac{1}{2}-\epsilon)^2 C}]$$

$$= N^{-2(\frac{1}{2}-\epsilon)^2 C}$$

For $N^{-2(\frac{1}{2}-\epsilon)^2 C} < N^{-1}$ $\because N \ge 1$

$$\Rightarrow -2(\tfrac{1}{2}-\epsilon)^2 C < -1$$

$$\Rightarrow 2(\tfrac{1}{2}-\epsilon)^2 C > 1$$

$$\Rightarrow C > \frac{1}{2(\frac{1}{2}-\epsilon)^2}$$

Therefore, we can make $C > \frac{1}{2(\frac{1}{2}-\epsilon)^2}$ such that $E_{in}(H) < \frac{1}{N}$, i.e. $E_{in}(H) = 0$ ✓

7.4

(1) $\Delta_i = \arg\min_{\Delta} \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n(\sum_{l=1}^{L} \beta_l^{old} h_l(\vec{x}_n)) - y_n(\Delta \cdot h_i(\vec{x}_n)))$

$$\Rightarrow \frac{d}{d\Delta}[\frac{1}{N}\sum_{n=1}^{N} \exp(-y_n(\sum_{l=1}^{L} \beta_l^{old} h_l(\vec{x}_n)) - y_n(\Delta \cdot h_i(\vec{x}_n)))] = 0$$

$$\Rightarrow \frac{d}{d\Delta}[\sum_{n=1}^{N} u_n \cdot e^{-y_n h_i(\vec{x}_n)\Delta}] = 0$$

$$\Rightarrow \sum_{n=1}^{N} u_n(-y_n h_i(\vec{x}_n)) \cdot e^{-y_n h_i(\vec{x}_n)\Delta_i} = 0$$

$$\Rightarrow e^{\Delta_i} \cdot \sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!] - e^{-\Delta_i} \sum_{n=1}^{N} u_n [\![y_n == h_i(\vec{x}_n)]\!] = 0$$

$$\Rightarrow e^{\Delta_i} \cdot \sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!] = e^{-\Delta_i} \cdot \sum_{n=1}^{N} u_n [\![y_n == h_i(\vec{x}_n)]\!]$$

$$\Rightarrow e^{2\Delta_i} = \frac{\sum_{n=1}^{N} u_n [\![y_n == h_i(\vec{x}_n)]\!]}{\sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!]} = \frac{\sum_{n=1}^{N} u_n - \sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!]}{\sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!]} = \frac{(1-\epsilon_i)\sum_{n=1}^{N} u_n}{\epsilon_i \cdot \sum_{n=1}^{N} u_n} = \frac{1-\epsilon_i}{\epsilon_i}$$

$$\Rightarrow 2\Delta_i = \ln \frac{1-\epsilon_i}{\epsilon_i}$$

$$\Rightarrow \Delta_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$$ ✓

(2) $\min_{\Delta, h_i} \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n(\sum_{l=1}^{L} \beta_l^{old} h_l(\vec{x}_n)) - y_n(\Delta \cdot h_i(\vec{x}_n)))$

$$\Leftrightarrow \min_{h_i} \sum_{n=1}^{N} \frac{1}{N} e^{-y_n(\sum_{l=1}^{L} \beta_l^{old} h_l(\vec{x}_n))} \cdot e^{-y_n h_i(\vec{x}_n) \cdot [\frac{1}{2}\ln \frac{1-\epsilon_i}{\epsilon_i}]}$$

$$\Leftrightarrow \min_{h_i} \sum_{n=1}^{N} u_n \cdot e^{-y_n h_i(\vec{x}_n)[\frac{1}{2}\ln \frac{1-\epsilon_i}{\epsilon_i}]}$$

$$\Leftrightarrow \min_{h_i} e^{\frac{1}{2}\ln \frac{1-\epsilon_i}{\epsilon_i}} \cdot \sum_{n=1}^{N} u_n [\![y_n \ne h_i(\vec{x}_n)]\!] + e^{-\frac{1}{2}\ln \frac{1-\epsilon_i}{\epsilon_i}} \cdot \sum_{n=1}^{N} u_n [\![y_n == h_i(\vec{x}_n)]\!]$$

