L1: Introduction

Tuesday, February 7, 2017 1:09 PM

6.036 - Introduction to Machine Learning

Grading:

10%	Homework
30%	Projects
30%	Midterm
30%	Final :(

Website, Stellar, Piazza

- HW #0 due this Friday

R1 Background

Friday, February 10, 2017 11:02 AM

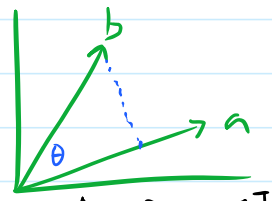
Notes on Stellar

Dot Product

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$

$$\langle \vec{a}, \vec{b} \rangle = \|\vec{a}\| \|\vec{b}\| \cos(\theta)$$

$$a^T b$$



$\vec{a} = [3, 1]^T$

$\vec{b} = [1, 2]^T$

Projection

Scalar projection: $AB = \|\vec{b}\| \underbrace{\cos(\theta)}_{\text{can use dot product to find}}$

$$\|a\| = \sqrt{3^2 + 1^2} = \sqrt{10} \qquad \|b\| = \sqrt{5}$$

$$\Rightarrow AB = \|\vec{b}\| \cos(\theta) = \cancel{\|\vec{b}\|} \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cancel{\|\vec{b}\|}}$$

Vector: $\vec{AB} = AB \cdot$ unit vector in direction $\vec{AB}$

$$= AB \cdot \frac{\vec{a}}{\|a\|}$$

make unit

Planes

Recall: can determine a line with:

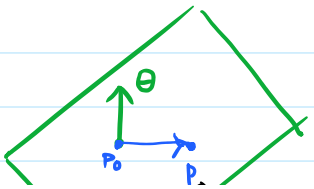
- 2 points
- 1 point + slope
- Vector $\perp$ to line + point

$\Rightarrow$ plane: uniquely determined by point in plane + vector $\perp$ to plane

P_0 $\quad$ $\vec{\theta}$

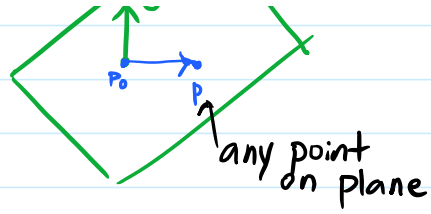
Find eqn of plane w/ P_0 and $\vec{\theta}$

$$P_0 = [\underset{x_0}{1}, \underset{y_0}{2}, \underset{z_0}{5}] \qquad \vec{\theta} = [2, 3, 4]$$



$P_0 = [1, 2, 5]$ $\quad \theta = [2, 3, 4]$

$x_0 \quad y_0 \quad z_0$



$$\overrightarrow{PP_0} \perp \theta \Rightarrow \overrightarrow{PP_0} \cdot \theta = 0$$

$$[x-x_0,\ y-y_0,\ z-z_0] \cdot [\theta_1, \theta_2, \theta_3] = 0$$

$$\Downarrow$$

$$\theta_1(x-x_0) + \theta_2(y-y_0) + \theta_3(z-z_0) = 0$$

$$\theta_1 x + \theta_2 y + \theta_3 z - \underbrace{(x_0\theta_1 + y_0\theta_2 + z_0\theta_3)}_{\vec{P_0} \cdot \vec{\theta} = \vec{\theta_0}}$$

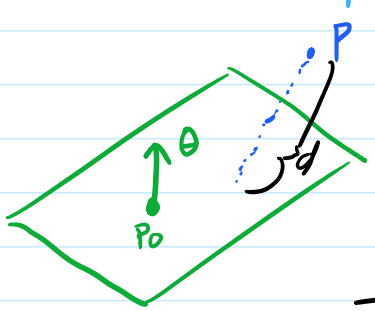
$$\boxed{\text{General Form: } \vec{\theta}\vec{P} + \theta_0 = 0}$$

$$[x-1,\ y-2,\ z-5][2, 3, 4] = 0$$

$$2(x-1) + 3(y-2) + 4(z-5) = 0$$

$$2x + 3y + 4z + 28 = 0$$

Distance between point and plane



signed distance between p and plane = scalar projection of $\overrightarrow{P-P_0}$ onto $\vec{\theta}$

$$= \frac{\overrightarrow{P-P_0} \cdot \vec{\theta}}{\|\vec{\theta}\|}$$

$$= \frac{[x-x_0,\ y-y_0,\ z-z_0] \cdot [\theta_1, \theta_2, \theta_3]}{\|\vec{\theta}\|}$$

$$= \frac{\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 - x_0\theta_1 - y_1\theta_2 - z_0\theta_3}{\|\vec{\theta}\|}$$

$$= \frac{\vec{\theta}\vec{P} + \vec{\theta}_0}{\|\theta\|}$$

$\Rightarrow$ $\boxed{\text{Given plane } \vec{\theta}x + \vec{\theta}_0 = 0 \quad \text{signed distance from point } p = \frac{\theta^T p + \vec{\theta}_0}{\|\theta\|}}$

Optimization

Gradient $\Leftrightarrow$ slope (vector) in more dimentions

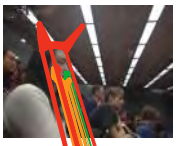
$f(x_1, x_2 \dots x_n)$

$$\nabla f(x_1, x_2 \dots x_n) = \left[\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2}, \frac{\delta f}{\delta x_3}, \dots, \frac{\delta f}{\delta x_n}\right]$$

$\boxed{\text{Gradient points in the direction of greatest increase of function}}$

L2: Linear Classification

Tuesday, February 14, 2017 1:07 PM



Labels / outputs : $x \in \{-1, 1\}$

Feature vectors / inputs: $x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^d$

Training Set : $= \{(x^{(i)}, y^{(i)}), \; i = 1 \ldots n\}$

Classifier/hypothesis/separator : $h : \mathbb{R}^d \to \{+1, -1\}$

Training error:

$$\varepsilon_n(h) = \frac{1}{n} \sum_{i=1}^{\infty} \left[[h(x^{(i)}) \neq y^{(i)}] \right]$$

Test/ Generalization error

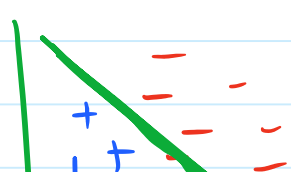
$$\varepsilon(h) = \frac{1}{n'} \sum_{i=n+1}^{n+n'} \left[[h(x^{(i)}) \neq y^{(i)}] \right] \quad (n' \to \infty)$$

Linear Separation

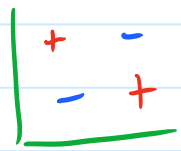
A set $S_n = \{(x^{(i)}, y^{(i)}) \; i = 1, \ldots n\}$

is linearly separable if $\exists \; \hat{\theta}, \hat{\theta}_0$ such that

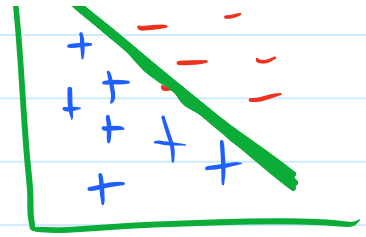
$$y^{(i)}(\hat{\theta} x^{(i)} + \hat{\theta}_0) > 0$$



$y \cdot (\theta x + \theta_0) > 0$



NOT Separable



Separable

Perceptron Algorithm

training set

$$S_n = \{ x^{(i)}, y^{(i)}, i = 1, n \} \Rightarrow \theta, \theta_0$$

INPUT OUTPUT

Code:

$\theta = 0$

$\theta_0 = 0$

cycle through examples

if mistake

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

$\theta_0 = \theta_0 + y^{(i)}$

L3: Maximum Margin Hyperplane

Thursday, February 16, 2017 1:04 PM

Online Algorithms

Prediction Game

L: $\theta^{(0)} = 0\ (vector)$

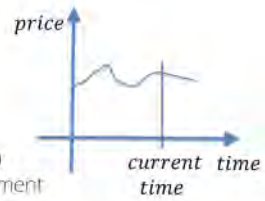
Round k

N: selects $x^{(i)} \in R^d$

L: predicts $sign(\theta^{(k-1)} \cdot x^{(i)})$

N: reveals $y^{(i)}$

L: suffers Loss $(\overbrace{y^{(i)}\theta^{(k-1)} \cdot x^{(i)}}^{\text{agreement}})$, updates θ



Goal: to minimize the overall loss

Exam 1 Review

Tuesday, March 21, 2017 7:01 PM

Supervised learning - Training with labled data

- 3 problems

Classification : feature vector → $\{-1, 1\}$

Regression : feature vector → $\mathbb{R}$ ← real number

Recommender :

$$\begin{array}{cc} & \text{items} \\ \text{user} & \begin{bmatrix} x & x & ? \\ x & ? & x \\ x & x & ? \end{bmatrix} \end{array}$$

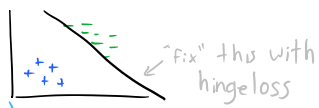
Outline

- = covered in this review

Classification

Perceptron: correctly classifies data if linearly separable

BAD: Can give different boundries based on ordering of points



"fix" this with hingeloss

SVM: (hingeloss)

Regression

- Linear Regression

SGD + closed form

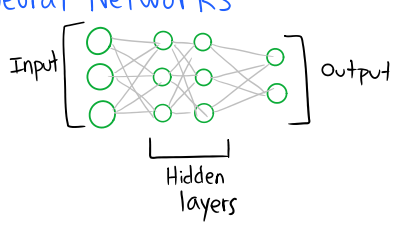
Non linear methods

- Non linear feature mappings
- Kernels
- Neural Nets

Recommender

- Low Rank Matrix Factorization

Neural Networks



- Each node calculates weighted sum of input plus a bias
  → feeds into function ← sigmoid, tanh

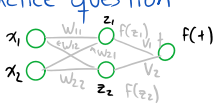
Stochastic Gradient Descent

- Backpropagation

Multi-way classification

- softmax $P(y=j) = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_k}}$

Practice question



f = ReLU

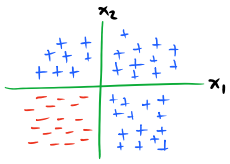
- 2 hidden units
  ⇒ learn 2 different boundries

Q: $f(z_1)$? $f(z_2)$? $f(t)$?

$f(z_1) = \max(0, x_1 W_{11} + x_2 W_{21})$

$f(z_2) = \max(0, x_1 W_{12} + x_2 W_{22})$

$f(t) = \max(0, V_1 f(z_1) + V_2 f(z_2))$



$f(t) > 0 \Rightarrow +$ $f(t) = 0 \Rightarrow -$

Q: what to set parameters for

-2 hidden units ⇒ 2 boundries

Q: what to set parameters for

- 2 hidden units ⇒ 2 boundries

Have one learn and another

⇒ $W_{11} = 1$ $W_{21} = 0$

$W_{12} = 0$ $W_{22} = 1$

- combine them

$V_1 = V_2 = 1$

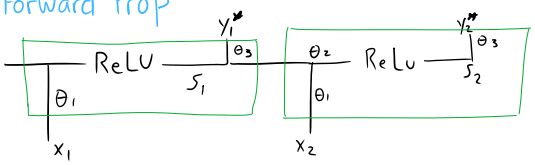
1	2
0	2

⇒ learned a non-linear boundry ✓

Recurrent Neural Networks RNN

Sample Problem

Forward Prop



$\theta_1 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ $\theta_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\theta_2 = -2$ $x_1 = 2$ $x_2 = -3$

$y_1^* = \theta_3 S_1$ $S_1 = \text{ReLU}(\theta_1 x_1)$

$y_2^* = \theta_3 S_2$ $S_2 = \text{ReLU}(\theta x_2 + \theta_2 S_1)$

$$\theta_1 x_1 = \begin{bmatrix} -2 \\ 4 \end{bmatrix} \xrightarrow{\text{ReLU}} \begin{bmatrix} 0 \\ 4 \end{bmatrix} = S_1$$

$$y_1^* = \theta_3 S_1 = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 4 \end{bmatrix} = 4$$

$$\theta_1 x_2 + \theta_2 S_1 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}(-3) + (-2)\begin{bmatrix} 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ -14 \end{bmatrix} \xrightarrow{\text{ReLU}} \begin{bmatrix} 3 \\ 0 \end{bmatrix} = S_2$$

$$y_2^* = \theta_3 S_2 = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix} = 3$$

Backprop ← In Rec6 notes

$$L = (y_2^* - y_2)^2 + (y_1^* - y_1)^2$$

$$\frac{\delta L}{\delta \theta_2} = \frac{\delta L}{\delta y_1^*} \cancel{\frac{\delta y_1^*}{\delta \theta_2}} \to 0 + \frac{\delta L}{\delta y_2^*} \frac{\delta y_2^*}{2\theta_2}$$

with respect to both summed because chain rule

$$\frac{\delta y_1^*}{\delta \theta_2} = \underbrace{\frac{\delta y_1^*}{\delta S_1}}_{\theta_3} \underbrace{\frac{\delta S_1}{\delta \theta_2}}_{0} = 0$$

$$\frac{\delta y_2^*}{\delta \theta_2} = \underbrace{\frac{\delta y_2^*}{\delta S_2}}_{\theta_3} \underbrace{\frac{\delta S_2}{\delta \theta_2}}_{\frac{\delta[\tanh(\theta_1 x_2 + \theta_2 S_1)]}{\delta \theta_2}} = \theta_3 S_1 (1 - \tanh^2(\theta_1 x_2 + \theta_2 S_1)$$

→ changed problem to tanh earlier – didn't have time to copy :(

$$\frac{\delta[\tanh(\theta_1 x_2 + \theta_2 S_1)]}{\delta \theta_2} = 1 - \tanh^2(\theta_1 x_2 + \theta_2 S_1))S_1$$

Gates

- Solve vanishing gradients problem

LSTM - composed of three gates TODO: lookup

Linear Regression

→ $X \to \mathbb{R}$

i.e 

$f(x; \theta_1 \theta_0) = \theta \cdot x + \theta_0$ Structural error: Not possible to model

GOAL: Low generalization error

Empirical Risk: $R_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(y^{(i)} - \theta \cdot x^{(i)})$

$$\text{Loss}(z) = \frac{z^2}{2}$$

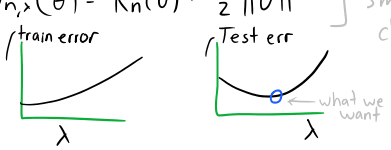
Minimizing empirical risk

Just minimizes training error ⇒ add regularization term ⇒ Ridge Reg

Ridge Regression

$$J_{n,\lambda}(\theta) = R_n(\theta) + \frac{\lambda}{2} \|\theta\|^2$$

- smooths out model so small changes in input lead to small changes in output



Solving linear regression

→ propably not covered

Solving linear regression

- Stochastic Gradient descent or closed form

propably not covered

SGD

$$\theta^{(t+1)} \longleftarrow \theta^{(t)} - \eta \nabla_\theta (J)$$

$$\theta^{(t+1)} \longleftarrow (1-\eta\lambda)\theta^{(t)} + \eta(y^{(i)} - \theta^{(t)} \cdot x^{(i)})\, x^{(i)}$$

Kernels

- Use linear algorithms with non-linear data

$x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^p \quad p \gg d$

↳ Map features to higher dimensions to make data linearly separable

Key properties

1) Inner products
2) θ is weighted sum of $x^{(i)}$'s

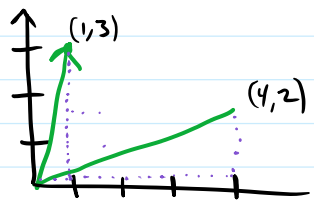
6.036 Assignment 0 Preliminaries

Wednesday, February 8, 2017 6:27 PM

Linear Algebra

1. Points and Vectors

a)



b)

$$\cos\theta = \frac{u \cdot v}{\|u\|\|v\|} \qquad \|u\| = \sqrt{u_1^2 + u_2^2}$$

$$u = [.4, .3] \qquad \|u\| = \sqrt{(.4)^2 + (.3)^2} = .5$$

$$v = [-.15, .2] \qquad v = \sqrt{(-.15)^2 + (.2)^2} = .25$$

$$\cos\theta = \frac{u \cdot v}{(.5)(.25)} = \frac{u_1 v_1 + u_2 v_2}{.125} = \frac{-.06 + .06}{.125} = 0$$

$$\theta = \arccos(0) = 90$$

normalized

$$\hat{u} = \frac{u}{\|u\|} = (.4/.5, .3/.5)$$

$$\hat{v} = \frac{v}{\|v\|} = (-.15/.25, .2/.25)$$

c) $x_1 = [a_1, a_2, a_3] \qquad x_2 = [a_1, -a_2, a_3]$

$$\cos\theta = \frac{x_1 \cdot x_2}{\|x_1\| \cdot \|x_2\|} = \frac{a_1^2 - a_2^2 + a_3^2}{\sqrt{a_1^2 + a_2^2 + a_3^2} \cdot \sqrt{a_1^2 + a_2^2 + a_3^2}}$$

$$= \frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}$$

$$\Rightarrow \theta = \arccos\left(\frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}\right)$$

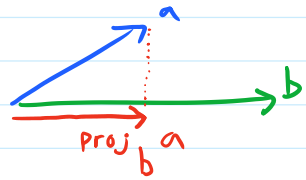
Orthogonal when $x_1 \cdot x_2 = 0$

or

$$\theta = \arccos\left(\frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}\right) = 90 \Rightarrow \frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2} = 0$$

d) $\text{Proj}_b a = \frac{a \cdot b}{\|b\|} \cdot b = \frac{a \cdot b}{\|b\|^2} \cdot b$

d) $\text{Proj}_b a = \frac{a \cdot b}{b \cdot b} \cdot b = \frac{a \cdot b}{\|b\|^2} \cdot b$



$$\text{Proj}_{x_2} x_1 = \frac{x_1 \cdot x_2}{\|x_2\|^2} \cdot x_2$$

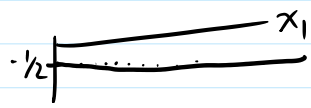
$$= \frac{a_1^2 - a_2^2 + a_3^2}{\sqrt{a_1^2 + a_2^2 + a_3^2}} \cdot [a_1, -a_2, a_3]$$

2. Planes

a) $1 + 2x_1 = 0 \Rightarrow x_1 = -1/2$

$\theta' = [2]$

$d = 1$



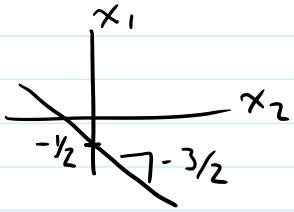
$\frac{-\theta_1}{\theta_2} = -1/2$

↳ infinite choices

① $1 + 2x_1 + 3x_2 = 0$

$\theta' = [2, 3]$

$x_1 = -\frac{1}{2} - \frac{3}{2}x_2$



② $\frac{1}{2} + x_1 + \frac{3}{2}x_2 = 0 \Rightarrow x_1 = -1/2 - 3/2\, x_2$

$\Rightarrow \theta' = \left[\frac{\theta_1}{2}, \frac{\theta_2}{2} \ldots \frac{\theta_d}{2}\right] \quad \theta_0' = \frac{\theta_0}{2}$

b) Find vector $\vec{n}$ that is normal to the plane P_1

if $\vec{d} \times \vec{n} = 0 \Rightarrow$ vector $\vec{d}$ is orthogonal to plane

c) ⋮ At end

3. Matrices

a) $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 1 \end{bmatrix}$ i) $A^T = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 5 & 2 \\ 3 & 6 & 1 \end{bmatrix}$

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} 3 & 6 & 1 \end{bmatrix}$$

$$\det(A) = |A| = 1 \cdot \begin{vmatrix} 5 & 6 \\ 2 & 1 \end{vmatrix} - 2 \begin{vmatrix} 4 & 6 \\ 1 & 1 \end{vmatrix} + 3 \begin{vmatrix} 4 & 5 \\ 1 & 2 \end{vmatrix}$$

$$= 1 \cdot [5(1) - 6(2)] - 2 \cdot [4(1) - 6(1)] + 3[8-5]$$

$$= -7 - (-4) + 9 = 6$$

$$\det(A) = \det(A^T) = 6$$

b)

```python
import numpy as np

a = np.matrix('1 2 3; 4 5 6; 1 2 1')

print "Transpose: \n" , a.transpose()
print "Determinant: ", np.linalg.det(a), np.linalg.det(a.transpose())
print "Inverse \n",  np.linalg.inv(a)

#Verify that AA^-1 = I (Identitity Matrix)
print "AA ^ -1 \n", a * np.linalg.inv(a)
```

c) $g = [2, 1, 3]$

$$gA = \underset{1\times 3}{[2,1,3]} \underset{3\times 3}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 1 & 2 & 1 \end{bmatrix}} = \underset{1\times 3}{[\quad\quad]}$$

$$= [2(1) + 1(4) + 3(1) \quad (2)(2) + (1)(5) + 3(2) \quad 15]$$

$$= [9 \quad 15 \quad 15]$$

d)

$$C = \underset{3\times 2}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} \qquad b = \underset{3\times 1}{\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix}}$$

i) $$CC^T = \underset{3\times 2}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} \cdot \underset{2\times 3}{\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}} = \underset{3\times 3}{\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}}$$

ii) $$C^TC = \underset{2\times 3}{\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}} \underset{3\times 2}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} = \underset{2\times 2}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}}$$

iii) $C^Tb = 2\times 3 \quad 3\times 1 = 2\times 1$

iv) $b^TC = 1\times 3 \quad 3\times 2 = 1\times 2$

e) $(AB)^{-1} = B^{-1}A^{-1}$

$$(AB)^T = B^T A^T$$

f) $A^T(AB - C) = 0$

$$A^T A B - A^T C = 0$$

$$(A^T A) B = A^T C$$

$$B = (A^T A)^{-1} A^T C = A^{-1} A^{T^{-1}} A^T C = A^{-1} C$$

g) $B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 4 \\ 5 & 6 & 4 \end{bmatrix}$

$$\begin{array}{c} r_1 \\ r_2 \\ r_3 \end{array} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 4 \\ 5 & 6 & 4 \end{bmatrix} R_3 \to r_3 - 2r_1 - r_2 \Rightarrow \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 4 \\ 0 & 0 & 0 \end{bmatrix}$$

two non-zero rows $\Rightarrow$ rank$(B) = 2$

4. Probability

a) PDF: $p_X(x)$

not necessarily $\boxed{\text{FALSE}}$

b) $\boxed{\text{TRUE}}$ by definition

c) $\boxed{\text{TRUE}}$

d) $\boxed{\text{FALSE}}$ $\int_{-\infty}^{\infty} p_X(x)\,dx = 1$

e) $\boxed{\text{TRUE}}$ (beat you to it :P)

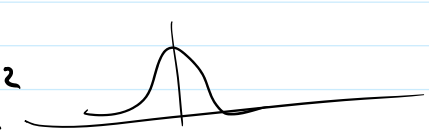
5. Univariate Gaussians

a) $X \sim N(1,2)$

$$P(X \in [.5, 2]) = \int_{.5}^{2} p_X(x)\,dx = \frac{1}{\sqrt{2 \cdot 2\pi}} \cdot e^{-\frac{(x-.5)^2}{2 \cdot 2}}$$

b) PDF of normal

$$= \frac{1}{\sqrt{2\sigma^2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



maximized at mean: $x = \mu$

max value $= \frac{1}{\sqrt{2\sigma^2\pi}}$

maximized at mean : $x = \mu$

max value $= \frac{1}{\sqrt{2\sigma^2 \pi}}$

c) Joint PDF of independent Normals

$$\text{PDF} = \prod_{i=1}^{n} N(\mu, \sigma^2) = \prod_{i=0}^{n} \frac{1}{\sqrt{2\sigma\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

$$= (2\sigma^2 \pi)^{-n/2} e^{\frac{-\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}}$$

6. Optimization, Gradients

a) $\frac{\partial}{\partial \theta_j} L(x, \theta)$ $\quad L(x,\theta) = \log\left(1 + e^{-\theta_1 x_1 - \theta_2 x_2}\right)$

$$\nabla_\theta L(x,\theta) = \left(\frac{\delta}{\delta\theta_1}, \frac{\delta}{\delta\theta_2}\right) = \left(\frac{-x_1 e^{-(\theta \cdot x)}}{1+e^{-(\theta \cdot x)}}, \frac{-x_2 e^{-(\theta \cdot x)}}{1+e^{-(\theta \cdot x)}}\right)$$

b) Direction:

Points in an increasing direction

$L(x,\theta)$ will be larger if we move in the direction of the gradient.