$$\Leftrightarrow \min_{h_i} \sqrt{\frac{1-\epsilon_i}{\epsilon_i}} \cdot (\epsilon_i \cdot \sum_{n=1}^{N} u_n) + \sqrt{\frac{\epsilon_i}{1-\epsilon_i}} [(1-\epsilon_i)\sum_{n=1}^{N} u_n]$$

$$\Leftrightarrow \min_{\lambda} \; 2\sqrt{\epsilon_\lambda(1-\epsilon_\lambda)} \sum_{n=1}^{N} u_n$$

$$\Leftrightarrow \min_{\lambda} \; \sqrt{\epsilon_\lambda(1-\epsilon_\lambda)}$$

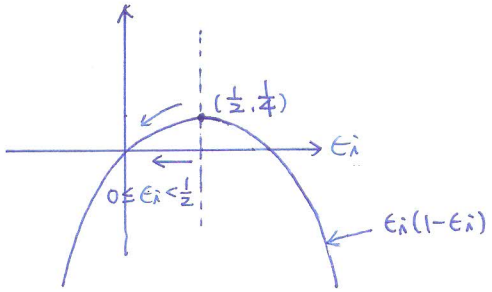
$$\Leftrightarrow \min_{\lambda} \; \epsilon_\lambda(1-\epsilon_\lambda)$$

$$\Leftrightarrow \min_{\lambda} \; \epsilon_\lambda$$

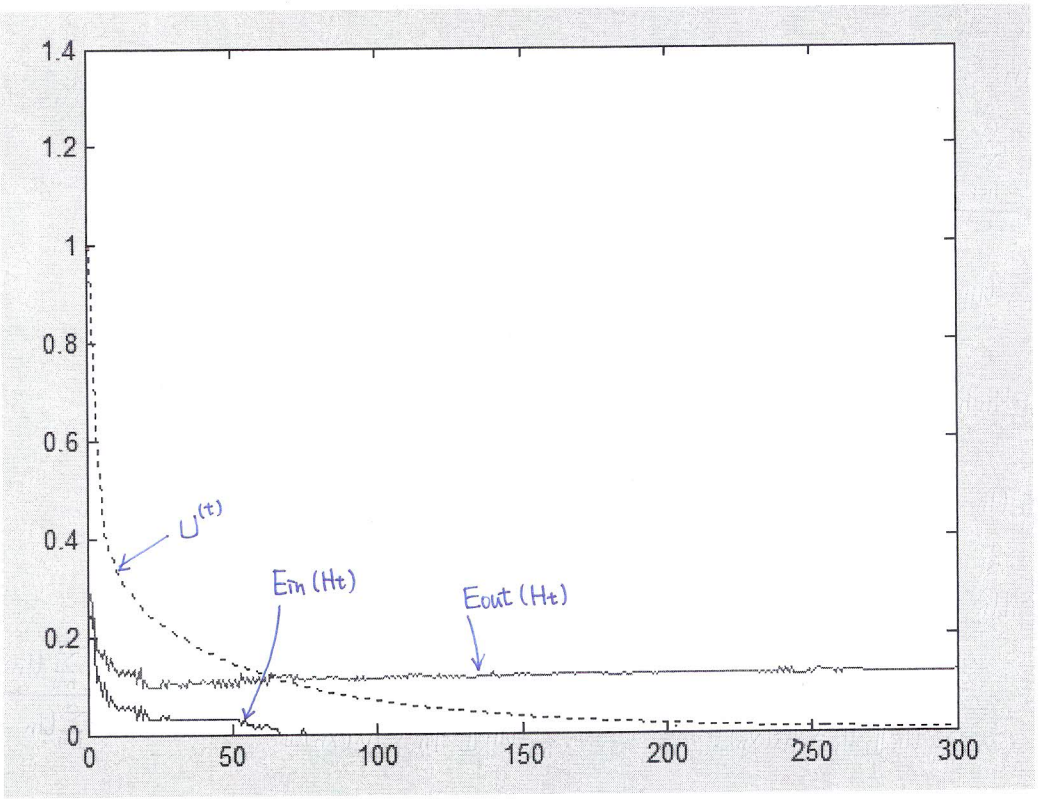
$$\Leftrightarrow \min_{\lambda} \; \frac{\sum_{n=1}^{N} u_n [\![y_n \neq h_\lambda(\vec{x}_n)]\!]}{\sum_{n=1}^{N} u_n}$$