Planes continued // TODO move to planes section

c)

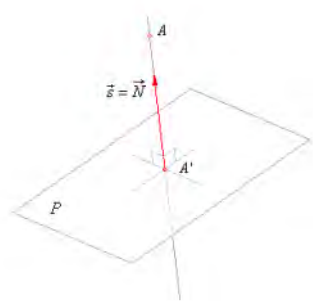
Projection of a point onto a plane

A given point $A(x_0, y_0, z_0)$ and its projection A' determine a line of which the direction vector s coincides with the normal vector N of the projection plane P.

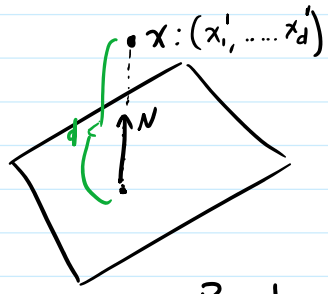
As the point A' lies at the same time on the line AA' and the plane P, the coordinates of the radius (position) vector of a variable point of the line written in the parametric form

$$x = x_0 + a \cdot t,\ y = y_0 + b \cdot t \text{ and } z = z_0 + c \cdot t,$$

These variable coordinates of a point of the line plugged into the equation of the plane will determine the value of the parameter t such that this point will be, at the same time, on the line and the plane.



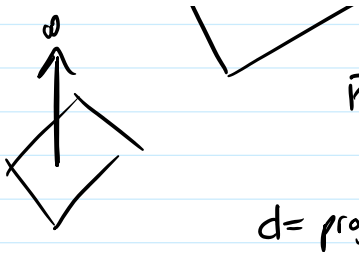
d)



Normal vector : θ

Vector from plane to point : $w = \begin{bmatrix} x_1 - x_1' \\ \vdots \\ x_n - x_1' \end{bmatrix}$

Projecting w onto normal θ gives d

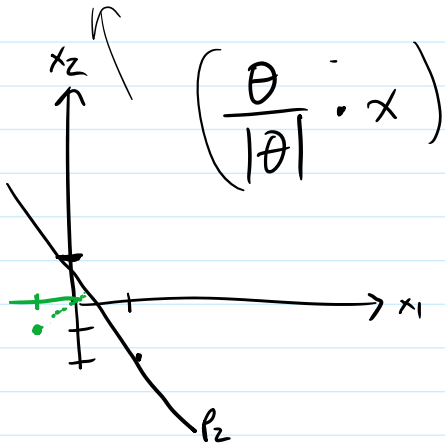


Projecting w onto normal θ gives d

$$d = \text{proj}_\theta w = \frac{\theta_0 + \theta_1 x_1' + \theta_2 x_2' + \dots \theta_n x_n'}{\sqrt{\theta_1^2 + \theta_2^2 + \dots + \theta_n^2}}$$

$$\left(\frac{\theta}{|\theta|} \cdot x\right) + \theta_0$$

e)



$x_2 = 1 - 3x_1$

Line defined by an equation [edit]

In the case of a line in the plane given by the equation $ax + by + c = 0$, where a, b and c are real constants with a and b not both zero, the distance from the line to a point (x_0, y_0) is[1]

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}.$$

The point on this line which is closest to (x_0, y_0) has coordinates:[2]

$$x = \frac{b(bx_0 - ay_0) - ac}{a^2 + b^2} \text{ and } y = \frac{a(-bx_0 + ay_0) - bc}{a^2 + b^2}.$$

i) $a = 3$
$b = 1$
$c = -1$

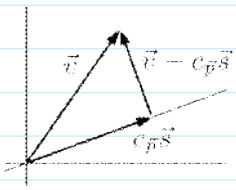
$$d = \frac{|3x_0 + 1y_0 + -1|}{\sqrt{3^2 + 1^2}}$$

$$d = \frac{|-3 - 1 - 1|}{\sqrt{10}} = \frac{5}{\sqrt{10}}$$

ii)

$$d = \frac{|0x_0 + 0y_0 - 1|}{\sqrt{10}} = 1/\sqrt{10}$$

iii)

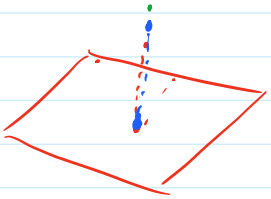


Definition 1.1

The **orthogonal projection of $\vec{v}$ onto the line** spanned by a nonzero $\vec{s}$ **is this vector.**

$$\text{proj}_{[\vec{s}]}(\vec{v}) = \frac{\vec{v} \cdot \vec{s}}{\vec{s} \cdot \vec{s}} \vec{s}$$

f)



There is an infinite # of points to project onto a finite plane

Any point across normal from any point

Any point across normal from any point to plane will project to the same point.

Assignment 0 submit

Friday, February 10, 2017 8:43 AM

c) $x_1 = [a_1, a_2, a_3]$ $x_2 = [a_1, -a_2, a_3]$

$$\cos\theta = \frac{x_1 \cdot x_2}{\|x_1\| \cdot \|x_2\|} = \frac{a_1^2 - a_2^2 + a_3^2}{\sqrt{a_1^2 + a_2^2 + a_3^2} \cdot \sqrt{a_1^2 + a_2^2 + a_3^2}}$$

$$= \frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}$$

$$\Rightarrow \theta = \arccos\left(\frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}\right)$$

Orthogonal when $x_1 \cdot x_2 = 0$

or

$$\theta = \arccos\left(\frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2}\right) = 90$$

$$\Rightarrow \frac{a_1^2 - a_2^2 + a_3^2}{a_1^2 + a_2^2 + a_3^2} = 0$$

d) normal $= \vec{n} = [\theta_1, \theta_2 \ldots]$

$$\hat{n} = \frac{\vec{n}}{|n|}$$

$$D = \theta_0 - d = \theta_0 - |\vec{x}| \cos\phi = \theta_0 - \hat{n}\, x$$

$$\underbrace{D = \hat{n} \cdot x + \theta_0}_{\text{Hessian normal form}}$$

f) $A^T(AB-C)=0$

$$A^TAB - A^TC = 0$$
$$(A^TA)B = A^TC$$
$$B = (A^TA)^{-1}A^TC = A^{-1}A^{T^{-1}}A^TC = A^{-1}C$$

4. Probability

a) PDF: $p_X(x)$

not necessarily $\boxed{\text{FALSE}}$

b) $\boxed{\text{TRUE}}$ by definition

c) $\boxed{\text{TRUE}}$

d) $\boxed{\text{FALSE}}$ $\int_{-\infty}^{\infty} p_X(x)\,dx = 1$

(beat you to it :P)

e) $\boxed{\text{TRUE}}$

b) PDF of normal

$$= \frac{1}{\sqrt{2\sigma^2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

maximized at mean: $x=\mu$

max value $= \frac{1}{\sqrt{2\sigma^2\pi}}$

c) Joint PDF of independent Normals

$$\text{PDF} = \prod_{i=1}^{n} N(\mu,\sigma^2) = \prod_{i=0}^{n} \frac{1}{\sqrt{2\sigma\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$
$$= (2\sigma^2\pi)^{-n/2} e^{\frac{-\sum_{i=1}^{n}(x_i-\mu)^2}{2\sigma^2}}$$

6. Optimization, Gradients

a) $\frac{\partial}{\partial \theta_j} L(x, \theta)$ $\quad L(x,\theta) = \log\left(1 + e^{-\theta_1 x_1 - \theta_2 x_2}\right)$

$$\nabla_\theta L(x,\theta) = \left(\frac{\delta}{\delta\theta_1}, \frac{\delta}{\delta\theta_2}\right) = \left(\frac{-x_1 e^{-(\theta \cdot x)}}{1+e^{-(\theta \cdot x)}}, \frac{-x_2 e^{-(\theta \cdot x)}}{1+e^{-(\theta \cdot x)}}\right)$$

b) Direction:

Points in an increasing direction

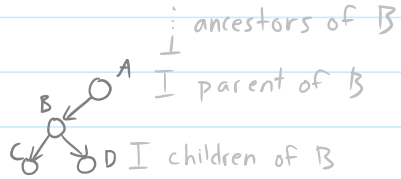
$L(x,\theta)$ will be larger if we move in the direction of the gradient.

R: Bayesian Networks

Friday, April 28, 2017 4:07 PM

Bayesian Network

1. Directed Acyclic Graph (DAG)
2. Encodes a probability distribution
   - Simplifies calculation of joint probability



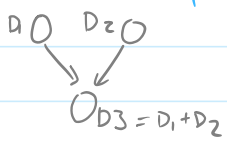
$$P(A,B,C,D) = P(A)\,P(B|A)\,P(C|B)\,P(D|B)$$

Independance $P(A \cap B) = P(A)P(B)$

Marginal Independence - don't know anything about other variables

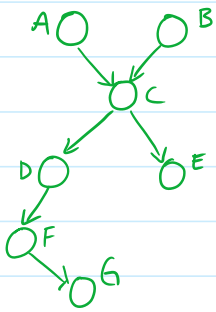
Conditional Independence

Induced dependance



D_1 and D_2 independent

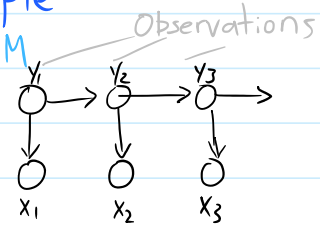
D_1 and D_2 dependent | D_3 ← given



- A,B marginally independent
- D,E NOT marginally independent → both depend on C
- D,E conditionally independent given C — knowing D could tell you about C which could tell you about E
- A,B NOT cond. indep. given C
- A,B NOT cond. indep. given G
- D,E NOT cond. indep. given A,B ← knowing A,B doesnt tell you everything about C

Example

HMM



$$P(y_1)\,P(x_1|y_1)\ \ldots$$

$$\prod_{i=2}^{n} p(y_i|y_{i-1})\,p(x_i|y_i)$$

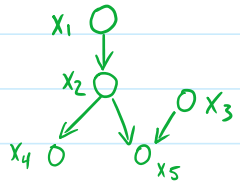
- y_3 does not depend on y_1 ⇒ last state captures all of the information

Viterbi Algorithm

6.036 Final Review

Friday, May 19, 2017 11:10 AM

Bayesian Nets p 109-113



$$p(x_1)\, p(x_2|x_1)\, p(x_3)\, p(x_1|x_2)\, p(x_5|x_3, x_4)$$

joint distribution over all the variables

Independence: $p(A,B) = P(A)P(B)$

Marginallization: summing over all possible values

Formulas:

① $\sum_a p(b|a)\, p(a) = p(b)$

② $\sum_b p(a|b)\, p(b|c) = p(a|c)$

← marginalize over b

Example

Q: Are x_3, x_4 marginally independent?

① Marginalize over every variables not considered

$$\sum_{x_1,x_2,x_5} p(x_1)\, p(x_2|x_1)\, p(x_3)\, p(x_4|x_2)\, p(x_5|x_2\, x_3)$$

x_5 mentioned once, ⇒ move to the right

$$= p(x_3) \sum_{x_1} \Big[p(x_1) \underbrace{\sum_{x_2} \Big[p(x_2|x_1)\, p(x_4|x_2)}_{p(x_4|x_1)} \cancel{\sum_{x_3} p(x_5|x_2 x_3)} \Big] \Big]$$

$$= p(x_3) \sum_{x_1} p(x_1)\, p(x_4|x_1) = p(x_3)\, p(x_4)$$

⇒ x_3, x_4 independent

Q Are x_2, x_5 independent

$$\sum_{x_1,x_3,x_4} p(x_1)\, p(x_2|x_1)\, p(x_3)\, p(x_4|x_2)\, p(x_5|x_2 x_3)$$

$$= \sum_{x_1 x_3} p(x_1)\, p(x_2|x_1)\, p(x_5|x_2, x_3) \cancel{\sum_{x_4} p(x_4|x_2)}$$

$$= \sum_{x_3} p(x_3)\, p(x_5|x_2, x_3) \sum_{x_1} p(x_1)\, p(x_2|x_1)$$

$$= \sum_{x_3} p(x_3)\, p(x_5|x_2, x_3)\, p(x_2)$$

$$= p(x_2) \sum_{x_3} p(x_3)\, p(x_5|x_2, x_3)$$

$$= p(x_2)\, p(x_5|x_2)$$

$$= P(x_2)\sum_{x_3} p(x_3)\,p(x_5 \mid x_2, x_3)$$

$$= p(x_2)\,p(x_5 \mid x_2)$$

$\Rightarrow$ NOT independent

Q $X_3 X_4 \mid X_2$

$$\sum_{x_1 x_2 x_5} p(x_1 \mid x_2)\,p(x_2 \mid x_1)\,p(x_3 \mid x_2)\,p(x_4 \mid x_2)\,p(x_5 \mid x_2\, x_3)$$

$$= \sum_{x_1 x_2} p(x_1)\,p(x_2 \mid x_1)\,p(x_3)\,p(x_4 \mid x_2) \cancelto{1}{\sum_{x_5} p(x_5 \mid x_2 x_3)}$$

$$= p(x_3)\sum_{x_2} p(x_4 \mid x_2) \sum_{x_1} p(x_1)\,p(x_2 \mid x_1)$$

$$= p(x_3)\underbrace{\sum_{x_2} p(x_4 \mid x_2)\,p(x_2)}_{\text{over just one term since } x_2 \text{ given}} = p(x_3)\,p(x_4)$$

$\Rightarrow$ independent

★ Q $X_3 X_4 \mid X_5$ summing over $x_5 \Rightarrow$ summing over one term

$$\sum_{x_1 x_2} p(x_1 \mid x_5)\,p(x_2 \mid x_1, x_5)\,p(x_3 \mid x_5)\,p(x_4 \mid x_2, x_5)\,p(x_5 \mid x_2, x_3)$$

$$= \sum_{x_2} p(x_3 \mid x_5)\,p(x_4 \mid x_2, x_5)\,p(x_5 \mid x_2\, x_3, x_5)\,p(x_2 \mid x_5)$$

$$= p(x_3) \sum_{x_2} p(x_2)\,p(x_4 \mid x_2)\,p(x_5 \mid x_2 x_3)$$

$$= p(x_3)\,p(x_4)\,p(x_5 \mid x_3)$$

$\Rightarrow$ NOT independent

Example: X can take: a, b or c

Free parameters: # options − 1 if no parents $\qquad FP(x_1) = 3 - 1 = 2$

options * Free params of parent $\qquad FP(x_2 \mid x_1) = 3 * 2$

X_5: $18 = 3 \times 3 \times 2$

Hidden Markov Models

2015 P4

Hidden: $Y_1 \rightarrow Y_2 \rightarrow Y_3 \rightarrow \ldots$

Observations: $X_1 \quad X_2 \quad X_3$

Y_t: location of robot on day t

X_t: temperature transmitted on day t

Y_1: ?

Initial State

A	.5
B	.5
C	0
D	0
E	0
F	0

~~Transmission~~ Emission?

Probability $p(x|y)$

Y_t \ X_t	hot	cold
A	.9	.1
B	.1	.9
C	.1	.9
D	.9	.1
E	.1	.9
F	.1	.9

Transmission Probabilities

Y_{t-1} \ Y_t	A	B	C	D	E	F
A		1				
B			1			
C			.5	.5		
D				.5	.5	
E					.5	.5
F						1

?: If transmitted X_1 = hot X_2 = cold X_3 = cold, where can robot be?

$Y_1 \rightarrow Y_2 \rightarrow Y_3$

$\downarrow \quad \downarrow \quad \downarrow$

hot cold cold

Can start A or B:

A→B→C

B→C→C

B→C→D

"Joint distribution of HMM"

markov

$$P(Y_1, Y_2, Y_3, X_1, X_2, X_3) = P(X_1)\, P(Y_2 \mid Y_1)\, P(Y_3 \mid Y_2) \cdot P(X_1 \mid Y_1)\, P(X_2 \mid Y_2)\, P(X_3 \mid Y_3)$$

$\underline{\underline{A \to B \to C}}$ $\Rightarrow$ most likely path given h, c, c

$P(A,B,C,h,c,c) = 1/2 \cdot 1 \cdot 1 \cdot .9 \cdot .9 \cdot .9 =$

$P(A,B,C,C,h,c,c) = 1/2 \cdot 1 \cdot 1/2 \cdot .1 \cdot .9 \cdot .9 =$

$P(B,C,D,h,c,c) = 1/2 \cdot 1 \cdot 1/2 \cdot .1 \cdot .9 \cdot .1 =$

For HMM and Final: No Viterbi and no EM

? $\underline{P(Y_3 \mid h,c,c)}$

$P(Y_3 = D \mid h,c,c) = \dfrac{1/2 \cdot 1 \cdot 1/2 \cdot .1 \cdot .9 \cdot .1}{\text{sum}}$

$P(Y_3 = C \mid h,c,c) = \dfrac{1/2 \cdot 1 \cdot .9 \cdot .9 \cdot .9 + 1/2 \cdot 1 \cdot 1/2 \cdot .10 \cdot .9 \cdot .9}{\text{sum}}$

$P(X_1, X_2, X_3 = h,c,c)$

$\hookrightarrow =$

sum $= P(A,B,C,h,c,c) + P(B,C,C,h,c,c) + P(B,C,D,h,c,c)$

All ways temps can come up

Generalization, Model Selection, Clustering

Model Selection: BIC, AIC, Cross-validation

<u>VC-dimension</u>: A set of classifiers have VC-dimenion $\geq d$

if $\exists$ d points which can arbitrarily classified (for any ~~change~~ choice of possible 2^d labelings $\exists$ h classifying them correctly)

<u>K-Means</u>

Partition points $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ to clusters $C_1, C_2, \ldots C_k$ with centers $z_1, z_2 \ldots z_k$

to minimize cost $= \sum_{i=1}^{k} \sum_{j \in C_i} d(x^{(i)}, z^{(i)})$

Algorithm

Start w initial centers $z_1 \dots z_k$

1: assign each x^j to z_i which $\arg\max \| z^{(i)} - x^{(i)} \|^2 \to C^i$

2: Update $z^{(i)} = \frac{1}{m_i} \sum_{j=C_i} x^j$

Repeat 1,2 until covergence

K-Medoid : variation of K-means

Generative Models, EM

↳ care about underlying structure

- Multinomial / Categorical
- Gaussians
- ~~Mixer~~ Mixture Model

Want to find a model θ that best fits the data

$$\theta = \arg\max_{\theta} P(D|\theta)$$

EM

1. E step - posterior probabilities $p(j|i)$

   M step - re-estimate θ

Reinforcement Learning - decide what actions to take in what state

Markov Decision Process (MDP)

S ← state $\qquad R(s, a, s')$ ← new state ← Reward function

A ← actions $\qquad T(s, a, s') = P(s' \mid s, a)$ ← Transition function

- Need to assign a value to each state

$$\text{value} \overset{?}{=} \sum_{t=0}^{\infty} R_t$$ ← can be too large, use discount factor $\gamma \in [0, 1]$

immediate reward ↘ ↑ longer term

$$= \sum_{t=0}^{\infty} \gamma^t R_t = R_0 + \gamma R_1 + \gamma^2 R_2 + \dots$$

$= V_0$ ← value at time 0

$$V_1 = R_1 + \gamma R_2 + \gamma^2 R_3 \dots \Rightarrow V_0 = R_0 + \gamma V_1$$

$\pi^*(s) = a$ ← optimal policy – which actions to chose

$V^*(s)$ ← optimal value function

Value at s if we are performing optimal

$Q^*(s,a)$ ← Value at s executing a

$$V^*(s) = \max_{a \in A} Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V(s')\right]$$

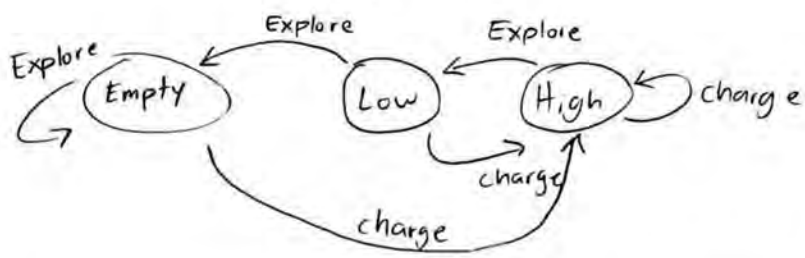
immediate reward

could end up in any s'

$$\Rightarrow V^*(s) = \max_{a \in A} \sum_{s'} T(s,a,s')\left[R(s,a,s' + \gamma V_i\right.$$

Example

P5 Empty, low, high



? Optimal policy if $\gamma = 0$ ← only immediate reward

$\pi_0^*(HIGH) = $ Explore $\quad \pi_0^*(LOW) = $ EXPLORE $\quad \pi_0^*(EMPTY) = $ Explore OR charge

$\gamma = 1/2$ ← equivalent to $\gamma = 0$ since only one move left

$\gamma = 0$ S	$\pi^*(s)$	$\gamma \neq 0$	$V_0^*(s)$	$V_1^*(s)$	$V_2^*(s)$
E	Either	Charge	0	-10	Explore: $-10 + \frac{1}{2}V_1(E)$ Charge: $-10 + \frac{1}{2}V_1(H)$ ✓ $= -9$
L	Explore	Charge	0	+2 ← Explore	Explore: $2 + \frac{1}{2}V_1(E)$ Charge: $-1 + \frac{1}{2}V_1(H)$ ✓ $= 0$
H	Explore	Explore	0	+2	Explore: $2 + \frac{1}{2}V_1(L)$ ✓ Charge: $0 + \frac{1}{2}V_1(H)$

$V_i^*(s)$ ← value at state s w/ optimal policy w/ i steps to go

$V_0^*(s) = 0$ for all s

$$V_{t+1}^*(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V_t^*(s')\right]$$

2017

6.036 Official Cheat Sheet

- A (hyper-)plane is a set of points $x \in \mathcal{R}^d$ such that $\theta \cdot x + \theta_0 = 0$. Vector θ is normal to the plane. The signed distance of any point x from the plane is $(\theta \cdot x + \theta_0)/\|\theta\|$. The value of distance is positive on the side where θ points to, and negative on the other side.
- A linear classifier with offset: $h(x;\theta) = \text{sign}(\theta \cdot x + \theta_0)$
- Training error (classification error): $\epsilon_n(h) = \frac{1}{n}\sum_{i=1}^{n}[[y^{(i)} \neq h(x^{(i)})]]$
- Loss functions:

$$z = y(\theta \cdot x + \theta_0) \text{ (agreement)}$$

$$\text{Loss}_{0,1}(z) = [[z \leq 0]]$$

$$\text{Loss}_h(z) = \max\{1 - z, 0\}$$

- SVM: Finds a large margin classifier by minimizing

$$\frac{1}{n}\sum_i \text{Loss}_h(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)) + \frac{\lambda}{2}||\theta||^2$$

which can be done using stochastic gradient descent (Pegasos). prediction $\theta x^{(n)} > 0$

- Linear regression: finds the parameters of a linear predictor $\theta \cdot x + \theta_0$ by minimizing

$$\frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta \cdot x^{(i)} - \theta_0)^2/2$$

- Low-rank matrix factorization for collaborative filtering: Minimize

$$J(U,V) = \sum_{(a,i)\in D}(Y_{ai} - [UV^T]_{ai})^2/2 + \frac{\lambda}{2}\sum_{a=1}^{n}\sum_{j=1}^{k}U_{aj}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{j=1}^{k}V_{ij}^2$$

Can be solved iteratively by fixing one matrix and using linear regression to find the other.