$$\Leftrightarrow \min_{\lambda} \; \sum_{n=1}^{N} u_n [\![y_n \neq h_\lambda(\vec{x}_n)]\!]$$

$$\Rightarrow h_\lambda = \operatorname*{argmin}_{h \in H} \sum_{n=1}^{N} u_n [\![y_n \neq h_\lambda(\vec{x}_n)]\!]$$

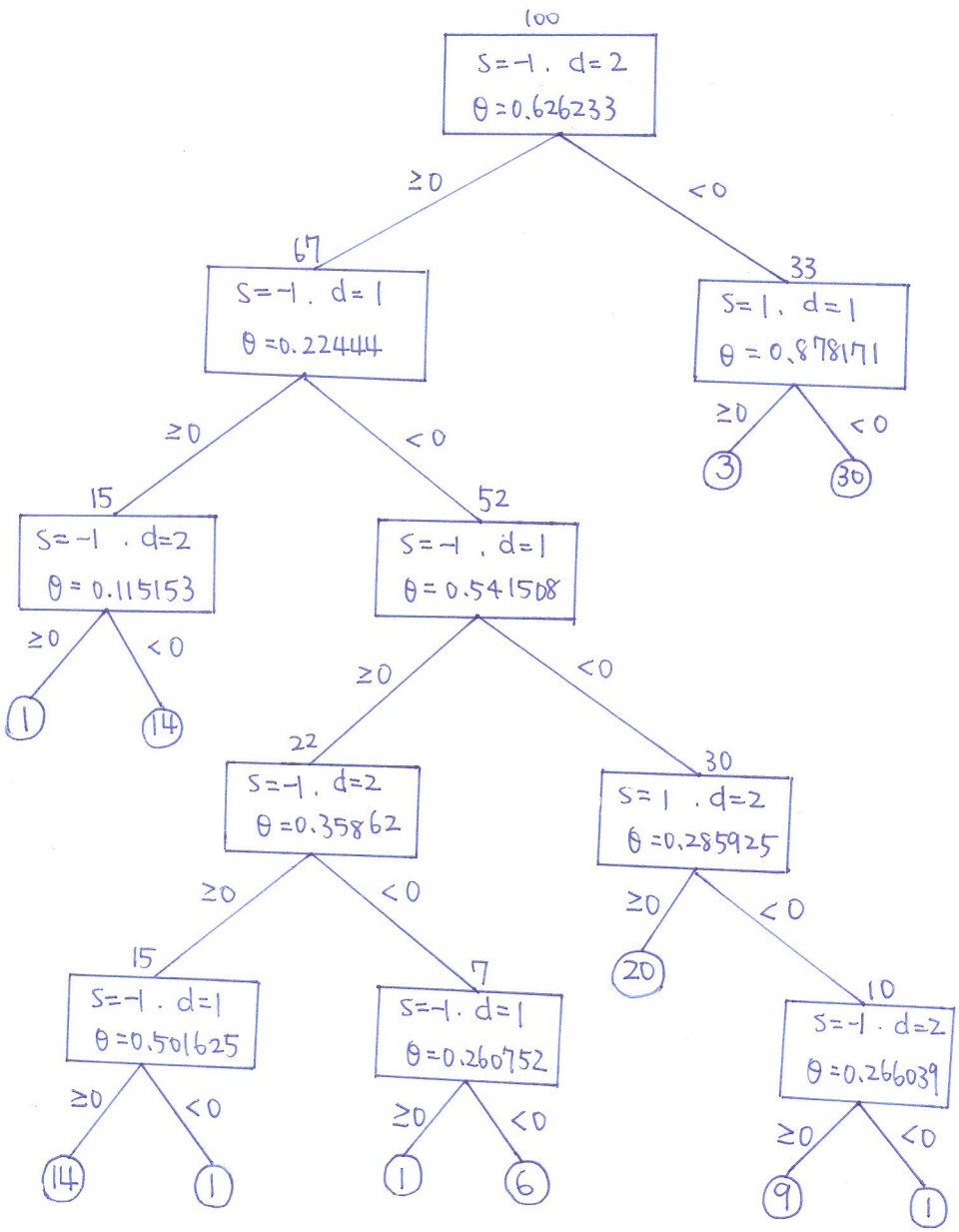


7.5



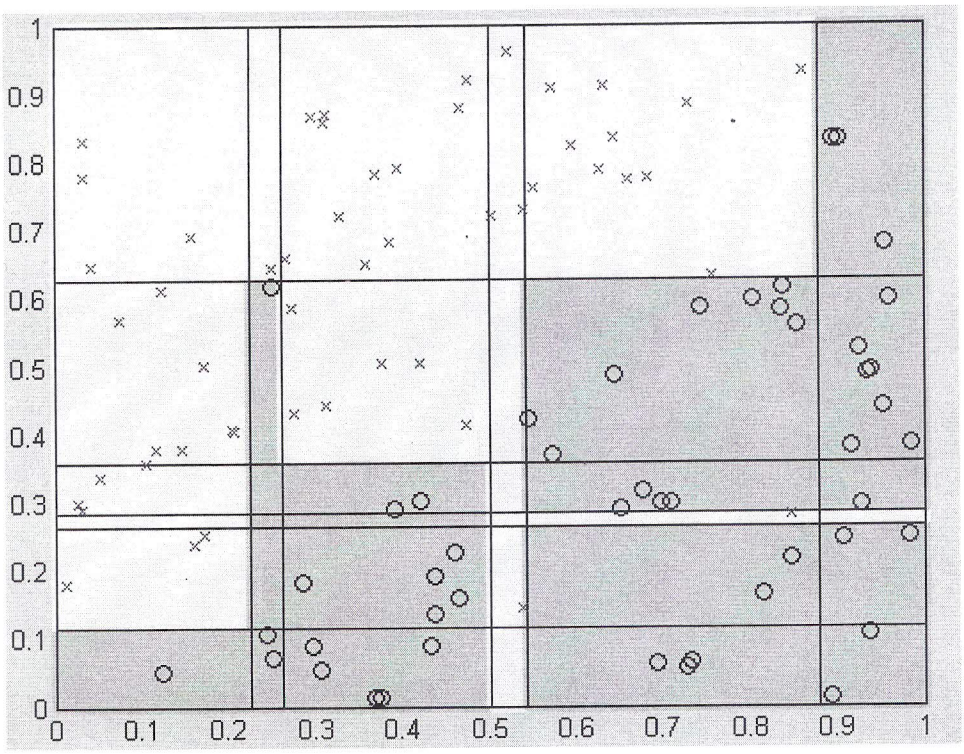
From the figure above, we can see that E_{in} became zero after about 80 iterations. And $U^{(t)}$ is indeed an upper bound of $E_{in}(H_t)$. Besides, even though $E_{in}(H_t) = 0$, $E_{out}(H_t)$ seemed to have the tendency towards bigger with more iterations. We may have the problem of overfitting after many iterations in AdaBoost.

7.6.



$E_{in}(H^{(4)}) = 0$

$E_{out}(H^{(4)}) = 0.126$



The unpruned decision tree algo will try to fit every training example, so we can always get $E_{in}(H^{(L)}) = 0$. However, sometimes some examples may be the noise information, the unpruned decision tree still have to find a boundry for these noise data such that the boundry will predict testing examples into the wrong class. So it causes overfitting problem. Even we have $E_{in}=0$, E_{out} may be very big.