- Kernels: $K(x,x') = \phi(x) \cdot \phi(x')$

Kernel	form
Linear	$x \cdot x'$
Quadratic	$x \cdot x' + (x \cdot x')^2$
Radial basis	$\exp(-\|\|x - x'\|\|^2/2)$

- Kernel Perceptron (with offset): Cycles through each point t=1,..n and checks if $y^{(t)}(\sum_{i=1}^{n}\alpha_i y^{(i)}[K(x^{(i)}, x^{(t)}) + 1]) \leq 0$. If true, $\alpha_t = \alpha_t + 1$. # of mistakes
- Neural Nets:
  - unit i in layer l evaluates its aggregate input based on the previous layer as $z_i^l = \sum_{j=1}^{m} f(z_j^{l-1})w_{ji}^l + w_{0i}^l$ and its activation as $f(z_i^l)$
  - common activation functions include ReLU ($f(z) = \max\{0, z\}$), tanh, and the identity function
  - backpropagation: $\delta_j^{l-1} = f'(z_j^{l-1})\sum_i w_{ji}^l \delta_i^l$
- RNN equations are given if used
- Good luck! ☺

Perception covergence

$\|x^{(i)}\| \leq R$

margin γ $\Rightarrow$ covergence after $(R/\gamma)^2$ mistakes

6.036 Midterm Notes

3/22/17
Fernando Trujano

Perceptron

initialize θ, θ_0 to 0 (vector, scalar)

for t in T: # epochs

  for i in n: # features

    if $y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0$: # mistake made (dot product)

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

      $\theta_0 = \theta_0 + y^{(i)}$

will always find a boundry if linearlly separable

Support Vector Machines

Hinge Loss: $\text{Loss}_h(y(\theta \cdot x + \theta_0)) = \max\{0, 1 - y(\theta \cdot x + \theta_0)\}$

Want to minimize: $\underbrace{\frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h(y^{(i)}(\theta \cdot x^{(i)}))}_{\text{minimize avg hinge loss}} + \underbrace{\frac{\lambda}{2}\|\theta\|^2}_{\text{push margins apart}}$ $\quad d = \frac{1}{\|\theta\|}$

Gradient Descent: $\theta' = \theta - \eta \nabla_\theta f$ (Objective function)

move opposite to gradient; learning rate; Gradient

Pegasos ← gradient descent

initialize $\theta = 0$

  for t in T:

    ~~for i~~ $i = \text{random}(1, n)$

    $\eta = 1/t$

    if $y^{(i)} \theta \cdot x^{(i)} \leq 1$

      $\theta = (1 - \eta\lambda)\theta + \eta y^{(i)} x^{(i)}$

    else

      $\theta = (1 - \eta\lambda)\theta$

Recommender / Collaborative Filtering / low rank Matrix Factorization

$$\underset{\text{Prediction}}{X} = UV^T \qquad U = \begin{bmatrix} - & u^{(1)} & - \\ - & u^{(2)} & - \\ - & & - \end{bmatrix} \qquad V^T = \begin{bmatrix} | & | & | & \\ v^{(1)} & v^{(2)} & \cdots \\ | & | & | & \end{bmatrix}$$

$n \times k$ (rank) $k \times m$

n: #users, m: #movies

Alternating Minimization

1. Initialize movie feature vectors randomly $v^{(1)} \ldots v^{(m)}$
2. Fix v's, solve for each $u^{(a)}$ by minimizing

$$\sum_{i:(a,i)\in D} \left(Y_{ai} - u^{(a)} \cdot v^{(i)}\right)^2 / 2 + \frac{\lambda}{2} \| u^{(a)} \|^2$$

3. Fix u's, solve for each $v^{(i)}$ by minimizing

$$\sum_{a:(a,i)\in D} \left(Y_{ai} - u^{(a)} v^{(i)}\right)^2 / 2 + \frac{\lambda}{2} \| v^{(i)} \|^2$$

Example

Q: $y = \begin{bmatrix} 2 & ? & 0 \\ ? & 2 & 1 \end{bmatrix}$ (2×3) $\lambda = 1$ want to uncover $\overbrace{Y}^{X}$ (find ?)

A:

1. Initialize V randomly

$$V^T = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}$$

2. Fix V's => Find $U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}$

$$UV^T = \underset{2 \times 2}{\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}} \underset{3 \times 2}{\begin{bmatrix} 1 & 0 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}} \neq \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} U_{11} + 0U_{12} & U_{11} + 2U_{12} & 2U_{11} + U_{12} \\ U_{21} + 0U_{22} & U_{21} + 2U_{22} & 2U_{21} + U_{22} \end{bmatrix}$$

$$UV^T = \begin{bmatrix} U_{11} & U_{11}+2U_{12} & 2U_{11}+U_{12} \\ U_{21} & U_{21}+2U_{22} & 2U_{21}+U_{22} \end{bmatrix} \qquad Y = \begin{bmatrix} 2 & ? & 0 \\ ? & 2 & 1 \end{bmatrix}$$

For known values of Y

$$\frac{(-2U_{11}-U_{12})^2}{2}$$

$$J(U,V) = \frac{(2-U_{11})^2}{2} + \frac{(0-(2U_{11}+U_{12}))^2}{2} + \frac{(2-(U_{21}+2U_{22}))^2}{2}$$

From cheat sheet

$$+ \frac{(1-(2U_{21}+U_{22}))^2}{2} + \frac{\lambda}{2}\left(U_{11}^2+U_{12}^2+U_{21}^2+U_{22}^2\right)$$

$$+ \frac{\lambda}{2}\left(1^2+1^2+2^2+0^2+2^2+1^2\right)$$

2.5 Minimize $J(U,V)$ with respect to $U_{11}, U_{12}, U_{21}, U_{22}$

$\underline{U_{11}}$

$$\frac{\delta J}{\delta U_{11}} = \overset{\text{chain rule}}{(-1)}(2-U_{11}) + (-2U_{11}-U_{12})\overset{\text{chain rule!}}{(-2)} + \lambda U_{11} \overset{\text{to minimize}}{=} 0$$

$$= -2 + U_{11} + 4U_{11} + 2U_{12} + \lambda U_{11} = 0$$

$$= 5U_{11} + \overset{=1}{\lambda} U_{11} + 2U_{12} = 2$$

$$6U_{11} + 2U_{12} = 2$$

$\underline{U_{12}}$

$$\frac{\delta J}{\delta U_{12}} = -(-2U_{11}-U_{12}) + \lambda U_{12} = 0$$

$$2U_{11} + U_{12} + U_{12} = 0$$

$$2U_{11} + 2U_{12} = 0$$

Can solve for U_1

$$\begin{array}{rl} & 6U_{11} + 2U_{12} = 2 \\ - & 2U_{11} + 2U_{12} = 0 \end{array} \quad = \quad 4U_{11} = 2 \qquad U_{11} = 2/4 = 1/2$$

$$6(1/2) + 2U_{12} = 2 \qquad 2U_{12} = -1 \qquad U_{12} = -1/2$$

Find U_{21}, U_{22}

$U = \begin{bmatrix} 1/2 & -1/2 \\ 1/5 & 2/10 \end{bmatrix}$ $X = UV^T$ = first prediction keep iterating $\Rightarrow$ 3

Kernels

Classify non linear data using a linear classifier by maping original points to higher dimensions where they <u>are</u> linearly separable. $\phi(x)$ Kernel Function: $K(x, x') = \phi(x) \cdot \phi(x')$

Kernel Perceptron

$\phi(x)$ can be difficult to calculate so we use a trick instead.

on cheatsheet

IF $y^{(t)}\left(\sum_{i=1}^{n} \alpha_i y^{(i)}\left(K(x^{(i)}, x^{(t)}) + 1\right)\right) \leq 0$

$$\alpha_t \leftarrow \alpha_t + 1$$

where α_t is the # of mistakes made and

$$\theta = \sum_{i=1}^{n} \alpha_i y^{(i)} \phi(x^{(i)}) \qquad \theta_0 = \sum_{i=1}^{n} \alpha_i y^{(i)}$$

Kernel Properties

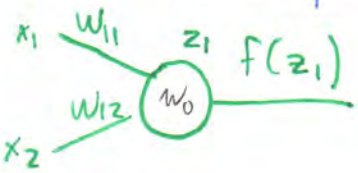
Kernel function valid if $\exists\ \phi(x)$ s.t. $K(x, x') = \phi(x) \cdot \phi(x')$

1. $K(x, x') = 1$ is valid
2. $f: \mathbb{R}^d \to \mathbb{R}$
   $\tilde{K}(x, x') = f(x) K(x, x') f(x')$ is valid
3. $K_1(x, x') + K_2(x, x')$ is valid
4. $K_1(x, x') K_2(x, x')$ is valid

Neural Networks (NN)

Forward Propagation

↳ Tells us what the NN predicts



just remember that each neuron adds all it's weighted inputs + a bias and applies an activation function.

$$z_1 = x_1 W_{11} + x_2 W_{12} + W_0$$

↳ f

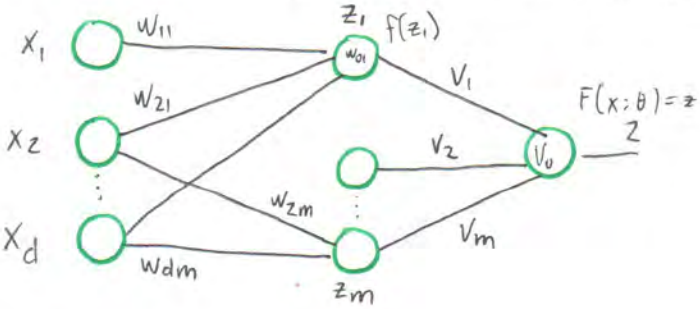
↖ bias

Back propagation

↳ update weights to train the network

- Recall Stochastic gradient descent

⇒ need to find gradient of the loss with respect to each weight



$$z_j = \sum_{i=1}^{d} x_i W_{ij} + W_{0j}$$

$$f(z_j) = \max\{0, z_j\} \quad \text{ReLU}$$

$$z = \sum_{j=1}^{m} f(z_j) V_j + V_0$$

- Notice that changing W_{ij} will affect $z_j \to f(z_j) \to z$

$$\frac{\delta \, Loss(y^{(t)}, z^{(t)})}{\delta W_{ij}} = \left[\frac{\delta z_j^{(t)}}{\delta W_{ij}}\right]\left[\frac{\delta f(z_j^{(t)})}{\delta z_j^{(t)}}\right]\left[\frac{\delta z^{(t)}}{\delta f(z_j^{(t)})}\right]\left[\frac{\delta \, Loss(y^{(t)}, z^{(t)})}{\delta z^{(t)}}\right]$$

↑ chain rule

↳ Changing W_{ij} will change z_j ⇒ changing z_j will change $f(z_j)$ . . . . .

← much easier to calculate than original

$$= [x_i] \begin{bmatrix} 1 \text{ if } z_j > 0 \\ 0 \text{ else} \end{bmatrix} [V_j] \left[\frac{\delta \, Loss}{\delta z^{(t)}}\right]$$

Can use gradient to update weights on error!

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.S064 Introduction to Machine Learning

Midterm exam (March 21, 2013)

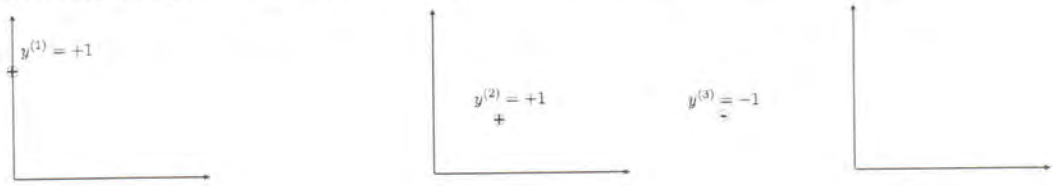
Your name & ID: Try 1

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

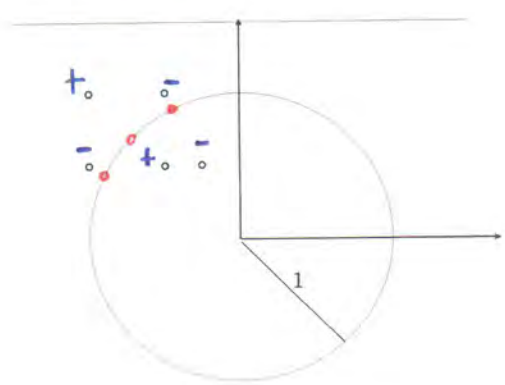
Problem 1.1	Problem 2.1	Problem 2.2	Problem 3.1	Problem 4.1	Total
12	12	10	14	9	57

(1.1) The perceptron algorithm remains surprisingly popular for a 50+ year old (algorithm). Hard to find a simpler algorithm for training linear or non-linear classifiers. Worth knowing this algorithm by heart, yes?

(a) **(6 points)** Let $\theta = 0$ (vector) initially. We run the perceptron algorithm to train $y = \text{sign}(\theta \cdot x)$, where $x \in \mathcal{R}^2$, using the three labeled 2-dimensional points in the figure below. In each figure, approximately draw both θ and the decision boundary *after* updating the parameters (if needed) based on the corresponding point.



(b) **(6 points)** We were wondering what would happen if we normalized all the input examples. In other words, instead of running the algorithm using x, we would run it with feature vectors $\phi(x) = x/\|x\|$. Let's explore this with linear classifiers that include offset, i.e., we use $y = \text{sign}(\theta \cdot x + \theta_0)$ or $y = \text{sign}(\theta \cdot \phi(x) + \theta_0)$ after feature mapping. In the figure below, label *all the points* such that 1) the perceptron algorithm wouldn't converge if they are given as original points, 2) the algorithm would converge if run with $\phi(x) = x/\|x\|$.



6.036 2013 Midterm

a) $\theta = 0$

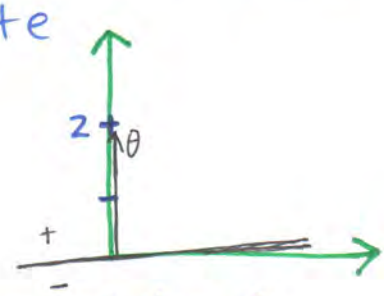
Perceptron: if $y^{(i)}(\theta \cdot x^{(i)}) \leq 0$

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

$\underline{i=1}$

$y^{(1)}(\theta \cdot x^{(1)}) = 1\left(\begin{bmatrix}0\\0\end{bmatrix} \cdot \begin{bmatrix}0\\2\end{bmatrix}\right) = 0 \Rightarrow$ update

$\theta = \begin{bmatrix}0\\0\end{bmatrix} + 1\begin{bmatrix}0\\2\end{bmatrix} = \begin{bmatrix}0\\2\end{bmatrix}$



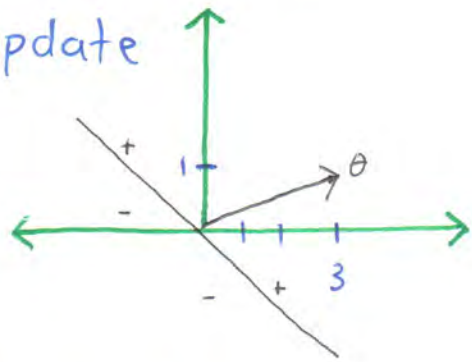
$\underline{i=2}$

$y^{(2)}(\theta \cdot x^{(2)}) = 1\left(\begin{bmatrix}0\\2\end{bmatrix} \cdot \begin{bmatrix}1\\1\end{bmatrix}\right) = 2 \Rightarrow$ no update same

$\underline{i=3}$

$y^{(3)}(\theta \cdot x^{(3)}) = (-1)\left(\begin{bmatrix}0\\2\end{bmatrix} \cdot \begin{bmatrix}-3\\1\end{bmatrix}\right) = -2 \Rightarrow$ update

$\theta = \begin{bmatrix}0\\2\end{bmatrix} + (-1)\begin{bmatrix}-3\\1\end{bmatrix} = \begin{bmatrix}3\\1\end{bmatrix}$



b) Make points not linearlly separable in x but separable in $\phi(x)$

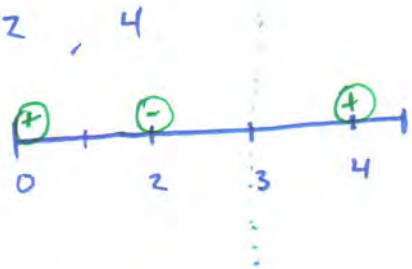
$\Rightarrow$ Map radially to unit circle

All points map to unit circle when normalizing

(2.1)

a) notice that $x \in \mathbb{R}$ and graph is 

$x = 0, 2, 4$

$\Rightarrow$ 

not linearly separable in this space

but in $x, x^2 \ldots$

b) see graph

(2.1) Support vector machines (SVMs) are so popular that we decided to try them out a bit in a simple setting where $x \in \mathcal{R}$. We used both linear and non-linear SVMs:

$$\min w^2/2 \quad \text{subject to} \quad y^{(t)}(wx^{(t)} + w_0) \geq 1, \quad t = 1, \ldots, n \tag{1}$$

$$\min \|\theta\|^2/2 \quad \text{subject to} \quad y^{(t)}(\theta \cdot \phi(x^{(t)}) + \theta_0) \geq 1, \quad t = 1, \ldots, n \tag{2}$$

where $\phi(x)$ is a feature vector constructed from the real valued input x. We wish to compare the resulting classifiers when $\phi(x) = [x, x^2]^T$.

(a) **(3 points)** Provide three input points $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$, where $x^{(i)} \in [0, 4]$, and their associated ± 1 labels such that 1) they cannot be separated with the linear classifier, but 2) are separable by the non-linear classifier with $\phi(x) = [x, x^2]^T$. You may find Figure 2.1 helpful in answering this question.
$(x^{(1)}, y^{(1)}) = (0 , 1)$
$(x^{(2)}, y^{(2)}) = (2 , -1)$
$(x^{(3)}, y^{(3)}) = (4 , 1)$

(b) **(4 points)** Map your three labeled points as labeled feature vectors in Figure 2.1. Approximately draw the resulting *decision boundary* in the feature space of the non-linear SVM classifier with $\phi(x) = [x, x^2]^T$.

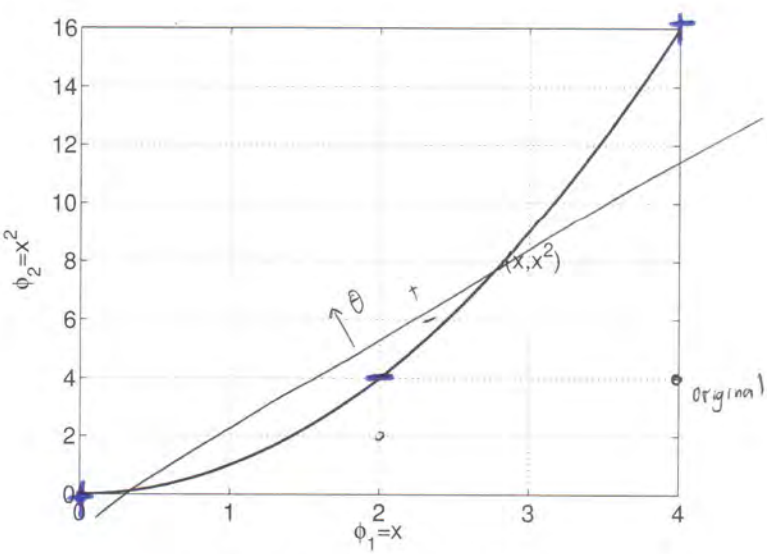


Figure 2.1. Feature space.

(c) **(3 points)** Consider two labeled points $(x = 1, y = 1)$ and $(x = 3, y = -1)$. Is the resulting geometric margin we attain in the feature space using feature vectors $\phi(x) = [x, x^2]^T$

(X) greater, ( ) equal, or ( ) smaller

than the geometric margin resulting from using the input x directly?

(d) **(2 points)** In general, is the geometric margin we would attain using scaled feature vectors $\phi(x) = 2[x, x^2]^T$

(X) greater, ( ) equal, ( ) smaller, or ( ) could be any of these

in comparison to the geometric margin resulting from using $\phi(x) = [x, x^2]^T$?

(2.2) **(10 points)** We were given a dataset with $n = 10$ labeled two-dimensional examples. Having learned about the support vector machine, we wanted to try it out on this data. However, we were unsure which kernel function to use. So we asked students in the class for possible kernel functions, and tried each one of them:

$$
\begin{aligned}
&(1)\ K(x, x') = x \cdot x' \\
&(2)\ K(x, x') = (x \cdot x') + 2(x \cdot x')^2 \\
&(3)\ K(x, x') = 1 - 2(x \cdot x')^2 \\
&(4)\ K(x, x') = (1 + x \cdot x')^5
\end{aligned}
$$

(a) For any kernel $K(x, x') = \phi(x) \cdot \phi(x')$. What can you say about $K(x, x)$ in general?
( ) $K(x, x)$ is never 0, (X) $K(x, x) < 0$ never, ( ) $K(x, x) > 1$ always

$K(x,x) = \phi(x)\phi(x) = (\phi(x))^2 \geq 0$

(b) The optimization routine for the dual SVM problem complained about one of the kernels, and it couldn't be used further. Which one (1-4)? (3)

(c) Only one of the kernels resulted in zero training error. Which one (1-4)? (4)

(d) As far as the training error is concerned, one of the kernels was by far the worst. Which one (1-4)? (1)

(e) After training the three classifiers (one we couldn't use), we obtained additional data to assess their test errors. Which one would you expect to have the lowest test error (1-4)? (2)

(c) Adding second coordinate can only increase margin.

(d) $x = 2$ $\quad \phi_1(x) = [4, 8]$ $\quad \phi(x)_2 = [2, 4]$

2.2) a) Recall $K(x, x') = \phi(x)\phi(x')$

$$\Rightarrow K(x,x) = \phi(x)\phi(x) = (\phi(x))^2 \geq 0$$

b) (3) can be negative if $x, x' = 1$

c) The higher the order polynomial the better the kernel will be on training set

d) (1) is the simples kernel so probably the worst

e) Since $n = 10$ only (9) is "too much" overfitting

3.1)

a) (1) ???

(2) We are minimizing $J(\theta)$ so if $\hat{\theta}$ is optimal $J(\hat{\theta})$ is minimum for all θs

(3)

(4) can just ignore new features by setting their weights to 0 and get the same results.

b) Set the threshold at $1/2$ since $y \in [0,1]$

c) $\uparrow \lambda \Rightarrow \downarrow \|\theta\| \Rightarrow$ predictions $\rightarrow 0$

(3.1) We are faced with a content filtering problem where the idea is to rank new songs by trying to predict how they might be rated by a particular user. Each song x is represented by a feature vector $\phi(x)$ whose coordinates capture specific acoustical properties. The ratings are binary valued $y \in \{0, 1\}$ ("need earplugs" or "more like this"). Given n already rated songs, we decided to use regularized linear regression to predict the binary ratings. The training criterion is

$$J(\theta) = \frac{\lambda}{2}\|\theta\|^2 + \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \theta \cdot \phi(x^{(t)}))^2/2 \tag{3}$$

(a) **(8 points)** Let $\hat{\theta}$ be the optimal setting of the parameters with respect to the above criterion, which of the following conditions must be true (check all that apply)

(1) (X) $\lambda\hat{\theta} - \frac{1}{n}\sum_{t=1}^{n}(y^{(t)} - \hat{\theta} \cdot \phi(x^{(t)}))\phi(x^{(t)}) = 0$

(2) (·) $J(\hat{\theta}) \geq J(\theta)$, for all $\theta \in \mathcal{R}^d$

(3) (X) If we increase λ, the resulting $\|\hat{\theta}\|$ will decrease

(4) (X) If we add features to $\phi(x)$ (whatever they may be), the resulting squared training error will NOT increase

(b) **(3 points)** Once we have the estimated parameters $\hat{\theta}$, we must decide how to predict ratings for new songs. Note that the possible rating values are 0 or 1. When do we choose rating $y = 1$ for a new song x? Please write the corresponding expression.

$y = 1$ if $\hat{\theta} \cdot \phi(x) \geq .5$

(c) **(3 points)** If we change λ, we obtain different $\hat{\theta}$, and therefore different rating predictions according to your rule above. What will happen to your predicted ratings when we increase the regularization parameter λ?

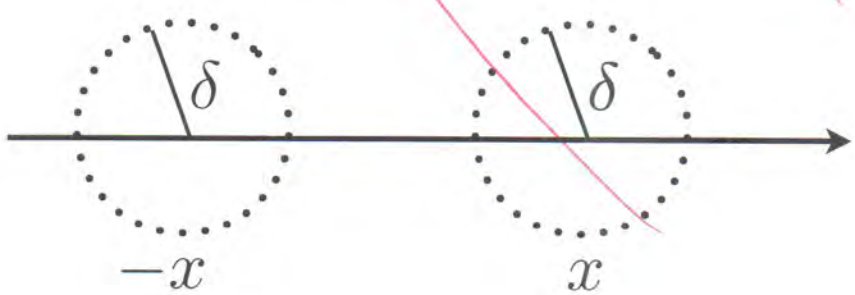
Predictions tend to 0

(4.1) Consider the simple K-means algorithm for clustering. Each iteration of the algorithm consists of two steps, assigning points to the centroids, and updating the centroids based on the points assigned to them. We will assume that $k = 2$.

(a) **(2 points)** If we initialize the centroids to be the means of the two well-separated clusters, will the centroids change after the first iteration? (Y/N) **( )**

(b) **(3 points)** If we initialize the centroids by drawing a random point from each of the two well-separated clusters, how many iterations does it take for the k-means to converge? **( )**

(c) **(4 points)** Consider two spherical (circle-like) clusters of radius δ as shown below. The clusters are centered at locations $-x$ and x. For which values of x would the k-means algorithm fail to find the centers of the two clusters regardless of the initialization?



Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 INTRODUCTION TO MACHINE LEARNING

Midterm exam (March 19, 2015)

Your name & ID: Try 2

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Total

Problem 1 The simple perceptron algorithm for estimating a linear classifier cycles through training examples $(x^{(i)}, y^{(i)})$, $i = 1, \ldots, n$, and performs an update of the parameters in response to each mistake. If we omit the offset parameter, then $\theta \leftarrow \theta + y^{(i)}x^{(i)}$ whenever $(x^{(i)}, y^{(i)})$ is misclassified with the current setting of the parameters.

(1.1) **(4 points) (T/F)** Please mark the statements as **T** or **F**.

(1) (F) Suppose we normalize each training example such that $\|x^{(i)}\| = 1$. If there exists any θ^* such that $y^{(i)}(\theta^* \cdot x^{(i)}) \geq 1$, $i = 1, \ldots, n$, then the perceptron algorithm converges after only a single mistake.

(2) (F) Whenever the perceptron algorithm converges, the parameters of the averaged perceptron algorithm also linearly separate the training examples.

We can turn the linear perceptron into a non-linear classifier by mapping each example to a feature vector $\phi(x)$. We can also rewrite the algorithm such that it uses only inner products between examples (feature vectors). In other words, only values of the kernel function $K(x, x') = \phi(x) \cdot \phi(x')$ are needed. The kernel perceptron algorithm cycles through the training examples as before, updating mistake counts α_i whenever a mistake occurs.

(1.2) **(2 points) (T/F)** (F): The kernel perceptron without offset classifies any new example x according to the sign of $\sum_{j=1}^{n} \alpha_j y^{(j)} K(x^{(j)}, x)$ where $\sum_{j=1}^{n} \alpha_j \leq n$.

Here we use a kernel perceptron to classify nodes in an undirected graph. You can think of the graph as a social network representing friendship relations. Each person is a node in the graph and there's an edge between two nodes whenever people are friends. Our goal is to learn to predict a positive/negative label for each node. We were thinking of using a particular type of kernel function based on how many neighbors any two nodes have in common. In other words, if $N(i)$ is the set of neighbors of node i, then

$$K(i, j) = |N(i) \cap N(j)| \quad (\text{\# of common neighbors}) \tag{1}$$

(1.3) **(4 points)** Which of the following properties of $K(i, j)$ are *also* properties of any valid kernel function over the nodes. Check all that apply.

(1) (X) $K(i, j) = K(j, i)$ for all i, j

(2) ( ) $K(i, j) \geq 0$ for all i, j

1.1)

(1) Perceptron Covergence Theorem

since $\|x^{(i)}\| = 1 \Rightarrow R = 1$

$\#$ mistakes $= \left(R/\gamma\right)^2$ margin could be anything $\Rightarrow$ False

(2) ???

1.2) False total # errors also depends on number of iterations

1.3)

(1) YES $K(x, x') = \phi(x) \cdot \phi(x') = \phi(x')\,\phi(x)$

(2) Not necesarily for all i, j but yes for all $i = j$

(3) (X) $K(i,i) \geq 0$ for all i

(4) ( ) $K(i,j) \leq K(i,i)$ for all i,j

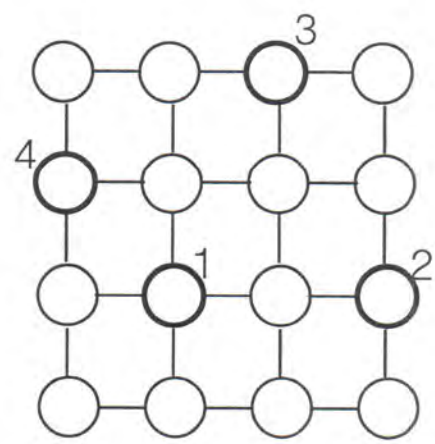
(1.4) **(4 points)** Write down the feature representation $\phi(i)$ corresponding to our chosen kernel $K(i,j) = \phi(i) \cdot \phi(j)$. Specifically, define the kth coordinate of $\phi(i)$ as

$$\phi(i)_k = \begin{cases} 1, & \text{if } \underline{\text{node k is neighbor of i}} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

(1.5) **(6 points)** Consider the following simple grid graph with four nodes numbered and highlighted. These are the training nodes (training examples) we have labels for. Once labels are made explicit, we can run the kernel perceptron algorithm with our kernel on these nodes, in the order given, i.e., cycling through 1,2,3, and 4.

(a) Is it possible to label the highlighted nodes such that our kernel perceptron algorithm would make only 1 mistake in total. Please answer **Y** or **N** ( )

(b) Please label each of the highlighted nodes in the graph with a "+" or "-" such that the kernel perceptron algorithm, if run with these labels, would make a mistake only on nodes 1 and 3, converging after the first round.



Problem 2 The passive-aggressive algorithm differs from the perceptron algorithm in that it updates its parameters in response to each training example (x, y) by minimizing

$$\frac{\lambda}{2}\|\theta - \theta^{(k)}\|^2 + \text{Loss}(y\,\theta \cdot x) \tag{3}$$

with respect to θ where $\theta^{(k)}$ is the current setting of the parameters. Here $\lambda > 0$ is a parameter we have to set and $\text{Loss}(z)$ defines how we measure errors. We will adopt a squared Hinge loss throughout this problem:

$$\text{Loss}(z) = \begin{cases} (1-z)^2/2, & \text{if } z \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

This loss, like Hinge loss, is zero when $z \geq 1$ but increases as $(1-z)^2/2$ when z drops below 1. We have seen in lecture that passive-aggressive updates have the form

$$\theta^{(k+1)} = \theta^{(k)} + \eta_k \, y \, x \tag{5}$$

where η_k depends on λ, the loss function, as well as the example.

(2.1) **(2 points)** Suppose we have parameters $\theta^{(k)}$ and see a training example (x, y). If $y\theta^{(k)} \cdot x < 1$ before the update, is it possible that $y\theta^{(k+1)} \cdot x > 1$ after the update? Please answer **Y** or **N** ( )

(2.2) **(2 points)** With our chosen loss function, if $y\theta^{(k)} \cdot x < 1$ before the update, we can always just solve for $\theta^{(k+1)}$ by minimizing

$$\frac{\lambda}{2}\|\theta - \theta^{(k)}\|^2 + (1 - y\theta \cdot x)^2/2 \tag{6}$$

Please answer **Y** or **N** ( )

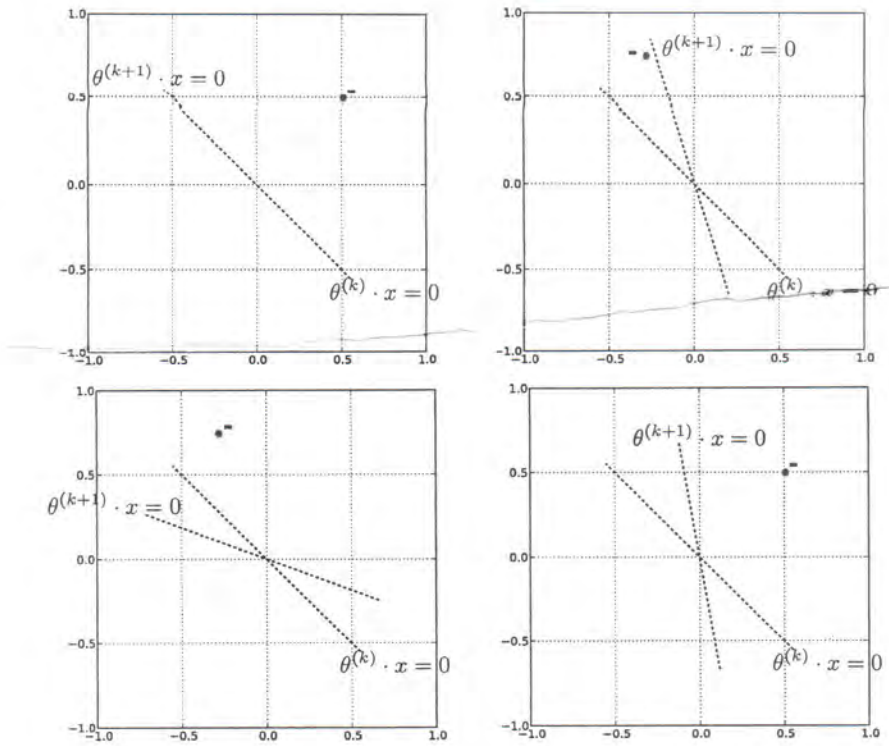
(2.3) **(4 points)** If $\theta^{(0)} = [0, 0]^T$, what is the value of $\theta^{(1)}$ in response to the first labeled training example (x, y)? Select the right expression for η_0 as a function of x, y, and (positive) λ.

$$(\quad) \qquad \eta_0 = \min\left\{\frac{\text{Loss}(0)}{\|x\|^2}, \frac{1}{\lambda}\right\} \tag{7}$$

$$(\quad) \qquad \eta_0 = \min\left\{\frac{1}{\|x\|^2}, \frac{1}{\lambda}\right\} \tag{8}$$

$$(\quad) \qquad \eta_0 = \frac{1}{\lambda + \|x\|^2} \tag{9}$$

(2.4) **(6 points)** The passive-aggressive algorithm will result in slightly different updates depending on how we set λ. Below you will see four plots representing decision boundaries before/after the update obtained with some value of λ in response to a negatively labeled point shown in the figure. Some of the plots cannot arise in this way, however. Please a) draw the orientation of $\theta^{(k)}$ (parameters before the update) in each figure and b) cross out all the plots which are **NOT** possible with any value $\lambda > 0$.



Problem 3 Recommender problems are everywhere, from Netflix, Amazon, to Google News. Here we aim to reconstruct a matrix of user-item preferences using two different methods learned in class, matrix factorization and neural networks. Yes, neural networks find their way here as well...

Suppose we have n users $a \in \{1, \ldots, n\}$ and m items $i \in \{1, \ldots, n\}$. Since each user is likely to provide ratings for only a small subset of possible items, we must heavily constrain the models so as not to overfit. In fact, our goal here is to understand how constrained they really are.

(3.1) **(4 points)** We begin by estimating a simple rank-1 model $X_{ai} = u_a v_i$ where $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ are just scalar parameters associated with users u_a and items v_i, respectively. Please select which (if any) of the following 2x2 matrices **cannot** be reproduced by the rank-1 model.

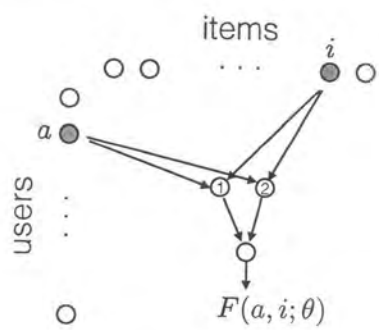
	v_1	v_2
u_a	+1	+1
u_b	+1	-1

(X)

	v_1	v_2
u_a	-1	+1
u_b	+1	-1

( )

We will also try to understand a simple neural network model applied to the same problem. To this end, we introduce an input unit corresponding to each user and each item. When querying about a selected entry, (a, i), only the ath user input unit and i^{th} item input unit are active (set to 1), the rest are equal to zero. Thus only the outgoing weights from these two units matter for predicting the value for (a, i). Figure below provides a schematic representation of the model.



User a has two outgoing weights, U_{a1} and U_{a2}, and item i has two outgoing weights, V_{i1} and V_{i2}. These weights are fed as inputs to the two hidden units in the model. The hidden units evaluate

$$z_1 = U_{a1} + V_{i1}, \quad f(z_1) = \max\{0, z_1\} \tag{10}$$
$$z_2 = U_{a2} + V_{i2}, \quad f(z_2) = \max\{0, z_2\} \tag{11}$$

Thus, for (a, i) entry, our network outputs

$$F(a, i; \theta) = W_1 f(z_1) + W_2 f(z_2) + W_0 \tag{12}$$

In a vector form, each user a has a two-dimensional vector of outgoing weights $\vec{u}_a = [U_{a1}, U_{a2}]^T$ and each item i has $\vec{v}_i = [V_{i1}, V_{i2}]^T$. The input received by the hidden units is then $\vec{z} = [z_1, z_2]^T = \vec{u}_a + \vec{v}_i$.

Consider again a simple matrix of ratings where we have two users, $\{a, b\}$, and two items $\{1, 2\}$. We will fix the first layer weights as shown in the Figure below.

(3.1)

$$\text{rank}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = 2 \qquad \text{rank}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = 1$$

won't work with rank 1 model

Get rank using ref([A]) in calculator

rank = # non-zero rows

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 INTRODUCTION TO MACHINE LEARNING

Midterm exam (March 18, 2014)

Your name & ID: _____

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Problem 1.1	Problem 2.1	Problem 3.1	Problem 3.2	Problem 4.1	Total

(1.1) The passive-aggressive algorithm is an on-line algorithm that updates its parameters in response to each training example and label. If $x \in \mathcal{R}^2$ is the example, y the corresponding label, and $\theta^{(k)}$ the parameter vector prior to seeing (x, y), then the algorithm finds $\theta^{(k+1)}$ by minimizing

$$\frac{1}{2}\|\theta - \theta^{(k)}\|^2 \text{ subject to } \mathrm{Loss}(y\theta \cdot x) = 0 \tag{1}$$

with respect to θ. Here $\mathrm{Loss}(z) = \max\{0, 1 - z\}$ is the Hinge loss. The loss is zero when $z \geq 1$. We have seen in lecture that the algorithm updates the parameters similarly to the perceptron algorithm. In other words,

$$\theta^{(k+1)} = \theta^{(k)} + \eta_k \, yx \tag{2}$$

where $\eta_k = 1$ for the perceptron algorithm but this is not typically the case for the passive aggressive algorithm. Note that x is a two dimensional vector for us here.

(a) **(3 points)** If $\theta^{(0)} = [0, 0]^T$, what is the value of $\theta^{(1)}$ in response to the first labeled training example (x, y)? Write down an expression for $\theta^{(1)}$ as a function of x and y.

(b) **(3 points)** The update changes based on where we start. Suppose $\theta^{(0)} = [-1, 0]^T$ and $x = [1, 1]^T$, $y = -1$. What is the numerical value of η_0 in the update if we receive this particular (x, y) as the first example?

(3.1)

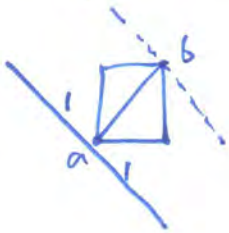
(a) Yes by observation

(b) ~~yes~~ No by observation

(c) see graph

(d) SVM Summary

(i)
- boundry in the middle
- margins as far apart
- points on margin = support vectors

(ii) 

$= \sqrt{2}$

(3.1) We can obtain a non-linear classifier by mapping each example x to a non-linear feature vector $\phi(x)$. Consider a simple classification problem in two dimensions shown in Figure 2. The points $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$ are labeled $y^{(1)} = 1$, $y^{(2)} = 1$, and $y^{(3)} = -1$.

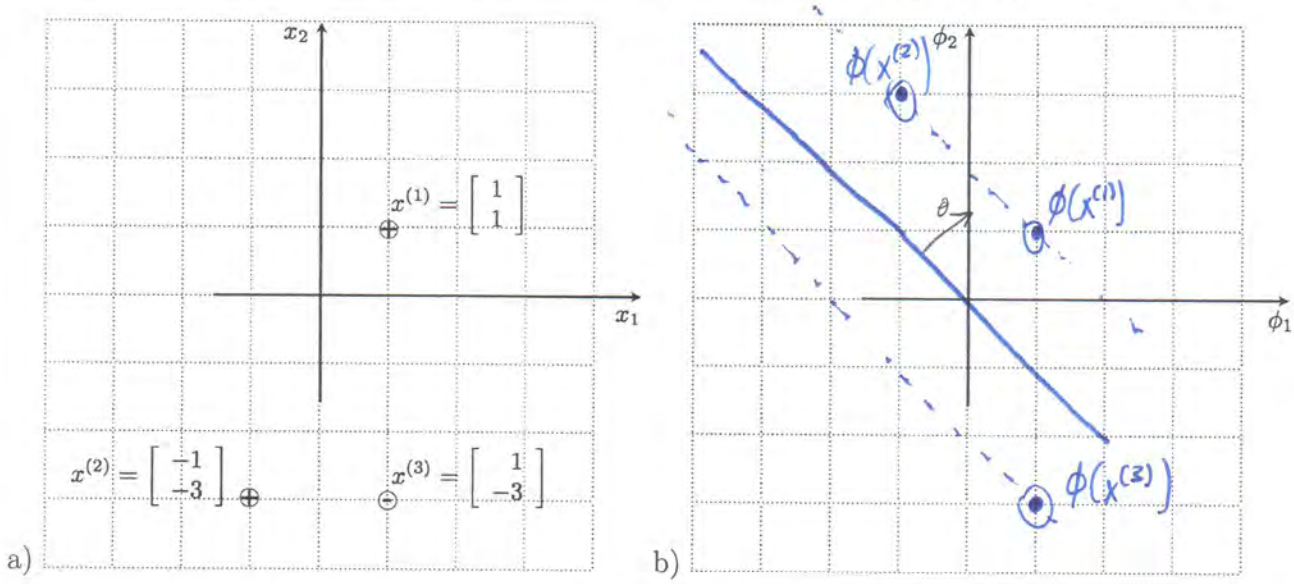


Figure 2: a) A labeled training set with three points. The labels are shown inside the circles. b) Examples in feature coordinates (to be mapped).

(a) **(2 points)** Are the points in Figure 2 linearly separable (Y/N) (Y)

(b) **(2 points)** Are the points in Figure 2 linearly separable through origin (Y/N) (N)

(c) **(2 points)** Consider a two dimensional feature mapping $\phi(x) = [x_1, x_2 x_1]^T$ where x_1 and x_2 are the coordinates of x. Map the three training examples in Figure 2a) to their feature coordinates in Figure 2b).

(d) We will use the support vector machine to solve the classification problem in the feature space. In other words, we will find $\theta = [\theta_1, \theta_2]^T$ that minimizes

$$\frac{1}{2}\|\theta\|^2 \text{ subject to } y^{(i)}\theta \cdot \phi(x^{(i)}) \geq 1, \quad i = 1, 2, 3 \tag{4}$$

We denote the solution by $\hat{\theta}$.