Homework #8

TA in charge: Yao-Nan Chen

RELEASE DATE: 12/27/2010

DUE DATE: 01/03/2011, 4:00 pm IN CLASS

TA SESSION: 12/30/2010, 6:00 pm IN R110

The homework is OPTIONAL. That is, if you choose to turn it in, your homework score would be calculated over HW1 to HW8; otherwise your homework score would be calculated over HW1 to HW7. In both cases, we will use the equation

$$\frac{\text{your best homework} * 1.5 + \text{your worse homework} * 0.5 + \sum(\text{your other homework})}{\#\text{ of homework}}$$

Please make a choice BEFORE the TA grades HW8. If you choose to not turn in HW8, we still encourage you to discuss the solutions with your classmates or TAs.

Unless granted by the instructor in advance, you must turn in a hard copy of your solutions (without the source code) for all problems. For problems marked with (*), please follow the guidelines on the course website and upload your source code to designated places.

Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.

Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

You should write your solutions in English with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.

8.1 A Bayesian View of Logistic Regression

(1) (10%) Prove that logistic regression (see Problem 4.6) equivalently minimizes the following error function:

$$E_{ce}(\mathbf{w}) = -\sum_{n=1}^{N}\left(\frac{1+y_n}{2}\ln\frac{1+\tanh(\frac{1}{2}\mathbf{w}\bullet\mathbf{x}_n)}{2} + \frac{1-y_n}{2}\ln\frac{1-\tanh(\frac{1}{2}\mathbf{w}\bullet\mathbf{x}_n)}{2}\right).$$

The error function is usually called the *cross-entropy*.

(2) (10%) Assume that the universe generates an example $(\mathbf{x}, y)$ by the following procedure:

(a) generate $\mathbf{x}$ from some probability density function $P(\mathbf{x})$

(b) use some fixed $\mathbf{w}^u$ (including $w_0^u = 1$) to evaluate $\rho = \mathbf{w}^u \bullet \mathbf{x}$

(c) evaluate $Q_+ = \exp\left(\frac{\rho}{2}\right)$ and $Q_- = \exp\left(-\frac{\rho}{2}\right)$

(d) generate $y \in \{+, -\}$ with the probability distribution $Q_y/(Q_+ + Q_-)$

If each $(\mathbf{x}_n, y_n)$ within $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is generated i.i.d from the procedure above, what is the likelihood $P(\mathcal{D}|\,\mathbf{w} = \mathbf{w}^u)$?

(3) (10%) Prove that logistic regression equivalently gives the maximum likelihood estimate of $\mathbf{w}^u$.

(4) (10%) If we take the solution $\hat{\mathbf{w}}$ from logistic regression as our estimate of the underlying $\mathbf{w}^u$. Show that $\frac{Q_+}{Q_+ + Q_-}$ can be estimated by $\frac{1}{1+\exp(-\hat{\mathbf{w}}\bullet\mathbf{x})}$.

Note that the cross-entropy error function is often used to design other learning algorithms (for example, some Neural Networks). In addition, the logistic function in (4) is often used to obtain "probability estimates" of linear classifiers.

8.2 A Bayesian View of Linear Regression

(1) (10%) Assume that the universe generates an example $(\mathbf{x}, y)$ by the following procedure:

(a) generate $\mathbf{x}$ from some probability density function $P(\mathbf{x})$

(b) use some fixed $\mathbf{w}^u$ (including $w_0^u = 1$) to evaluate $\rho = \mathbf{w}^u \bullet \mathbf{x}$

(c) generate $y \in \mathbb{R}$ from ρ by the probability density function $P(y|\,\rho) = \frac{1}{\sqrt{2\pi}} \exp\left(-(y-\rho)^2\right)$

If each $(\mathbf{x}_n, y_n)$ within $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is generated i.i.d from the procedure above, what is the likelihood $P(\mathcal{D}|\,\mathbf{w} = \mathbf{w}^u)$?

(2) (10%) Prove that least-square linear regression (see Problem 4.5) equivalently gives the maximum likelihood estimate of $\mathbf{w}^u$.

(3) (10%) Assume that the universe generates an example $(\mathbf{x}, y)$ by the following procedure:

(a) generate some $\mathbf{w}^u$ from

$$P(\mathbf{w}^u) = \frac{1}{(\sqrt{2\pi})^{d+1} \cdot \sigma^{d+1}} \cdot \exp\left(-\frac{\|\mathbf{w}^u\|^2}{2\sigma^2}\right)$$

(b) generate $\mathbf{x}$ from some probability density function $P(\mathbf{x})$

(c) use the $\mathbf{w}^u$ to evaluate $\rho = \mathbf{w}^u \bullet \mathbf{x}$

(d) generate $y \in \mathbb{R}$ from ρ by the probability density function $P(y|\,\rho) = \frac{1}{\sqrt{2\pi}} \exp\left(-(y-\rho)^2\right)$

If each $(\mathbf{x}_n, y_n)$ within $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is generated i.i.d from the procedure above, what is the posterior probability $P(\mathbf{w} = \mathbf{w}^u|\,\mathcal{D})$?

(4) (10%) Prove that regularized linear regression (see Problem 5.3) equivalently gives the maximum posterior estimate of $\mathbf{w}^u$. What is the relationship between λ (in Problem 5.3) and σ (here)?

(5) (10%) Prove or disprove that on any new input vector $\mathbf{x}^{test}$, the prediction (see Problem 5.3)

$$g(\mathbf{x}^{test}) = \mathbf{w}_{reg}(\lambda) \bullet \mathbf{x}^{test}$$

happens to correspond to the Bayes estimate (in the regression sense) with respect to the posterior probability derived above.

(6) (10%) Assume that you have a strong *prior* belief that the universe generates an example $(\mathbf{x}, y)$ by the following procedure:

(a) generate some $\mathbf{w}^u$ that is similar to a "prior" target $\hat{\mathbf{w}}$ by

$$P(\mathbf{w}^u) = \frac{1}{(\sqrt{2\pi})^{d+1} \cdot \sigma^{d+1}} \cdot \exp\left(-\frac{\|\mathbf{w}^u - \hat{\mathbf{w}}\|^2}{2\sigma^2}\right)$$

(b) generate $\mathbf{x}$ from some probability density function $P(\mathbf{x})$

(c) use the $\mathbf{w}^u$ to evaluate $\rho = \mathbf{w}^u \bullet \mathbf{x}$

(d) generate $y \in \mathbb{R}$ from ρ by the probability density function $P(y|\,\rho) = \frac{1}{\sqrt{2\pi}} \exp\left(-(y-\rho)^2\right)$

If each $(\mathbf{x}_n, y_n)$ within $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ is generated i.i.d from the procedure above, what is the posterior probability $P(\mathbf{w} = \mathbf{w}^u|\,\mathcal{D})$?

(7) (Bonus 5%) Derive a closed-form solution that gives the maximum posterior estimate of $\mathbf{w}^u$ in Problem 8.2(6).

Final Project

RELEASE DATE: 11/26/2010

DUE DATE: 01/10/2011, **3:10 pm** IN CLASS

TA SESSION: by appointment

Unless granted by the instructor in advance, no late submissions will be allowed.

You should write your report in English with the common math notations introduced in class. We do not accept reports written in any other languages.

Introduction

The main theme of the final project is a machine learning competition. Imagine that you are a manager who leads a research team in a data analysis company. Recently, your company receives an important case that is worth millions of dollars. Then, the board of directors asks each research team to study some machine learning approaches for dealing with the case in order to provide concrete recommendations. To get more year-end bonus and future research funds, you have to offer a comprehensive report based on your professional expertise. The report will be evaluated not only by the prediction performance of the recommended approaches, but also by the reasoning behind your recommendations.

Data Sets

The case received by your company contains a big data set of size 50000. The board has decided to reserve 10000 of them as test examples, and *you are not allowed to peep the true answers of those.* The following file contains the examples without the answers (labels) in a zipped format.

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/proj_test.zip`

The other 40000 is taken as the training set that you can freely use.