(i) **(6 points)** Draw the resulting $\hat{\theta}$ (orientation is fine), the corresponding decision boundary, and the margin boundaries in the feature coordinates in Figure 2b). Circle the support vectors.

(ii) **(2 points)** What is the value of the resulting margin? ($\sqrt{2}$)

(iii) **(3 points)** What is the value of $\|\hat{\theta}\|$? ($1/\sqrt{2}$)

(e) **(4 points)** Draw the resulting decision boundary $\{x : \hat{\theta} \cdot \phi(x) = 0\}$ in the original x-coordinates in Figure 2a). The boundary divides the space in two or more areas. Mark each area based on how the points there would be classified. Show any calculations below.

(3.2) Perhaps we can solve the classification problem in Figure 2a) a bit easier using kernel perceptron. Recall that in this case we would predict the label for each point x according to

$$\sum_{j=1}^{3} \alpha_j y^{(j)} K(x^{(j)}, x) \tag{5}$$

The algorithm requires that we specify a kernel function $K(x, x')$. But, for training, it suffices to just have the kernel function evaluated on the training examples, i.e., have the 3x3 matrix $K_{ij} = K(x^{(i)}, x^{(j)})$, $i, j = 1, 2, 3$, known as the Gram matrix. The matrix K in our case was

$$K = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix} \tag{6}$$

(a) **(6 points)** Based on what you know about kernels, which of the following matrices could **not** be Gram matrices. Check all that apply.

negative

$$(\ \ ) \begin{bmatrix} 2 & -1 & 0 \\ -1 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (X) \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}, \quad (X) \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \tag{7}$$

$G_{ij} \neq G_{ji}$

(b) **(4 points)** Now, let's apply the kernel perceptron using the kernel K given in Eq (6). We cycle through the three training points in the order $i = 1, 2, 3$. What is the **the 2nd** misclassified point? ( )

(iii) Margin $= 1/\|\hat{\theta}\|$ $\Rightarrow \|\hat{\theta}\| = 1/\text{margin} = 1/\sqrt{2}$

(e) ???

(3.2)

a) Gram Matrix: each row contains inner products (similarities) of a feature vector w/ every other feature vector.

- Entries in diagonal must be positive or 0
- $G_{i,j} = G_{j,i}$ $\begin{bmatrix} \diagup & 0 \\ 0 & \diagup \end{bmatrix}$

b) if $y^{(t)}\left(\sum_{i=1}^{n} \alpha_i y^{(i)}\left[K(x^{(i)}, x^{(t)}) + 1\right]\right) \leq 0$

$\vdots$

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 Introduction to Machine Learning

Midterm exam (March 17, 2016)

Your name & ID: _____

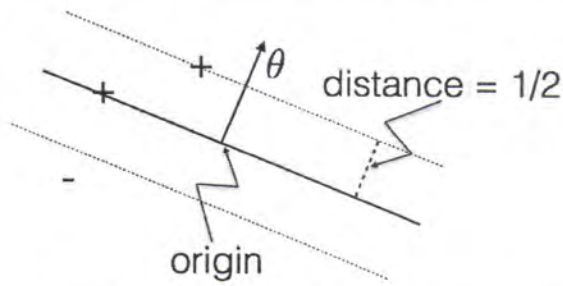
- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Total
20	7	6	4	23	60

Problem 1

(1.1) **(2 points)** Consider a set of linearly separable examples $(x^{(i)}, y^{(i)})$, $i = 1, \ldots, n$, $x^{(i)} \in \mathcal{R}^d$. If we modify the input vectors by eliminating the first coordinate (e.g., setting it to zero for all $x^{(i)}$), are the resulting set of examples guaranteed to be linearly separable? (Y/N)(N)

(1.2) **(2 points)** Can the perceptron algorithm be viewed as a stochastic gradient descent algorithm, just applied to minimize the zero one loss? (Y/N) (N)



Decision boundary together with the margin boundaries

(1.3) **(6 points)** SVM is an offline algorithm for estimating a linear separator. If we omit the offset parameter, SVM finds θ that minimizes the objective function

$$\left[\frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h(y^{(i)}\theta \cdot x^{(i)})\right] + \frac{\lambda}{2}\|\theta\|^2 \tag{1}$$

Suppose we set $\lambda = 1$ and $n = 3$ as in the figure above. What is the value of the objective function based on the figure?

Based on the figure, is there a solution which has lower loss and smaller $\|\theta\|$?

(1.1) No

(1.2) No

$n = 3$
$\lambda = 1$

(1.3) margin $= 1/2$ $\quad \|\theta\| = 1/\text{margin} = 2$

Objective function: $\left[\frac{1}{n}\sum_{i=1}^{n} \text{Loss}_h\left(y^{(i)}\theta \cdot x^{(i)}\right)\right] + \frac{\lambda}{2}\|\theta\|^2$

$$\frac{1}{3}\left[\text{Loss}_h\left(y^{(1)}\theta \cdot x^{(1)}\right) + \text{Loss}_h\left(y^{(2)}\theta \cdot x^{(2)}\right) + \text{Loss}_h\left(y^{(3)}\theta \cdot x^{(3)}\right)\right] + \frac{1}{2}(2)^2$$

$$= 1/3 + 2$$

smaller $\|\theta\| \Rightarrow$ larger margin

could shift boundry down and extend?

$\Rightarrow$ YES

(1.4) Take gradient

(1.4) **(2 points)** Pegasos is an on-line stochastic gradient descent (SGD) method for optimizing the SVM objective. Which one of the following update rules is the correct Pegasos update in response to a training example (x, y)? Mark only the correct one or leave blank if all are incorrect.

( ) $\hat{\theta} = \theta + \eta(yx - \lambda\theta)$ (2)

( ) $\hat{\theta} = \theta + \eta yx[\![1 - y\theta \cdot x \geq 0]\!]$ (3)

(X) $\hat{\theta} = \theta + \eta\big(yx[\![1 - y\theta \cdot x \geq 0]\!] - \lambda\theta\big)$ (4)

( ) $\hat{\theta} = \theta + \eta\big(yx[\![1 - y\theta \cdot x \geq 0]\!]/n - \lambda\theta\big)$ (5)

(1.5) **(2 points)** Passive-Aggressive algorithm is more akin to the perceptron algorithm, though it operates with Hinge loss rather than mistakes. Its update rule in response to a training example (x, y) is simply $\hat{\theta} = \theta + \eta yx$ where η comes from minimizing

$$\text{Loss}_h(y\hat{\theta} \cdot x) + \frac{\lambda}{2}\|\hat{\theta} - \theta\|^2 \tag{6}$$

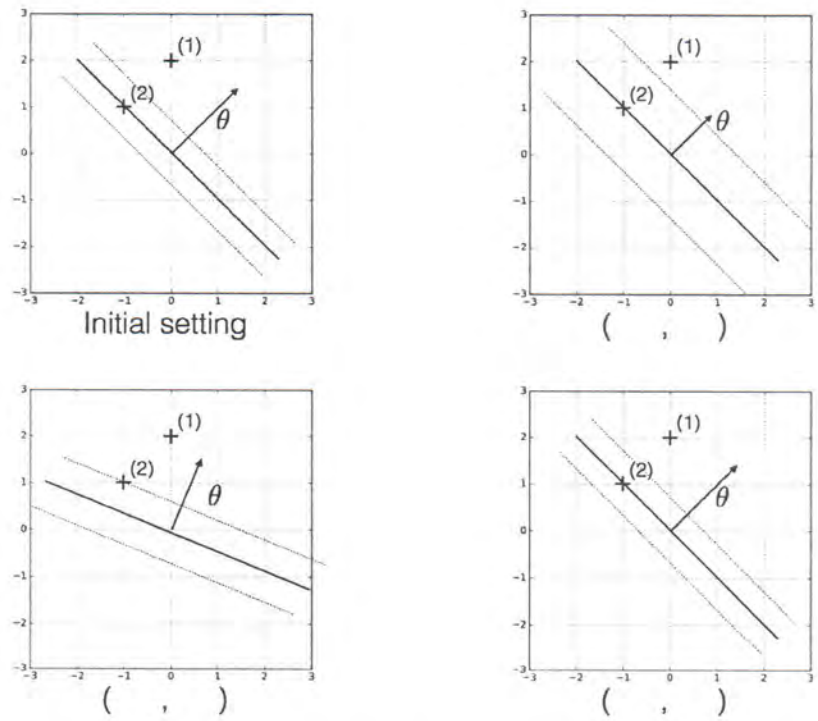
with respect to $\hat{\theta}$. We will use θ as the previous setting of the parameters and $\hat{\theta}$ as the new setting after updating in response to (x, y). Which of the following statements are correct?

( ) $y\hat{\theta} \cdot x > 1$ if and only if $y\theta \cdot x > 1$ (7)

( ) $y\hat{\theta} \cdot x = 1$ if $y\theta \cdot x < 1$ (8)

( ) $\text{Loss}_h(y\hat{\theta} \cdot x) \leq \text{Loss}_h(y\theta \cdot x)$ always (9)

(1.6) **(6 points)** The top left plot in the figure below shows the initial parameter vector θ along with the associated margin boundaries. We can update the parameters based on either of the two positive points, (1) or (2), and by using either the Passive-Aggressive or the Pegasos algorithm. Please assign the remaining plots to the most plausible combination of (algorithm,point) that was used to generate the figure. You should mark each of the three figures as one of (PA,1), (PA,2), (Peg,1), or (Peg,2).



Problem 2 Given training samples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$, ridge regression seeks to predict each response $y^{(i)}$ with a linear model $\theta \cdot x^{(i)}$ while encouraging θ to have a small norm. We omit the offset parameter for simplicity. Specifically, θ is estimated by minimizing

$$\left[\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta \cdot x^{(i)})^2/2\right] + \frac{\lambda}{2}\|\theta\|^2, \tag{10}$$

where $\lambda \geq 0$ is the regularization parameter, typically chosen in advance.

(2.1) **(2 points)** What is the solution $\hat{\theta}$ that minimizes Eq(10) if $\lambda \to \infty$?

6.036 Official Cheat Sheet

- A (hyper-)plane is a set of points $x \in \mathcal{R}^d$ such that $\theta \cdot x + \theta_0 = 0$. Vector θ is normal to the plane. The signed distance of any point x from the plane is $(\theta \cdot x + \theta_0)/\|\theta\|$. The value of distance is positive on the side where θ points to, and negative on the other side.
- A linear classifier with offset: $h(x;\theta) = \text{sign}(\theta \cdot x + \theta_0)$
- Training error (classification error): $\epsilon_n(h) = \frac{1}{n}\sum_{i=1}^{n}[[y^{(i)} \neq h(x^{(i)})]]$ The same formula can be used to calculate the test error.
- Loss functions:

$$z = y(\theta \cdot x + \theta_0) \text{ (agreement)}$$

$$\text{Loss}_{0,1}(z) = [[z \leq 0]]$$

$$\text{Loss}_{hinge}(z) = \max\{1 - z, 0\}$$

- Passive-aggressive algorithm (no offset): At step k, in response to (x, y), find $\theta^{(k+1)}$ that minimizes $\lambda\|\theta - \theta^{(k)}\|^2/2 + Loss_{hinge}(y\theta \cdot x)$
- SVM: Find a maximum-margin classifier by minimizing $\frac{1}{n}\sum_i \text{Loss}_{hinge}(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)) + \frac{\lambda}{2}||\theta||^2$ which can be done using stochastic gradient descent (Pegasos).
- Linear regression: predict value $\theta \cdot x + \theta_0$ by minimizing $\frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \theta \cdot x^{(i)} - \theta_0)^2/2$
- Low-rank matrix factorization for collaborative filtering: Minimize $J(U,V) = \sum_{(a,i)\in D}(Y_{ai} - [UV^T]_{ai})^2/2 + \frac{\lambda}{2}\sum_{a=1}^{n}\sum_{j=1}^{k}U_{aj}^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\sum_{j=1}^{k}V_{ij}^2$ Can be solved iteratively by fixing one matrix and using linear regression to find the other.
- Kernels: $K(x, x') = \phi(x) \cdot \phi(x')$

Kernel	form
Linear	$x \cdot x'$
Quadratic	$x \cdot x' + (x \cdot x')^2$
Radial basis	$\exp(-\|\|x - x'\|\|^2/2)$

- Kernel Perceptron (with offset): Cycles through each point t=1,..n and checks if $y^{(t)}(\sum_{i=1}^{n}\alpha_i y^{(i)}[K(x^{(i)}, x^{(t)}) + 1]) \leq 0$. If true, $\alpha_t = \alpha_t + 1$.
- Neural Nets:
  - output of neuron with activation function f is $Z = \sum_{j=1}^{m} f(z_j)V_j + V_0$
  - a unit in layer l with net input $z_j^l = \sum_i f_i^{l-1}w_{ij}^l + w_{0j}^l$ will output $f_j^l = f(z_j^l)$
  - common activation functions include ReLU ($f(z) = \max\{0, z\}$), tanh, and the identity function
  - backpropagation: $\delta_i^{l-1} = f'(z_i^{l-1})\sum_j w_{ij}^l\delta_j^l$
- Good luck! ☺

Unsupervised Learning & Clustering

- find clusters/meaning in unlabeled data
  - assign each data point to exactly one cluster
  - assign clusters in a way that minimizes total cost
  - a point's cost is measured relative to the cluster's representative:
    - the centroid (mean of points in cluster) in k-means
    - the exemplar (cost minimizing point from cluster) in k-medoids
  - a simple measurement of cost is mean squared distance
  - number of clusters used affects quality of solutions
    - if every point had own cluster, cost would be zero
      - but would provide no meaningful information
    - * see p87 for some methods for choosing optimal number
  - a Voronoi diagram shows the representatives and boundaries between clusters
- K-means algorithm
  > see cheatsheet for K-means formulas
  - 1. initialize cluster means
  - 2. repeat as needed:
    - a. assign each point to cluster that minimizes cost
    - b. calculate new mean for each cluster (average of points in cluster)
  - cost is monotonically decreasing
  - converges, but not necessarily to optimal solution
    - optimality depends on initialization
    - * see p87 for one solution, the K-means++ initializer
- K-medoids algorithm
  - like K-means, but uses an exemplar instead of centroid
  - 1. initialize cluster exemplars
  - 2. repeat as needed:
    - a. assign each point to cluster that minimizes cost
    - b. find exemplar (member that minimizes cost) for each cluster

Generative Models & Mixtures

- like clusters, but are probabilistic instead of cost based
  - assign parameters to create maximum likelihood (ML) of generating data
  - assign each cluster a mixing proportion (probability of generating points)
  - assign each data point to each cluster with some probability
- Gaussian mixture
  - parameters θ
    - k mixing proportions p that sum to 1
    - k d-dimensional means μ
    - k variances σ^2

> k-1 + k*d + k independent parameters (2013 2.1)
- points x
  - likelihood $P(x|\theta) = \Sigma\_k\ p\ N(x;\mu,\sigma^2 I)$
  - log likelihood $\log(P(x|\theta))$
- mixtures over data D
  - k mixtures represented by circles centered at μ with radius σ
  - likelihood $L(D|\theta) = \Pi\_x\ P(x|\theta)$
  - log likelihood $\Sigma\_x \log(P(x|\theta))$
- maximum likelihood (ML)
  > see cheatsheet for ML μ and σ^2 formulas
  - μ is d-dimensional
  - σ^2 is scalar and must be averaged across all d-dimensions
- Expectation-Maximization (EM) algorithm
  > see cheatsheet for EM formulas
  - 1. initialize parameters
  - 2. repeat as needed:
    - a. E step: find conditional probability p(j|i) of each point x^i having been generated by each cluster j
    - b. M step: find new ML mixing proportions, means, and variances

Bayesian Networks
- events are modeled as a directed graph between variables
  - nodes represent variables
  - directed edges (arcs) represent probabilistic dependencies
    - root nodes have no dependencies
  - each variable has a probability conditional on its parents
    - this is visualized with a probability table of the form

```
p(A|B,C): B,C | T F
          ---------
          T,T | 0 1
          T,F | 1 0
          F,T | 1 0
          F,F | 0 1
```

    - each row in the table must sum to one
  > practice transforming amongst graphs, probability equations, and tables
- probability concepts
  - joint probability, conditional probability, marginal probability
  - chain rule $P(A,B,C,\ldots) = P(B,C,\ldots|A)P(A) = P(C,\ldots|A,B)P(B|A)P(A)$ ...
  - independence, marginal independence, induced dependence
    - B,C are marginally independent of A if $\Sigma\_A\ P(B,C|A)P(A) = P(B)P(C)$

- this means that if A is unknown, B and C are fully independent
- but if A is dependent on B and C, and A is known, then
  - there is a reverse dependency $P(B,C|A)$, usually not equal to $P(B)P(C)$
  - this is called induced dependence between B and C

* see explaining away p111

Hidden Markov Models (HMM)

- at each time-step, we transition to a state and make an observation
  - there are N possible states with one of them being the "stop state"
  - there is a set (alphabet) Σ of possible observations (output symbols)
  - there is a probability $p(j|i)$ that the next state is j if the current one is i
    - $(N-1)N$ of these parameters with $(N-1)^2$ of them independent
    - not defined for the stop state
    - next state is independent of all but the current state
  - there is a probability $p(o|j)$ of observing symbol o when in state j
    - $(N-1)(|\Sigma|)$ of these parameters with $(N-1)(|\Sigma|-1)$ of them independent
    - not defined for the stop state
    - current observation is independent of all but the current state
  - there is a probability π_i of starting in state i
    - N of these parameters with N-1 of them independent
- the states traversed and observations made form a Markov sequence
  - sequence of states/labels/tags $Y = (y1, y2, ..., y_n)$
  - sequence of corresponding observations $X = (x1, x2, ..., x_n)$
  - unknown model parameters θ can be estimated from data
    - $p(j|i)$ = count(transition from i to j)/count(was in i)
    - $p(o|j)$ = count(observed o while in j)/count(was in j)
    - π_i = [[starting state was i]]
    > EM algorithm for HMM is not on final (@1349)
    > Forward-Backward algorithm is not on final (@1410)
  - likelihood of model generating given data can be found from θ
    - $p(X,Y;\theta) = (\pi_{y1})p(x1|y1)p(y2|y1)p(x2|y2)...$
    - $p(X;\theta) = \Sigma_Y P(X|Y;\theta)P(Y)$
  - the most likely Y given X can be found
    - by brute force enumerating each possible Y and comparing $p(X,Y;\theta)$
    > Viterbi algorithm is not on final (@1410)
- the model is best visualized with a state diagram
  - nodes correspond to states
  - transitions are labeled with their probability

MDP & Reinforcement Learning

- agent does actions that help it move between states and earn rewards
  - doing action a in state s moves it to state s' with probability T(s,a,s')
  - making the transition in this way earns it reward R(s,a,s')
- agent tries to maximize its utility, not its reward
  - maximizing pure reward is often a bad idea
    - agent may loop infinitely to earn infinite reward
    - setting a finite amount of moves (horizon) will complicate things
      - every choice then depends both on rewards and time left
    - agent may delay earning any reward to earn more in the distant future
  - utility is measured as $\Sigma_t \gamma^t R(s_t,a_t,s_{t+1})$ beginning at $t=0$
    - reward at time $t=0$ is earned in full
    - reward at time $t\geq 1$ is discounted by γ^t
    - γ is chosen such that $0\leq\gamma<1$
      - implies finite utility for finite values of R (geometric series)
    - smaller γ leads to greater preference for immediate reward
- Markov decision process (MDP)
  - transition probabilities and rewards are already known
  - agent chooses action in state s that maximizes earned "value"
    - optimal policy $\pi^*(s) = \text{argmax}_a Q^*(s,a)$
  - "value" is expected utility when starting at s and acting optimally
    - $V^*(s) = \max_a Q^*(s,a) = Q^*(s,\pi^*(s))$
  - "Q-value" is expected utility when doing a at s, then acting optimally
    - $Q^*(s,a) = \Sigma_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
  - values can be found using the value iteration algorithm
    > see cheatsheet for algorithm
    - converges to optimal value $V^*(s)$
  - Q-values can be found using the Q-value iteration algorithm
    > see cheatsheet for algorithm
    - converges to optimal value $Q^*(s,a)$
- reinforcement learning
  - transition probabilities and rewards are initially unknown
  - model based learning
    - use observations to estimate transition probabilities and rewards
    - use this data to estimate π^* and Q^*
    - in practice, noise in data will make it hard to infer anything useful
    - additionally, storing all the estimates can quickly become prohibitive
  - model-free learning
    - Q-learning algorithm
      > see cheatsheet for algorithm
      - uses an exponential running average to estimate Q^*

- no need to directly estimate transition probabilities or rewards
- gives recent observations more weight than old ones
  - helps overcome noise and changes in world
- * see p131 for convergence conditions

- exploration/exploitation trade-off
  - random policy choices will usually be suboptimal
  - but if don't explore, will never improve policy
  - with some probability ε choose action at random and explore
  - with some probability $(1-\varepsilon)$ choose the best currently known policy
  - as knowledge is gained and more options sampled, reduce ε

2017 Final

6.036 Official Cheat Sheet

- A (hyper-)plane is a set of points $x \in \mathcal{R}^d$ such that $\theta \cdot x + \theta_0 = 0$. Vector θ is normal to the plane. The signed distance of any point x from the plane is $(\theta \cdot x + \theta_0)/\|\theta\|$. The value of distance is positive on the side where θ points to, and negative on the other side.
- A linear classifier with offset: $h(x;\theta) = \text{sign}(\theta \cdot x + \theta_0)$
- Training error (classification error): $\epsilon_n(h) = \frac{1}{n}\sum_{i=1}^n [[y^{(i)} \neq h(x^{(i)})]]$
- Loss functions:

$$z = y(\theta \cdot x + \theta_0) \text{ (agreement)}$$
$$\text{Loss}_{0,1}(z) = [[z \leq 0]]$$
$$\text{Loss}_h(z) = \max\{1 - z, 0\}$$

- SVM: Finds a large margin classifier by minimizing

  $\frac{1}{n}\sum_i \text{Loss}_h(y^{(i)}(\theta \cdot x^{(i)} + \theta_0)) + \frac{\lambda}{2}||\theta||^2$

  which can be done using stochastic gradient descent (Pegasos).
- Linear regression:
  finds the parameters of a linear predictor $\theta \cdot x + \theta_0$ by minimizing

  $\frac{\lambda}{2}||\theta||^2 + \frac{1}{n}\sum_{i=1}^n (y^{(i)} - \theta \cdot x^{(i)} - \theta_0)^2/2$
- Low-rank matrix factorization for collaborative filtering: Minimize

  $J(U,V) = \sum_{(a,i)\in D}(Y_{ai} - [UV^T]_{ai})^2/2 + \frac{\lambda}{2}\sum_{a=1}^n \sum_{j=1}^k U_{aj}^2 + \frac{\lambda}{2}\sum_{i=1}^m \sum_{j=1}^k V_{ij}^2$

  Can be solved iteratively by fixing one matrix and using linear regression to find the other.
- ~~Kernels: $K(x,x') = \phi(x) \cdot \phi(x')$~~
- Kernel Perceptron (with offset): Cycles through each point t=1,..n and checks if

Kernel	form
Linear	$x \cdot x'$
Quadratic	$x \cdot x' + (x \cdot x')^2$
Radial basis	$\exp(-\|\|x - x'\|\|^2/2)$

  $y^{(t)}(\sum_{i=1}^n \alpha_i y^{(i)}[K(x^{(i)}, x^{(t)}) + 1]) \leq 0$. If true, $\alpha_t = \alpha_t + 1$.
- Neural Nets:
  - unit i in layer l evaluates its aggregate input based on the previous layer as $z_i^l = \sum_{j=1}^m f(z_j^{l-1})w_{ji}^l + w_{0i}^l$ and its activation as $f(z_i^l)$
  - common activation functions include ReLU ($f(z) = \max\{0, z\}$), tanh, and the identity function
  - backpropagation: $\delta_j^{l-1} = f'(z_j^{l-1})\sum_i w_{ji}^l \delta_i^l$
- RNN equations are given if used
- Generalization:

  In the realizable case, $\epsilon(\hat{h}) \leq \epsilon_n(\hat{h}) + \frac{\log|H| + \log(\frac{1}{\delta})}{n}$ where $\epsilon_n(h)$ is the training error and H is a finite set of classifiers.