`http://www.csie.ntu.edu.tw/~htlin/course/ml10fall/doc/proj_train.zip`

The data sets above are processed from the original "KDDCup2009" data set that you can get on public websites. Each line of the training set represents an example that takes the form:

$$y_n \quad , \mathbf{x}_n[1], \mathbf{x}_n[2] \quad \ldots, \mathbf{x}_n[190]$$

Here each $\mathbf{x}_n[i]$ is either a number or ?, which indicates a *missing value.* Also, $y_n \in \{1, 2, 3, 4\}$—that is, the task belongs to multiclass classification with possibly missing features. Each line of the test set represents an example without the y part.

One final remark: To maximize the level of fairness, you are not allowed to download the original data set from the website at any time.

Survey Report

You are asked by the board to study at least THREE machine learning approaches using the training set above. Then, you should make a comparison of those approaches according to some different perspectives, such as accuracy, efficiency, scalability, popularity, and interpret-ability. Based on the results of your comparison, you are asked to choose THE BEST ONE of those approaches as your final recommendation, and provide the "cons and pros" of the choice.

The survey report should be less than or equal to **six** A4-pages with readable font sizes and formats. The one most important criterion for evaluating your report is reproducibility. Thus, in addition to the outlines above, you should also describe how you pre-process your data; introduce the approaches you tried and provide specific references, especially for those approaches that we didn't cover in class; list your experimental settings and the parameters you used (or chose) clearly. Other criteria for evaluating your survey report would include, but are not limited to, clarity, strength of your reasoning, "correctness" in using machine learning techniques, the work loads of team members, and properness of citations.

For grading purposes, a minor but required part in your survey report for a two-people team (see the rules below) is how you balance your work loads.

Competition

To heat things up, the board has decided to set up an in-company competition. In the competition, the goal is simply to beat other teams' approaches in terms of *weighted accuracy*, which is defined as

$$E_{\text{test}}(g) = \underset{(\mathbf{x},y)}{\text{average}}\left(\left(1 \cdot [\![y = 1]\!] + 10 \cdot [\![y = 2]\!] + 10 \cdot [\![y = 3]\!] + 50 \cdot [\![y = 4]\!]\right)[\![g(\mathbf{x}) \neq y]\!]\right).$$

In other words, an example with $y = 4$ is 50 times more important than an example with $y = 1$.

The submission site would be announced in early December, 2010. Each team can freely submit the predictions on all 10000 test examples as an entry for each track. But use your submissions wisely—*you do not want to leave the board with a bad impression that you just want to "query" or "overfit" the test examples.* After submitting, there will be a score board for each track showing the accuracy evaluated on a randomly chosen (but fixed) 5000 out of the 10000 test examples. The board will secretly evaluates you on the other 5000.

The competition ends at noon on 01/09/2011. We'll have a mini-ceremony to honor the best team(s).

Misc Rules

Report: Please upload your report electronically on CEIBA. You do not need to submit a hard-copy.
Teams: By default, you are asked to work as a team of size TWO. A one-person team is allowed only if you are willing to be as good as a two-people team. It is expected that both team members share balanced work loads. Any form of unfairness in a two-people team, such as the intention to cover the other member's work, is considered a violation of the honesty policy and will cause both members to receive zero score.

Algorithms: You can use any algorithms, regardless of whether they were taught in class.

Packages: You can use any software package for the purpose of experiments, but please provide proper references in your report for reproducibility.

Source Code: You do not need to upload your source code for the final project. Nevertheless, please keep your source code until 02/28/2011 for the graders' possible inspections.

Grade: The final project is worth 300 points. That is, it is equivalent to three usual homework sets. At least 270 of them would be reserved for the report. The other 30 may depend on some minor criteria such as your competition results, your work loads, etc..

Collaboration: The general collaboration policy applies. In addition to the competitions, we still encourage collaborations and discussions between different teams.

Data Usage: You can use only the data sets provided in class for your experiments, and you should use the data sets properly. Getting other forms of the data sets (such as the original one on the website) is strictly prohibited and is considered a serious violation of the honesty policy. Using any tricks to query the labels of the test set is strictly prohibited, too.