  In the non-realizable case, we obtain a weaker bound: $\epsilon(\hat{h}) \leq \epsilon_n(\hat{h}) + \sqrt{\frac{\log|H| + \log(\frac{1}{\sigma})}{2n}}$.

  When H is not finite, $\log|H|$ will be roughly speaking replaced by the growth function $\log N_H(n)$ which relates to the VC-dimension.
- K-Means

  $\text{cost}(\mu^{(1)}, \ldots, \mu^{(k)}) = \sum_{i=1}^n \min_{j=1,\ldots,k} \|x^{(i)} - \mu^{(j)}\|^2$

  1. initialize $\mu^{(1)}, \ldots, \mu^{(k)}$
  2. $\delta(j|i) = [[\, j = \text{argmin}_l \|x^{(i)} - \mu^{(l)}\|^2 \,]]$

2017 Final

6.036 Official Cheat Sheet

3. $\hat{\mu}^{(j)} = \frac{1}{\sum_{i=1}^{n} \delta(j|i)} \sum_{i=1}^{n} \delta(j|i) x^{(i)}$

- log-likelihood $\ell(D;\theta) = \sum_{i=1}^{n} \log P(x^{(i)};\theta)$
- max-likelihood estimates for $N(x;\mu,\sigma^2 I) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp(-\frac{1}{2\sigma^2}\|x-\mu\|^2)$
  - If $x \in R$ (1-dimensional): $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$, $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)}-\hat{\mu})^2$
  - If $x \in R^d$ (d-dimensional): $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$, $\hat{\sigma}^2 = \frac{1}{dn}\sum_{i=1}^{n}\|x^{(i)}-\hat{\mu}\|^2$
- EM for Gaussians:
  1. initialize $\theta = \{p_1, ..., p_k, \mu^{(1)}, ..., \mu^{(k)}, \sigma_1^2, ..., \sigma_k^2\}$
  2. E-Step: $p(j|i) = \frac{p_j N(x^{(i)};\mu^{(j)},\sigma_j^2 I)}{\sum_{z=1}^{k} p_z N(x^{(i)};\mu^{(z)},\sigma_z^2 I)}$
  3. M-step: $\max_\theta \sum_{i=1}^{n}\sum_{j=1}^{k} p(j|i)\log[p_j N(x^{(i)};\mu^{(j)},\sigma_j^2 I)]$, giving

     $p_j = \frac{\sum_{i=1}^{n} p(j|i)}{n}$

     $\hat{\mu}^{(j)} = \frac{1}{\sum_{i=1}^{n} p(j|i)} \sum_{i=1}^{n} p(j|i) x^{(i)}$

     $\hat{\sigma}_j^2 = \frac{1}{d\sum_{i=1}^{n} p(j|i)} \sum_{i=1}^{n} p(j|i)\|x^{(i)}-\hat{\mu}^{(j)}\|^2$
- Model selection: BIC criterion

  $BIC(D;\hat{\theta}) = l(D;\hat{\theta}) - \frac{\text{number of params}}{2}\log(n)$

  where D is the data with n examples and $\hat{\theta}$ is ML estimate of the parameters.
- HMM (as a Bayesian network)

  $P(Y_{1:T}, X_{1:T}) = P(Y_1)P(X_1|Y_1)\Pi_{t=2}^{T} P(Y_t|Y_{t-1})P(X_t|Y_t)$
- Q-Value Iteration Algorithm:
  1. $Q_0(s,a) = 0$
  2. $Q_{i+1}(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_i(s',a')]$
- Value Iteration Algorithm:
  1. $V_0(s) = 0$
  2. $V_{i+1}(s) = \max_a \left[\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i(s')]\right]$

  Note that $V_i(s') = \max_{a'} Q_i(s',a')$
- Q-learning

  Model-free estimation (to avoid explicitly computing T,R) $Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a)]$

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 Introduction to Machine Learning

Final exam (May 18, 2016)

Your name & ID: _____

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Prob 1	Prob 2	Prob 3	Prob 4	Prob 5	Prob 6	Prob 7	Total
8	16	19	13	12	12	5	85

1.1)

a) YES

By bayes net property

"Each variable is independent of its nondescendents given the state of its parents"

b) YES

$$P(Z \mid Y, X) = \frac{P(Z)\, P(Y \mid Z)\, \cancel{P(X \mid Y)}}{\sum_{z'} P(z')\, P(Y \mid z')\, \cancel{P(X \mid Y)}} = P(Z \mid Y)$$

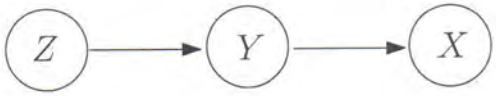
1.2)

NO

If $z = no$, knowing that $X = yes$ means that y must be "no". So X and Y are not independent

Problem 1 This problem revisits some basic questions about Bayesian Networks.

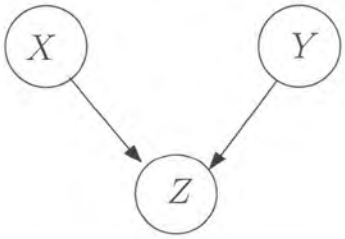
(1.1) **(4 points)** Consider the Bayes Net on three variables X, Y, Z defined by the figure below.



(a) Is $P(X|Y,Z) = P(X|Y)$? (Y/N) **( )**

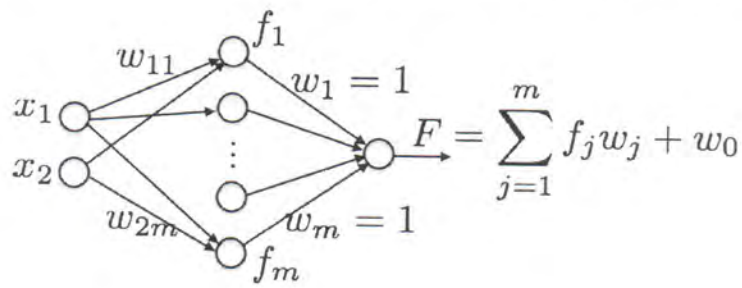
(b) Can we conclude $P(Z|Y,X) = P(Z|Y)$? Briefly justify your answer.

(1.2) **(4 points)** Suppose we have the same three random variables but now they are related in a different way.



For example, the value of X could be a yes/no suggestion from an advisor, and Y value (also yes/no) from another advisor. The president Z decides yes if both advisors agree (say yes), otherwise no. In this specific case, are X and Y independent given $Z = no$? Briefly justify your answer.

Problem 2 Consider a simple two layer neural network for classifying points on the plane. Our network has additional constraints beyond the two-layer architecture. The main constraint is that all the incoming weights to the output layer, w_j, $j = 1, \ldots, m$, are set equal to one, save for the offset parameter w_0 which remains adjustable. The hidden layer units can be chosen arbitrarily, including their number m.



The weights w_{ij}, $i = 0, 1, 2$, $j = 1, \ldots, m$, can be chosen as needed where w_{0j} is the offset parameter for the jth hidden unit.

(2.1) **(2 points)** If the activation function is sign($\cdot$), hidden units act as linear classifiers and can be drawn graphically as such. Write down the normal vector to the decision boundary of the linear classifier corresponding to the ith hidden unit?

$F_i = \text{sign}(x \cdot \theta + w_{0i}) \qquad \theta = \begin{bmatrix} w_{1i} \\ w_{2i} \end{bmatrix}$

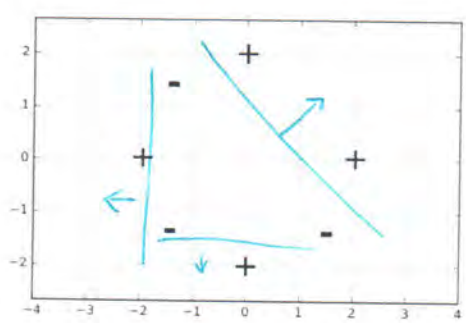


Figure NN1: training set of points

(2.2) **(6 points)** Figure NN1 shows the points we wish to classify correctly. Please draw graphically (as linear classifiers, including orientation) the smallest number of hidden units that enable our constrained output layer to classify the points correctly. For this part you must assume that hidden units are ReLU units.

2.3)

$$w_1 = w_1 - \eta \frac{\delta Loss(F, y)}{\delta w_1}$$

$$w_1 = w_1 - \eta \frac{\delta Loss(F, y)}{\delta f_1} \frac{\delta f_1}{\delta w_i}$$

$$= w_1 - \eta \underbrace{\frac{\delta f_1}{\delta w_i}}_{f_1 = ReLu(w_1 f_0 + w_{10})} d_i$$

$$= w_1 - \eta [\![f_1 > 0]\!] d_i$$

(2.3) **(4 points)** Neural networks are powerful but can be challenging to train. Consider a simple deep architecture shown below where there is a single unit in each layer.

Each unit uses ReLU activation such that $f_i = \text{ReLU}(w_i f_{i-1} + w_{i0})$, $i = 1, \ldots, m$, where $f_0 = x$. The output unit is linear $F = w f_m + w_0$. For a given input x, we observe target output y, and measure loss $\text{Loss}(F, y)$, where F is the activation of the final linear unit in response to x. In order to train these models with gradient descent, we must be able to calculate gradients. Let

$$d_i = \frac{\partial}{\partial f_i} \text{Loss}(F, y), \quad i = 1, \ldots, m \tag{1}$$

Write down a gradient descent update rule for parameter w_1 using d_i, $i = 1, \ldots, m$.

$w_1 = w_1 - \eta [\![f_i > 0]\!] d_i$

(2.4) **(4 points)** Suppose $\frac{\partial}{\partial F} \text{Loss}(F, y) = 1$, i.e., we didn't quite predict the response correctly. In this case, which of the following statements are necessarily true in our deep architecture? Check all that apply.

( ) $d_{i-1} = w_i f_i d_i, \quad i = 2, \ldots, m$
(X) $d_{i-1} = w_i [\![f_i \geq 0]\!] d_i, \quad i = 2, \ldots, m$
(X) $d_m = w$
( ) $d_1 \to 0$ as m increases (vanishing gradient)

(2)

Problem 3

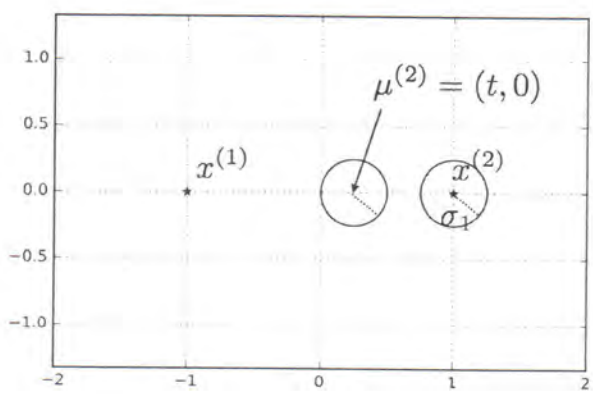
(3.1) **(3 points)** Suppose we fit the mean and the variance of a *single* two-dimensional spherical Gaussian distribution based on points $x^{(1)} = (-1, 0)$ and $x^{(2)} = (1, 0)$. What is the maximum likelihood estimate of the variance of this Gaussian?

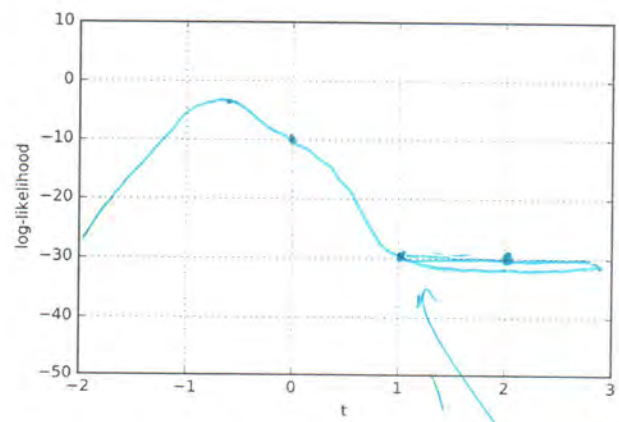
$1/2$

(3.2) **(6 points)** Let's consider a two component mixture of spherical Gaussians, i.e.,

$$P(x; \theta) = \pi_1 N(x; \mu^{(1)}, \sigma_1^2 I) + \pi_2 N(x; \mu^{(2)}, \sigma_2^2 I) \tag{3}$$

where we set $\pi_1 = \pi_2 = 0.5$, $\sigma_1 = \sigma_2 = 1/4$, and $\mu^{(1)} = (1, 0)$. We only vary $\mu^{(2)}$ by moving it along the horizontal axis, i.e., we set $\mu^{(2)} = (t, 0)$ where t varies. Figure below (left) illustrates the setting.





The log-likelihood of the data, i.e., points $x^{(1)} = (-1, 0)$ and $x^{(2)} = (1, 0)$, under a single Gaussian $N(x; \mu^{(1)}, \sigma_1^2 I)$ is approximately -30. Qualitatively sketch the log-likelihood of the data under the two component mixture model as a function of t where $\mu^{(2)} = (t, 0)$. Please use the figure above (right).

same as one Gaussian after $t=1$

3.1) From cheat sheet:

$$\sigma^2 = \frac{1}{dn} \sum_{i=1}^{n} \| x^{(i)} - \mu \|^2$$

$\mu = (0,0)$

$$\sigma^2 = \frac{1}{\underset{\substack{\uparrow \\ \text{dimension}}}{2} \times 2} \left[\| (-1,0) - (0,0) \|^2 + \| (1,0) - (0,0) \|^2 \right]$$

$$= 1/4 \cdot [1^2 + 1^2] = 1/2$$

3.3)

$p(1 \mid x^{(1)}) > p(2 \mid x^{(1)})$ = $x^{(1)}$ is more likely to be from cluster 1 than 2

FALSE

$p(1 \mid x^{(2)}) > p(2 \mid x^{(2)})$ = $x^{(2)}$ " " cluster 1 than 2

FALSE

$p(2 \mid x^{(1)}) > p(2 \mid x^{(2)})$

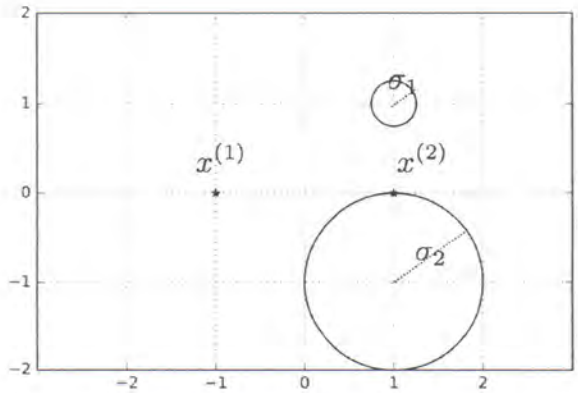


Figure MIX: Initialization of the two component mixture model

(3.3) **(4 points)** We now move on to estimating a two component mixture with a different initialization. The initialization is shown in Figure MIX. Specifically, $\mu^{(1)} = (1, 1)$, $\mu^{(2)} = (1, -1)$, $\sigma_1 = 1/4$, $\sigma_2 = 1$, and $\pi_1 = \pi_2 = 0.5$. Define

$$p(j|x^{(i)}) = \frac{\pi_j N(x^{(i)}; \mu^{(j)}, \sigma_j^2 I)}{\sum_{l=1,2} \pi_l N(x^{(i)}; \mu^{(l)}, \sigma_l^2 I)} \tag{4}$$

Which of the following statements are true for first E-step of the EM algorithm? Check all that apply.

$$\begin{array}{ll} (\quad) & p(1|x^{(1)}) > p(2|x^{(1)}) \\ (\quad) & p(1|x^{(2)}) > p(2|x^{(2)}) \\ (\times) & p(2|x^{(1)}) > p(2|x^{(2)}) \\ (\times) & \sum_{i=1,2} p(1|x^{(i)}) < 1 \end{array} \tag{5}$$

(3.4) **(6 points)** Consider the same mixture as in the previous question. Let $\hat{\mu}^{(1)}$, $\hat{\mu}^{(2)}$, $\hat{\sigma}_1$, $\hat{\sigma}_2$, and $\hat{\pi}_1$, $\hat{\pi}_2$ be the parameters after a single M-step. Which of the following statements are true? Check all that apply.

$$\begin{array}{ll} (\times) & \hat{\mu}_1^{(2)} < 0 \text{ (horizontal component)} \\ (\times) & \hat{\mu}^{(1)} \approx (1, 0) \\ (\times) & \hat{\pi}_1 \approx 0 \\ (\times) & \sigma_1 > \hat{\sigma}_1 \text{ (before vs after the 1st update)} \\ (\times) & \sigma_2 > \hat{\sigma}_2 \text{ (before vs after the 1st update)} \\ (\quad) & \hat{\mu}^{(2)} = (0, 0) \text{ when EM converges} \end{array} \tag{6}$$

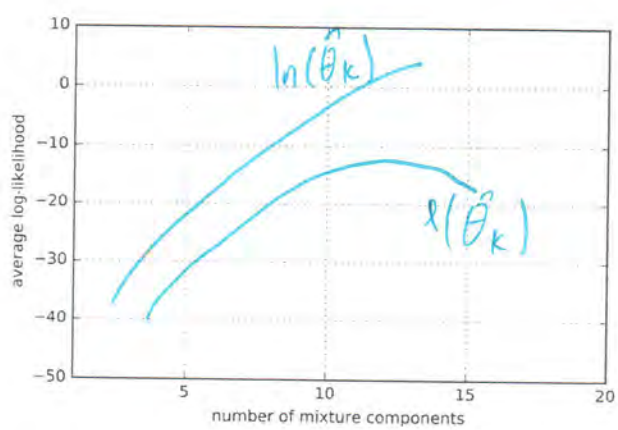
Problem 4 Understanding generalization is one of the core problems in machine learning. Here we scratch the surface a bit in the context of Gaussian mixture models. Suppose we have a training set with n examples $S_n = \{x^{(i)}, i = 1, \ldots, n\}$ sampled from some underlying distribution we don't know. Luckily, we can assume that the training and test examples are sampled from the same underlying distribution. Our goal is to find a mixture model that generalizes well to test examples.

We will measure the performance in terms of the average log-likelihood of data, i.e.,

$$l_n(\hat{\theta}_k) = \frac{1}{n} \sum_{i=1}^{n} \log P(x^{(i)}; \hat{\theta}_k) \tag{7}$$

where $P(x; \hat{\theta}_k)$ refers to a k-component mixture model where the associated parameters $\hat{\theta}_k$ were estimated from the training set via the EM-algorithm. The average test log-likelihood is measured in the same way, only with respect to test examples that were not available during training. We denote this average test log-likelihood as $l(\hat{\theta}_k)$.

(4.1) **(4 points)** In the plot below, qualitatively sketch how training and test log-likelihoods, $l_n(\hat{\theta}_k)$ and $l(\hat{\theta}_k)$, behave as a function of the number of mixture components or k.



(4.2) **(2 points)** Is it possible that the two curves would cross for some choice of training set? (Y/N) **(Y)**

(4.3) **(3 points)** Which of the following statements are true as n (the size of the training set) increases? Select all that apply.

( ) $l_n(\hat{\theta}_k)$ would typically increase, for any k

(✓) $l(\hat{\theta}_k)$ would typically increase, for any k — better models

(✓) $|l(\hat{\theta}_k) - l_n(\hat{\theta}_k)|$ would typically decrease, for any k — generalizes well

(8)

(4.4) **(4 points)** Our goal here is to estimate a mixture model that generalizes well. We make use of three sub-routines: 1) Init(k, data) that returns a randomly initialized Gaussian mixture model with k components with the help of 'data'; 2) EM(mix, data) which returns a trained mixture model based on 'data' and the initialization 'mix'; and 3) Eval(mix, data) returns the average log-likelihood of 'data' for a specific mixture model 'mix'. The pseudo-code below should return two things

(a) the mixture model that is likely to generalize the best

(b) a fair estimate of the resulting average test log-likelihood

The problem is that you only have a training set, randomly divided into five equal size pieces, train[i], $i = 1, \ldots, 5$. You are free to combine pieces into larger sets, e.g., train[1, 2, 3]. Fill in the datasets for the pseudo-code as well as the values to return corresponding to part (a) and (b)

```
For k = 1,...,K
   mix[k] = Init(k, train[1,2,3] )  % mixture with k components
   mix[k] = EM(mix[k], train[1,2,3] )
   LL[k]  = Eval(mix[k], train[4] )
mix* = EM( argmax(LL[k]) )          % answer to part (a)
LL*  = Eval( mix*, train[5] )       % answer to part (b)
```

must be a new set

Problem 5 Suppose you are playing an outdoors treasure hunt game. The playing field is divided into a grid as shown below. We know that there is gold buried underneath two of the squares, C and D. You have a heavy metal detector that beeps when on top of a square that may contain gold. However, the metal detector can be misled by the presence of other heavy metals (that we are uninterested in) and can thus beep falsely. From a prior calibration we know that the metal detector is able to detect the gold 75% of the time. The detector never beeps in the absence of gold.

A	B	C
D	E	F

The game becomes interesting because you do not directly observe where you are. With equal probability you start on either square 'A' or 'F'. At each step, you either stay put or move horizontally or vertically (not diagonally), all with equal probability. We will model your location as a hidden state $Y_t \in \{A, B, C, D, E, F\}$, whereas the observation X_t is a "beep" or "no beep".

(5.1) **(4 points)** Specify the initial state, state transition, and the emission probabilities for this HMM.

Y_1	
A	1/2
B	0
C	0
D	0
E	0
F	1/2

Y_{t-1} \ Y_t	A	B	C	D	E	F
A	1/3	1/3		1/3		
B	1/4	1/4	1/4		1/4	
C		1/3	1/3			1/3
D	1/3			1/3	1/3	
E		1/4		1/4	1/4	1/4
F			1/3		1/3	1/3

Y_t \ X_t	"beep"	"no beep"
A	0	1
B	0	1
C	.75	.25
D	.75	.25
E	0	1
F	0	1

(5.2) **(4 points)** Suppose you observed the following sequence of sounds from the metal detector: (X_1, X_2, X_3) =("no beep", "beep", "beep"). What are the possible sequences (Y_1, Y_2, Y_3) of locations that you may be in, given the three observations?

A → D → D

F → C → C

(5.3) **(4 points)** Which sequence of locations (Y_1, Y_2, Y_3) is most likely to occur together with (X_1 = "no beep", X_2 = "beep, X_3 = "beep"), and what is the corresponding joint probability $P(Y_1, Y_2, Y_3, X_1, X_2, X_3)$? If the answer isn't unique, select one.

A → D → D

$(1/2)(1) \cdot (1/3)(.75) \cdot (1/3)(.75)$

Start at A

No beep at A

A → D

beep at D

D → D

beep at D

Problem 6 You are running a 3 mile race. Every 10 minutes you must decide whether to walk or run for the next 10 minutes based on your current distance from the start (represented as states 0, 1, and 2 but no actions will be taken from state 3 because you will have already finished). If you walk, you will advance 1 mile over the next 10 minutes. If you run, you have a 50% chance to advance 1 mile and a 50% chance to advance 2 miles over the next 10 minutes. You want to finish the race, but running is tiring and takes effort. You will receive a reward of 10 for finishing the race (ending up in state 3). However, every time you run, you get an additional "reward" -1. You decide to use a Markov Decision Process with γ=0.5 to determine what action you should take from each state. The full table of transition probabilities and rewards is shown below.

s	a	s'	T(s,a,s')	R(s,a,s')
0	WALK	1	1.0	0
1	WALK	2	1.0	0
2	WALK	3	1.0	10
0	RUN	1	0.5	-1
0	RUN	2	0.5	-1
1	RUN	2	0.5	-1
1	RUN	3	0.5	+9
2	RUN	3	1.0	+9

(6.1) **(3 points)** Suppose we initialize $Q_0(s,a) = 0$ for all s ϵ {0, 1, 2} and all a $\in$ {WALK, RUN}. We assume that the values in state $s = 3$ are always zero for any action. Evaluate the Q-values $Q_1(s,a)$ after exactly one Q-value iteration.

a	s=0	s=1	s=2
WALK	0	0	10
RUN	-1	4	9

(6.2) **(3 points)** What is the ideal policy derived from $Q_1(s,a)$?

$$\begin{aligned} \pi_1^*(s=0) &= \text{walk} \\ \pi_1^*(s=1) &= \text{run} \\ \pi_1^*(s=2) &= \text{walk} \end{aligned} \tag{9}$$

(6.3) **(3 points)** What are the values $V_1(s)$ using the values of $Q_1(s,a)$ calculated above?

s=0	s=1	s=2
0	4	10

6.1) From cheatsheet:

$$Q_{i+1}(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \lambda \max_{a'}\left(\hat{Q}_i(s',a')\right)\right]$$

$$Q_1(0,RUN) = T(0,RUN,1)\left[R(0,Run,1) + .5\cancel{(0)}^{0}\right]$$
$$+ T(0,RUN,2)\left[R(0,Run,2) + .5\cancel{(0)}^{0}\right] =$$

$$= (.5)(-1) + (.5)(-1) = -1$$

$Q_1(1,RUN) = (.5)(-1) + (.5)(9) = 4$

$Q_1(2,RUN) = 9$

$Q_1(0,walk) = 0$

$Q_1(1,walk) = 0$

$Q_1(2,walk) = 10$

6.2) Maximize Q-value at each state

6.4)

$$Q_2(0, \text{Run}) = .5\left[-1 + \overbrace{\gamma \max_{a'} Q_1(1, a')}^{4}\right]$$
$$+ .5\left[-1 + \underbrace{\gamma \max_{a'} Q_1(2, a')}_{10}\right]$$
$$= -.5 + 2\gamma + -.5 + 5\gamma$$
$$= 7\gamma - 1$$

$$Q_2(0, \text{walk}) = 1[0 + \gamma 4] = 4\gamma$$

If $\gamma = 0 \Rightarrow$ walk

$Q_2(0, \text{Run}) \geq Q_2(0, \text{walk})$

$7\gamma - 1 \geq 4\gamma$

$3\gamma \geq 1$

$\gamma \geq 1/3$

If $\gamma \geq 1/3 \Rightarrow$ RUN

(6.4) **(3 points)** Consider now iterating Q-values one more time to obtain $Q_2(s, a)$. We are only interested in here what happens at $s = 0$. For what range of values of the discount factor $0 \leq \gamma \leq 1$ would the action derived from $Q_2(0, a)$ suggest that we RUN?

$\gamma \geq 1/3$

Problem 7

(7.1) **(5 points)** Choose T/F for each of the following statements.

( ) While neural networks are powerful, classifiers based on SIFT features still perform better at object recognition tasks

( ) The tasks of tracking people and objects are solved quite well for short time spans but remain challenging for longer trajectories

( ) Bayes filtering is unsuitable for robot localization.

( ) Aerial robots typically make use of laser range finders (rather than cameras) to carve out open space

( ) SLAM stands for simultaneous localization and mapping and is the process by which a robot explores to create a map of its environment

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 Introduction to Machine Learning

Final exam (May 19, 2015)

Your name & ID: _____

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Prob 1	Prob 2	Prob 3	Prob 4	Prob 5	Prob 6	Total
23	16	18	16	15	6	**94**

Problem 1

(1.1) **(5 points)** VC-dimension is a measure of complexity of a set of classifiers such as the set of linear classifiers. Let $\mathcal{H}$ denote the set of classifiers in question and $d_{VC}(\mathcal{H})$ the corresponding VC-dimension. Select all the true statements.

a) (✓) If $\mathcal{H}$ has only one classifier, then $d_{VC}(\mathcal{H}) = 0$
b) (✓) If $\mathcal{H} = \{h_1, \ldots, h_K\}$ (finite), then necessarily $d_{VC}(\mathcal{H}) \leq \log_2 K$
c) ( ) If $d_{VC}(\mathcal{H}) > n$, then any n labeled points can be correctly classified by some $h \in \mathcal{H}$.
d) ( ) No set of $n > d_{VC}(\mathcal{H})$ labeled points can be correctly classified by $h \in \mathcal{H}$.
e) (✓) There exists $d_{VC}(\mathcal{H})$ points that can be always correctly classified by some $h \in \mathcal{H}$

(1.2) **(4 points)** Consider a simple set of classifiers in two dimensions where, for each classifier, the positive set is always between two vertical lines or between two horizontal lines. In other words, any $h \in \mathcal{H}$, parameterized by $\theta = \{a, b, i\}$, can be written as

$$h_\theta(x) = \begin{cases} +1 \text{ if } a \leq x_i \leq b, \\ -1 \text{ otherwise} \end{cases}$$

Please write $h_\theta(x)$ where $\theta = \{a, b, i\}$ as an ensemble of decision stumps.

$$h_\theta(x) = \text{sign}\left[\text{sign}(x_i - a) + \text{sign}(b - x_i) - 1\right]$$

$\text{sign}(0) = 1$

1.1) VC-Dimension: Largest set of points that the classifier can shatter

arranged in any chosen way (for all trials)

classify all correctly with all possible labels

(1)
- 2^d different classifiers $\in \mathcal{H} \Rightarrow$ VC-Dimension $\leq d$
- linear classifier w/ dimension $d \Rightarrow$ VC-Dimension $= d+1$

a) By (1) $2^0 = 1$ classifier $\Rightarrow$ VC-Dimension $\leq 0 = 0$ TRUE

b) Rewriting (1) TRUE

c) No. VC-D shows MAX in <u>best case</u> with chosen point positions

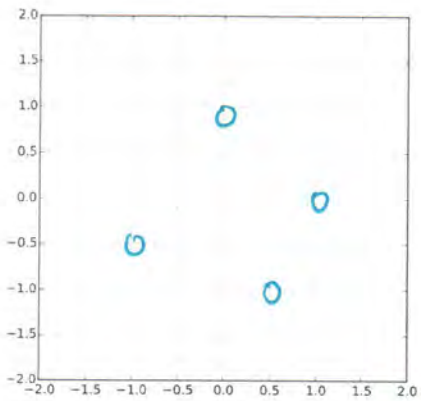
d) Points could be arranged so that they can 

e) Definition of VC-D

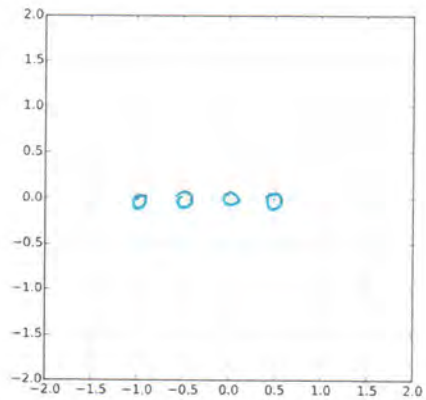
1.2) Ensamble: combine different kinds of predictors

Decision stump: one level decision tree 

(1.3) **(4 points)** In the figure below, draw a set of 4 points that can be shattered (classified in all possible ways) by this set of classifiers (left) and a set of 4 points that cannot (right).



can be shattered $\Rightarrow$ VC-D = 4 ($\mathcal{H}$)



cannot be shattered

(1.4) **(4 points)** So, we got n labeled training examples in $\mathcal{R}^2$ and found the best classifier in the above set $\mathcal{H}$. The resulting training error was zero. We also fit a linear classifier to the same training set and also got zero training error. Which classifier should we prefer? Briefly explain why.

VC-D(linear) < VC($\mathcal{H}$) $\Rightarrow$ linear is prefered

3 < 4

Higher VC-D tends to overfit the training data

(1.5) **(3 points)** In practice, we typically resort to cross-validation as a proxy to test error in order to select among different sets of classifiers. Leave-one-out cross-validation takes out each example in turn as a held-out test example, and averages the error on these held-out examples when the classifier is trained on the remaining points. Our training set is given in the figure (CV) below. Suppose we use a kernel perceptron algorithm with a radial basis kernel for this problem. What is the resulting leave-one-out cross-validation error? (1.0)

FAIL

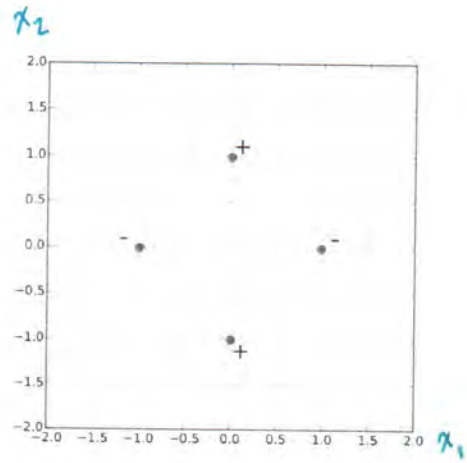
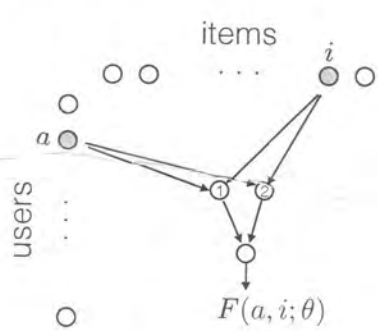


Figure (CV): labeled training set

(1.6) **(3 points)** Specify a feature vector $\phi(x)$ for a linear classifier that would attain the lowest leave-one-out cross-validation error on the training points in figure (CV).

$\phi(x) = |x_1|$ $\Rightarrow$ + + =

Problem 2 Suppose we have a recommender problem with n users $a \in \{1, \ldots, n\}$ and m items $i \in \{1, \ldots, m\}$. For simplicity, we will treat the target rating values as class labels, i.e., using $\{-1, 1\}$ ratings (dislikes,likes). Each user is likely to provide feedback for only a small subset of possible items and thus we must constrain the models so as not to overfit. Our goal here is to understand how a simple neural network model applies to this problem, and what its constraints are. To this end, we introduce an input unit corresponding to each user and each item. In other words, there are $n + m$ input units. When querying about a selected entry, (a, i), only the ath user input unit and i^{th} item input unit are active (set to 1), the rest are equal to zero and will not affect the predictions. Put another way, only the outgoing weights from these two units matter for predicting the value (class label) for entry (a, i). Figure below provides a schematic representation of the model.



2.1)

(a,1)

$$z_1 = \overset{v_{a1}}{1} + \overset{v_{11}}{-1} = 0 \qquad f(z_1) = 0$$
$$z_2 = 1 + 1 = 2 \qquad f(z_2) = 2 \qquad \Rightarrow (0,2)$$

(b,1)

$$z_1 = \overset{v_b}{1} + \overset{v_1}{-1} = 0$$
$$z_2 = -1 + 1 = 0 \qquad \Rightarrow (0,0)$$

(a,2)

$$z_1 = 1 + 1 = 2$$
$$z_2 = 1 + 0 = 1 \qquad \Rightarrow (2,1)$$

(b,2)

$$z_1 = 1 + 1 = 2$$
$$z_2 = -1 + 0 = -1 \qquad \Rightarrow (2,0)$$

User a has two outgoing weights, U_{a1} and U_{a2}, and item i has two outgoing weights, V_{i1} and V_{i2}. These weights are fed as inputs to the two hidden units in the model. The hidden units evaluate

$$z_1 = U_{a1} + V_{i1}, \quad f(z_1) = \max\{0, z_1\}$$
$$z_2 = U_{a2} + V_{i2}, \quad f(z_2) = \max\{0, z_2\}$$

Thus, for (a, i) entry, our network outputs

$$F(a, i; \theta) = W_1 f(z_1) + W_2 f(z_2) + W_0$$

where θ denotes all the weights U, V, and W. In a vector form, each user a has a two-dimensional vector of outgoing weights $\vec{u}_a = [U_{a1}, U_{a2}]^T$ and each item i has $\vec{v}_i = [V_{i1}, V_{i2}]^T$. The input received by the hidden units, if represented as a vector, is then $\vec{z} = [z_1, z_2]^T = \vec{u}_a + \vec{v}_i$.

Consider a simple version of the problem where we have only two users, $\{a, b\}$, and two items $\{1, 2\}$. So the recommendation problem can be represented as a 2x2 matrix. We will initialize the first layer weights as shown in Figure (NN) below.

(2.1) **(4 points)** Using the initial input-to-hidden layer weights, each of the four user-item pairs in the 2x2 matrix are mapped to a corresponding feature representation $[f(z_1), f(z_2)]^T$ (hidden unit activations). Please mark the points on the right with the correct pair, e.g., (a,1), that it corresponds to.

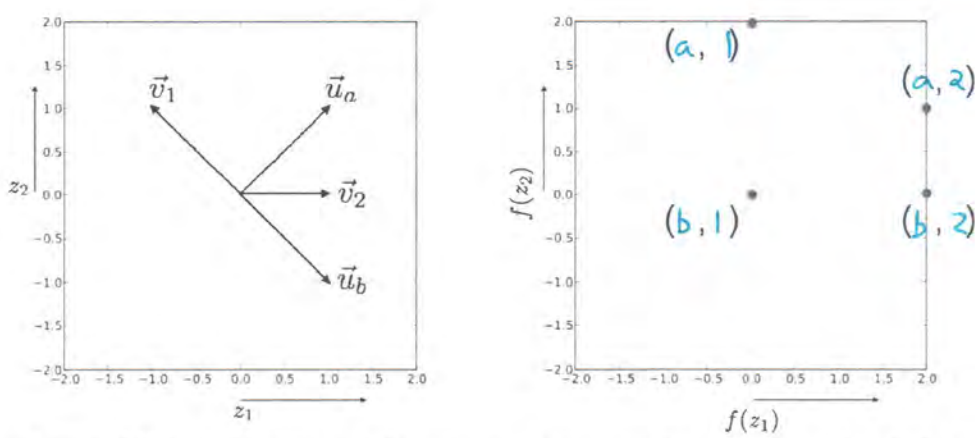


Figure (NN): Outgoing weight vectors from user/item input units (left); hidden layer activations (right)

(2.2) **(4 points)** Suppose we keep the input to the hidden layer weights (U's and V's) at their initial values shown in Figure (NN), and only estimate the weights W corresponding to the output layer. Different choices of the output layer weights will result in different predicted 2x2 matrices of $\{-1, 1\}$ labels. Which (if any) of the following matrices the neural network **cannot** reproduce with any choice of the output layer weights W_1, W_2, and W_0?

$f(z_2) + (-1)f(z_1)$

	1	2
a	+1	+1
b	+1	-1

( ) ✓

	1	2
a	-1	+1
b	+1	-1

(✓)

Learning a new representation for examples (hidden layer activations) is always harder than learning the linear classifier operating on that representation. In neural networks, the representation is learned together with the end classifier using stochastic gradient descent. We initialize the output layer weights as $W_1 = W_2 = 1$ and $W_0 = -1$.

(2.3) **(2 points)** Assume that all the weights are initialized as provided above. What is the class label (+1/-1) that the network would predict in response to $(b, 2)$ (user b, item 2)? (+1) $F = f(z_2) + f(z_1) - 1 = 0 + 2 - 1 = +1$

(2.4) **(6 points)** Assume that we observe the opposite label from your answer to the previous question. In other words, there is a training signal at the network output. All the weights are initialized as before. Please mark (check the boxes) of all the weights in Figure (SGD) that would change (have non-zero update) based on a single stochastic gradient descent step in response to $(b, 2)$ with our specific weight initialization and the target label. Note that the input units a, b and $1, 2$ are activated with 0's and 1's as shown inside the circles. We are not asking about whether W_0 would change.

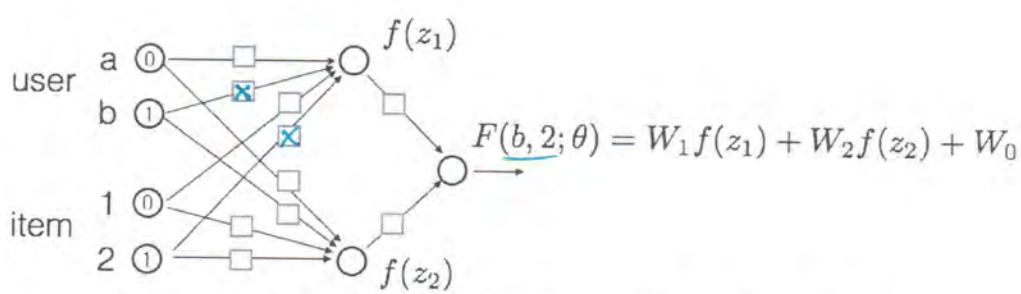


Figure (SGD): Neural network for stochastic gradient descent.

Problem 3 Consider initializing a two component Gaussian mixture model as shown below in the figure. The variances of the two Gaussians are equal $\sigma_1^2 = \sigma_2^2 = 0.5^2$ and they have the same prior probabilities (mixing proportions) $p_1 = p_2 = 0.5$. The two means $\mu^{(1)}$ and $\mu^{(2)}$ are exactly at the grid points shown in the figure (EM).

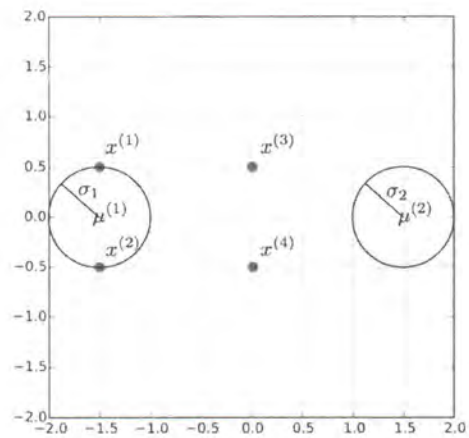


Figure (EM): Initial two component Gaussian mixture model.

(3.1) **(4 points)** Please select the right intervals for the soft assignments of points to mixture components in the first E-step of the EM-algorithm. Here $p(j|i)$ is the posterior probability that component j is assigned to point i. Each point should be marked to exactly one interval

	$p(1\|i) < 0.5$	$p(1\|i) = 0.5$	$p(1\|i) > 0.5$
$x^{(1)}$			✓
$x^{(2)}$			✓
$x^{(3)}$		✓	
$x^{(4)}$		✓	

(3.2) **(3 points)** Let $\hat{\mu}^{(2)}$ be the mean for the 2nd component after the first M-step. Which of the following holds for the horizontal component $\hat{\mu}_1^{(2)}$ of this mean?

(✓) $\hat{\mu}_1^{(2)} < 0$, ( ) $\hat{\mu}_1^{(2)} = 0$, ( ) $\hat{\mu}_1^{(2)} > 0$

Pulled left by all points

(3.3) **(3 points)** Which of the following holds for the variances $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ after the first M-step?

(✓) $\hat{\sigma}_1^2 > 0.5^2$, ( ) $\hat{\sigma}_2^2 > 0.5^2$, (✓) $\hat{\sigma}_1^2 > \hat{\sigma}_2^2$

(3.4) **(4 points)** Is $\hat{p}_1 > 2\hat{p}_2$ after the first M-step? Please answer **Y/N ( )**.
For ballpark estimation, $\exp(-1) \approx 0.37, \exp(-3) \approx 0.05, \exp(-6) \approx 0.0025$.

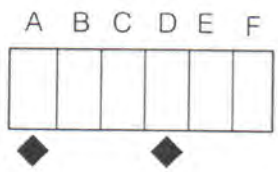
(3.5) **(4 points)** If we continue to iterate the E- and M- steps, what are the values of variances and the mixing proportions that the algorithm converges to with the above initialization?

$$\sigma^{*2}_1 = \_\_\_\_\_, \quad \sigma^{*2}_2 = \_\_\_\_\_$$

$$p^*_1 = \_\_\_\_\_, \quad p^*_2 = \_\_\_\_\_$$

HMM

Problem 4 We will consider here robot localization from measurements. We assume that the robot operates in a simple 1-dimensional grid shown below. Unfortunately, our fancy on-board localization software failed and the robot is left with only measuring and transmitting the temperature of its immediate surroundings, i.e., the temperature at the grid point it is at. It can transmit only a binary value $X =$"cold" or $X =$"hot". Two locations in the grid – A and D – are known to have volcanoes and are therefore "hot". The robot's temperature gauge is 90% accurate whether it is in "hot" or "cold" locations. For example, it would correctly transmit "hot" in A with probability 0.9.



We know that the robot starts in positions A or B on day 1 so $Y_1 = A$ or $Y_1 = B$ with equal probability. The robot always tries to move to the right (towards F) over night. If it is currently in locations A or B, it will succeed in moving to the right overnight. In all other locations, it will either succeed in moving one step to the right overnight (with probability 0.5) or else it would remain in the same location (with probability 0.5). Let the robot's position on day t be $Y_t \in \{A, B, C, D, E, F\}$. We will model the robot's position with an HMM where the state is Y_t and the corresponding observation is the robot's transmission X_t.

4.3) $\underline{A \to B \to C}$

$$.5 \cdot .9 \cdot 1 \cdot .9 \cdot 1 \cdot .9$$

Start	Hot	A→B	Cold	B-C	Cold
A	A		B		C

$\underline{B \to C \to C}$

$$(.5)(.1) \cdot (1 \cdot .9) \cdot (.5)(.9)$$

Start	Hot	B→C	Cold	C→C	cold
B	B		C		C

$\underline{B \to C \to D}$ $(.5)(.1) \cdot (1)(.9) \cdot (.5)(.1)$

4.4) We know Y_3 can be either C or D

$$P(Y_3 = C \mid hcc) = \frac{P(A \to B \to C) + P(B \to C \to C)}{\text{normalize}}$$

$$P(Y_3 = D \mid hcc) = \frac{P(B \to C \to D)}{\text{normalize}}$$

(4.1) **(4 points)** Specify the initial state, state transition, and the emission probabilities for this HMM.

Y_1	
A	.5
B	.5
C	0
D	0
E	0
F	0

Y_{t-1} \ Y_t	A	B	C	D	E	F
A		1				
B			1			
C			.5	.5		
D				.5	.5	
E					.5	.5
F						1

Y_t \ X_t	"hot"	"cold"
A	.9	.1
B	.1	.9
C	.1	.9
D	.9	.1
E	.1	.9
F	.1	.9

(4.2) **(4 points)** What are the possible (non-zero probability) sequences of locations (Y_1, Y_2, Y_3) that the robot could have followed up to and including day 3 if it has transmitted $X_1 =$"hot", $X_2 =$"cold", and $X_3 =$"cold"?

A→B→C

B→C→C

B→C→D

(4.3) **(4 points)** Which sequence of locations (Y_1, Y_2, Y_3) is most likely to occur together with $(X_1 =$ "hot", $X_2 =$ "cold", $X_3 =$ "cold"), and what is the corresponding joint probability $P(Y_1, Y_2, Y_3, X_1, X_2, X_3)$?

A B C

$P = .5 \cdot .9 \cdot 1 \cdot .9 \cdot 1 \cdot .9$

(4.4) **(4 points)** What is $P(Y_3 | X_1 =$ "hot", $X_2 =$ "cold", $X_3 =$ "cold"), i.e., the posterior probability distribution over the robot's possible locations on day 3, given the observations? Please check exactly one box on each row.

$P(Y_3 | X_1 =$ "hot", $X_2 =$ "cold", $X_3 =$ "cold")

Y_3	0.0	< 0.1	≈ 0.5	> 0.9
A	✓			
B	✓			
C				✓
D		✓		
E	✓			
F	✓			

Problem 5 Consider a robot that can either stand still and CHARGE (using its solar panels) or it can scurry around and EXPLORE. The robot's state (as we measure it) represents only how charged its battery is and can be EMPTY, LOW or HIGH. The robot is very eager to explore and this is how the rewards are set. The MDP transition probabilities and rewards are specified as shown below.

s	a	s'	T(s,a,s')	R(s,a)
HIGH	EXPLORE	LOW	1.0	+2
LOW	EXPLORE	EMPTY	1.0	+2
EMPTY	EXPLORE	EMPTY	1.0	-10
HIGH	CHARGE	HIGH	1.0	0
LOW	CHARGE	HIGH	1.0	-1
EMPTY	CHARGE	HIGH	1.0	-10

Note that the reward only depends on the robot's current state and action, not the state that it transitions to.

(5.1) **(3 points)** Based on the transitions and rewards (without further calculation), what is the optimal policy for this robot if we set the discount factor $\gamma = 0$?

$\gamma = 0$ = maximize immediate reward

$\pi_0^*(HIGH)$ = EXPLORE
$\pi_0^*(LOW)$ = EXPLORE
$\pi_0^*(EMPTY)$ = EXPLORE OR CHARGE

(5.2) **(4 points)** Could changing the discount factor γ change the optimal action to take in any state? (check all that apply)

( ) when s =HIGH? ×

(✓) when s =LOW?

(✓) when s =EMPTY?

(5.3) **(4 points)** Let's see how the robot values its states, and recovers those values through value iteration, when the discount factor is set to $\gamma = 0.5$. We start with all zero values as shown in the first value column. Please fill out the table

s	$V_0(s)$	$V_1(s)$	$V_2(s)$
EMPTY	0	-10	-9
LOW	0	2	0
HIGH	0	2	3

5.3) From cheatsheet:

$$V_{i+1}(s) = \max_a \left[\sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right] \right]$$

$$V_1(\overset{\text{Empty}}{E}) = \max \begin{cases} \text{charge: } T(E, \text{charge}, \text{high}) \left[R(E,C,H) + \gamma V_0(\text{high}) \right] \\ \qquad = 1\left[-10 + .5\cancel{(0)}^{0}\right] = -10 \\ \text{explore: } T(E, \text{explore}, E) \left[R(E,E,E) + \gamma \cancel{V_0(\text{empty})}^{0} \right] \\ \qquad = 1(-10) = -10 \qquad = -10 \end{cases}$$

$\vdots$

$$V_2(E) = \max \begin{cases} \text{charge: } T(E, \text{charge}, \text{high}) \left[R(E,C,H) + (.5) V_1(\text{high}) \right] \\ \qquad = 1\left[-10 + (.5)(2)\right] = -9 \\ \text{explore: } 1\left[-10 + (.5)(-10)\right] = -15 \end{cases} \qquad = -9$$

$$V_2(L) = \max \begin{cases} \text{charge: } 1 \cdot \left[-1 + .5 \cdot 2 \right] = 0 \\ \text{explore: } 1 \cdot \left[+2 + (.5)(-10) = -3 \right. \end{cases} \qquad = 0$$

$$V_2(H) = \max \begin{cases} \text{charge: } 1 \cdot \left[0 + \gamma V_0(\text{high}) \right] = 1 \\ \text{explore: } 1 \cdot \left[2 + \gamma V_0(\text{low}) \right] = 3 \end{cases} \qquad = 3$$

(5.4) **(4 points)** If the robot uses values $V_2(s)$ as the true (converged) values, which action would it take in state $s = LOW$? (Show your calculation)

Charge

(From 5.3)

Problem 6

(6.1) **(4 points)** Two of the guest lectures emphasized the role of "transfer learning" in their applied contexts, especially medicine. Briefly describe what transfer learning is.

Using data from one domain to help another

(6.2) **(2 points)** In using medical data for prediction, one may often have to face "small data" rather than "big data" problems. One of the guest lectures emphasized ways to deal with issues arising from applying machine learning methods in the "small data" regime. These include (select all that apply)

- ✓(X) Dimensionality reduction such as PCA
- ✓(X) Shrinkage (using estimates tied to broader categories to inform more specific ones)
- ( ) Expanding feature vectors to include more potentially useful features
- ( ) Making high dimensional feature vectors sparse

END OF EXAM

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 Introduction to Machine Learning

Final exam (May 16, 2014)

Your name & ID: _____

- This is a closed book exam
- You do not need nor are permitted to use calculators
- The value of each question – number of points awarded for full credit – is shown in parenthesis
- The problems are not necessarily in any order of difficulty. We recommend that you read through all the problems first, then do the problems in whatever order suits you best.
- Record all your answers in the places provided

Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	**Total**

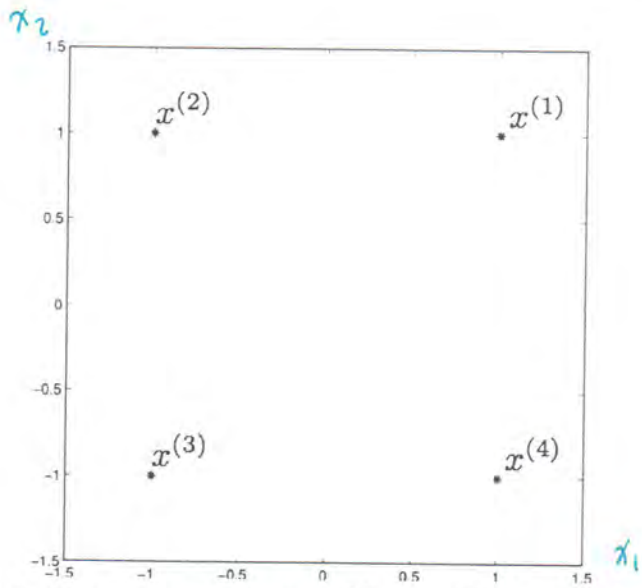


Figure 1. Four training points, $x^{(1)}, \ldots, x^{(4)}$, without labels.

Problem 1 We will consider here four two dimensional training examples, $x^{(1)}, \ldots, x^{(4)}$, illustrated in Figure 1. These points are labeled $y^{(1)}, \ldots, y^{(4)}$. We will explore different ways of labeling the points below as well as their effect on the algorithm. The simple perceptron algorithm with offset is given by

Initialize: $\theta = 0$ (vector), $\theta_0 = 0$

Cycle through $i = 1, \ldots, n$

If $\left(y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \leq 0 \right)$ then

update $\theta \leftarrow \theta + y^{(i)} x^{(i)}$ and $\theta_0 \leftarrow \theta_0 + y^{(i)}$

(a) **(4 points)** We would like to turn this simple perceptron algorithm into a kernel perceptron algorithm. This can be done by mapping each example x into a feature vector $\phi(x)$ and reformulating the algorithm such that it only uses the kernel function $K(x, x') = \phi(x) \cdot \phi(x')$. We will do so below. If we use the kernel

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

as the kernel function, what is the feature vector $\phi(x)$?

1.a) $K(x, x') = (1 + x \cdot x')^2 = \phi(x) \cdot \phi(x')$

$x = [x_1, x_2]^T$

$\Rightarrow (1 + x \cdot x')^2 = \left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right)^2 = \left(1 + x_1 x_1' + x_2 x_2'\right)^2$

$= 1 + 2(x_1 x_1' + x_2 x_2') + (x_1 x_1')^2 + 2(x_1 x_1')(x_2 x_2') + (x_2 x_2')^2$

Factor

$= [1, \sqrt{2} x_1, \sqrt{2} x_2, x_1^2, \sqrt{2} x_1 x_2, x_2^2] \cdot [\ \dots\ x_1' \quad x_2']$

$\Rightarrow \phi(x) =$

$\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2]$

(b) **(6 points)** We are now ready to formulate the kernel perceptron algorithm. Once trained, the algorithm would predict label for x according to

$$\text{sign}\Big(\sum_{j=1}^{n} \alpha_j y^{(j)} K(x^{(j)}, x) + \theta_0\Big) \tag{2}$$

where $K(x, x')$ is the kernel function and $n = 4$ is the number of training examples. Please fill in the algorithmic steps for the kernel perceptron algorithm <u>with offset</u>.

Initialize: _____

Cycle through $i = 1, \ldots, n$

If _____ then

update _____

(c) **(4 points)** Suppose we run the algorithm using the radial basis kernel function. In other words, we would use

$$K(x, x') = \exp\big(-\frac{1}{2}\|x - x'\|^2\big) \tag{3}$$

Consider now the training examples in Figure 1. Does it matter how these four points are labeled in terms of whether or not the algorithm converges? Briefly justify your answer.

no

(d) **(6 points)** Provide a labeling of the four training points that satisfies two criteria: 1) the kernel perceptron algorithm with the radial basis kernel converges after only one update, and 2) the simple perceptron algorithm converges but requires multiple updates.

$$y^{(1)} = (\quad),\ y^{(2)} = (\quad),\ y^{(3)} = (\quad),\ y^{(4)} = (\quad), \tag{4}$$

Problem 2 Here we consider solving a classification problem using Support Vector Machines (SVMs). Specifically, given two dimensional training examples $x^{(1)}, \ldots, x^{(n)}$ and labels $y^{(1)}, \ldots, y^{(n)}$, we find $\hat{\theta}$ and $\hat{\theta}_0$ by solving

$$\min_{\theta, \theta_0, \xi} \quad \frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{n} \xi_i \tag{5}$$

$$\text{subject to} \quad y^{(i)}(\theta \cdot x^{(i)} + \theta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{6}$$

Let $\hat{\theta} = [1, 2]^T$, $\hat{\theta}_0 = -1$ be the solution we obtained with a specific value of C.

(a) **(8 points)** Which of the following training examples are support vectors. Check all that apply.

( ) $x^{(1)} = [2, 1]^T$, $y^{(1)} = 1$

( ) $x^{(2)} = [-1, 1]^T$, $y^{(2)} = -1$

( ) $x^{(3)} = [0, 1]^T$, $y^{(3)} = 1$

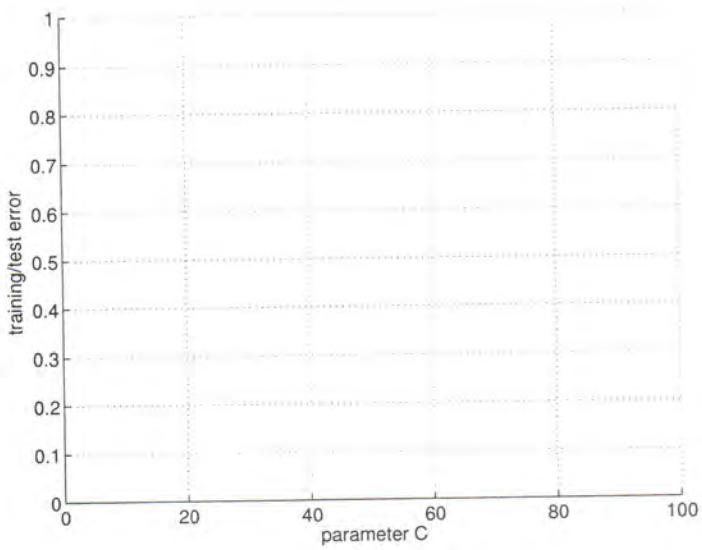
( ) $x^{(4)} = [1, -1]^T$, $y^{(4)} = 1$

(b) **(2 points)** What is the orthogonal distance from the decision boundary to either of the margin boundaries?

(c) **(6 points)** We can try to understand how training and test errors behave as a function of parameter C in the SVM optimization problem. We are asking you to draw a plausible pair of training and test error curves as a function of C. In the plot below, clearly mark

1) A plausible training error curve as a function of C
2) A plausible test error curve as a function of C
3) Regions of C values where SVM under-fits and where it over-fits.

Make sure that your error curves demonstrate both under-fitting and over-fitting.



Problem 3 The EM algorithm is very useful general method for estimating probability models that involve latent variables, i.e., variables that are not directly observed in the data. We will consider here a language modeling problem, how to predict the next word based on the current one. Since we assume we will have limited data for estimating the model, we expect that it would be better to assign the current word first into a cluster (denoted by $z = 1, \ldots, k$) and then predict the next word from the cluster. In other words, given w_1 (current word), we will predict w_2 (next word), according to

$$P(w_2|w_1) = \sum_{z=1}^{k} P(w_2|z)P(z|w_1) = \sum_{z=1}^{k} \beta_{w_2|z}\theta_{z|w_1} \tag{7}$$

where $P(w_2|z) = \beta_{w_2|z}$ and $\sum_{w\in\mathcal{W}} \beta_{w|z} = 1$ for any $z = 1, \ldots, k$. Similarly, $\sum_{z=1}^{k} \theta_{z|w} = 1$ for any current word $w \in \mathcal{W}$. Here $\mathcal{W}$ denotes our vocabulary. Note that our model is parameterized by two probability tables, $\beta_{w|z}$ and $\theta_{z|w}$.

(a) **(4 points)** Suppose we observe $\hat{w}_1$ (current) and $\hat{w}_2$ (next word). We would like to find the posterior probability of cluster $z = 1$ given this information. Which of the following expressions corresponds to this posterior.

$$(\quad) \qquad \theta_{z=1|\hat{w}_1} / \Big(\sum_{z'=1}^{k} \theta_{z'|\hat{w}_1} \Big) \tag{8}$$

$$(\quad) \qquad \beta_{\hat{w}_2|z=1}\theta_{z=1|\hat{w}_1} / \Big(\sum_{z'=1}^{k} \beta_{\hat{w}_2|z'}\theta_{z'|\hat{w}_1} \Big) \tag{9}$$

$$(\quad) \qquad \beta_{\hat{w}_2|z=1}\theta_{z=1|\hat{w}_1} / \Big(\sum_{w\in\mathcal{W}} \beta_{\hat{w}_2|z=1}\theta_{z=1|w} \Big) \tag{10}$$

(b) **(4 points)** The EM algorithm tries to maximize the log-probability of generating the data. In our case, based on a sequence of words $\hat{w}_1, \ldots, \hat{w}_T$ (document), we aim to predict $\hat{w}_2$ given $\hat{w}_1$, $\hat{w}_3$ given $\hat{w}_2$, and so on. Write down an expression for the log-probability of words $\hat{w}_2, \ldots, \hat{w}_T$ given $\hat{w}_1$ in terms of the parameters $\beta_{w|z}$ and $\theta_{z|w}$.

(c) **(4 points)** In the EM algorithm, each M-step updates parameters $\beta_{w|z}$ and $\theta_{z|w}$ based on counts evaluated in the E-step. In order to estimate $\beta_{w|z}$ on the basis of a single document $\hat{w}_1, \ldots, \hat{w}_T$, which counts do we need? Check only one.

( ) $\hat{n}(w, w') = \sum_{t=1}^{T-1} [\![\hat{w}_t = w]\!] [\![\hat{w}_{t+1} = w']\!]$

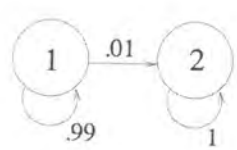
( ) $\hat{n}(z, w') = \sum_{t=1}^{T-1} P(z_t = z | \hat{w}_t) [\![\hat{w}_t = w']\!]$

( ) $\hat{n}(z, w') = \sum_{t=1}^{T-1} P(z_t = z | \hat{w}_t) [\![\hat{w}_{t+1} = w']\!]$

( ) $\hat{n}(z, w') = \sum_{t=1}^{T-1} P(z_t = z | \hat{w}_t, \hat{w}_{t+1}) [\![\hat{w}_t = w']\!]$

( ) $\hat{n}(z, w') = \sum_{t=1}^{T-1} P(z_t = z | \hat{w}_t, \hat{w}_{t+1}) [\![\hat{w}_{t+1} = w']\!]$

Problem 4 We will use here Hidden Markov Models for biological sequence modeling. There are two hidden states in the HMM and four output symbols $\{A, G, T, C\}$ corresponding to DNA bases. The transition probabilities between the states are shown in the diagram below



So, for example, $P(s_{t+1} = 1 | s_t = 1) = 0.99$. The initial state is chosen at random with equal probability for $s_1 = 1$ and $s_1 = 2$. The output symbols are generated from each state according to $P(x|s)$ given by the table

	$x = A$	$x = G$	$x = T$	$x = C$
$s = 1$	0.0	0.19	0.8	0.01
$s = 2$	0.1	0.0	0.7	0.2

(11)

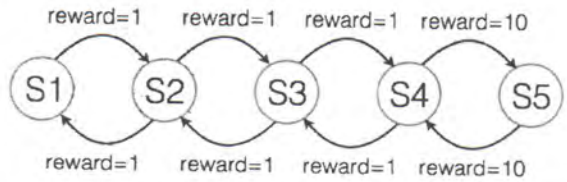
For example, $P(x = T|s = 2) = 0.7$. Note that each row must sum to one.

(a) **(4 points)** Please provide an output sequence of length two (first two output symbols) that cannot be generated from this model.

(b) **(4 points)** Suppose we generate an output sequence of length 1,000,000 from this model. From which state is the last symbol in this sequence likely to have been generated from?

(c) **(4 points)** If we observe $x_1 = T$ and $x_2 = T$ (first two symbols). What is the most likely underlying hidden state sequence, i.e., s_1 and s_2, that generated these observations?

Problem 5 Consider a reinforcement learning problem specified by the following Markov Decision Process (MDP).



We have five states representing steps along one direction. Call these states $S1$, $S2$, $S3$, $S4$, and $S5$. From each state, except the end states, we can move either left or right. The available actions in state $S1$ is just to move right while the action available in $S5$ is to move left. We can move left or right in each intermediate state. The reward for taking any action is 1 except when moving right from $S4$ or left from $S5$ which provide reward 10. Assume a discount factor $\gamma = 0.5$. Note that $\sum_{i=1}^{\infty} 0.5^i = 1$.

(a) **(5 points)** What is the optimal policy for this MDP? Specify action (L/R) to take in each state.

$$S1: (\quad) \quad S2: (\quad) \quad S3: (\quad) \quad S4: (\quad) \quad S5: (\quad) \tag{12}$$

(b) Suppose we apply value iteration on this MDP. What is the value of state S3 after

(2 points) one value iteration

(2 points) two value iterations

(3 points) ∞ number of value iterations

6. b) Want to show that: $P(x_1, x_3) = P(x_1)P(x_3)$

1) Marginalize out all other variables: x_2, x_4

$$P(x_1, x_3) = \sum_{x_2, x_4} P(x_1) P(x_2 | x_1) P(x_3) P(x_4 | x_2, x_3)$$

$$= P(x_1)P(x_3) \sum_{x_2, x_4} \underbrace{\cancelto{1}{P(x_2 | x_1)}}_{\text{all } x_2\text{'s}} \underbrace{\cancelto{1}{P(x_4 | x_2, x_3)}}_{\text{all } x_4\text{'s}}$$

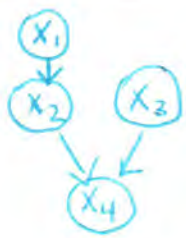
$$= P(x_1)P(x_3) \quad \square$$

Problem 6 Consider a modeling problem involving four tertiary variables, $x_1, \ldots, x_4$, each taking values in $\{-1, 0, 1\}$. In our first Bayesian network model, the joint distribution over these four variables factors according to

$$\text{BN 1}: \quad P(x_1, x_2, x_3, x_4) = P_1(x_1)P_2(x_2|x_1)P_3(x_3)P_4(x_4|x_2, x_3) \tag{13}$$

Let's assume that we can set the parameters in the conditional tables as we wish, i.e., that there are no constraints other than that the distribution must factor as shown above. This is known as a fully parameterized model.

(a) **(4 points)** Draw the Bayesian network graph corresponding to this model



(b) **(6 points)** Show based on the joint distribution in Eq(13) that x_1 is marginally independent of x_3. ie, $P(x_1, x_3) = P(x_1)P(x_3)$

(c) **(4 points)** Let's introduce an alternative model, also fully parameterized, given by

$$\text{BN 2}: \quad P(x_1, x_2, x_3, x_4) = P_1(x_1|x_2)P_2(x_2|x_3)P_3(x_3)P_4(x_4|x_2) \tag{14}$$

Suppose we estimate the parameters of these models, BN 1 and BN 2, based on the same training data and they assign the same log-likelihood to the data. Which model should we prefer based on the data? Briefly justify your answer.

Perform the same, but BN 2 is more constrained. ⇐ Better