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**

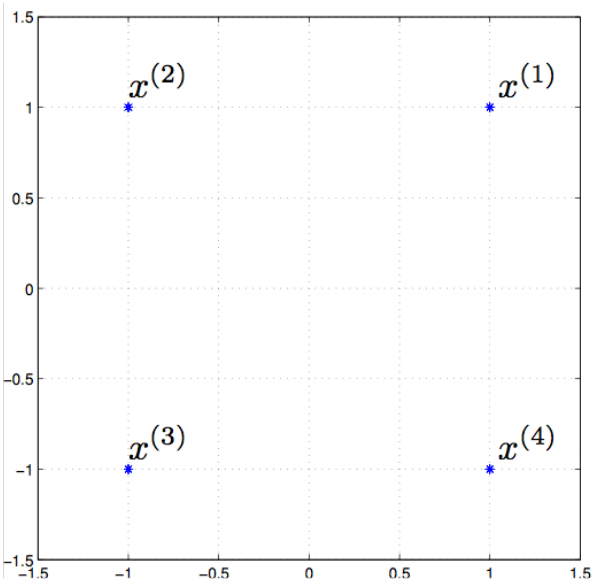


Figure 1. Four training points, $x^{(1)}, \ldots, x^{(4)}$, without labels.

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 $\phi(x)$?

$x = [x_1, x_2]^T$ so that

$$\begin{aligned}(1 + x \cdot x')^2 &= (1 + x_1 x_1' + x_2 x_2')^2 \\ &= 1 + 2(x_1 x_1' + x_2 x_2') + (x_1 x_1')^2 + 2(x_1 x_1')(x_2 x_2') + (x_2 x_2')^2 \\ &= [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1 x_2, x_2^2] \cdot [1, \sqrt{2}x_1', \sqrt{2}x_2', x_1'^2, \sqrt{2}x_1' x_2', x_2'^2]\end{aligned}$$

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

(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 with offset.

Initialize: $\alpha_j = 0, \quad j = 1, \ldots, n, \quad \theta_0 = 0$

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

If $y^{(i)}(\sum_{j=1}^{n} \alpha_j y^{(j)} K(x^{(j)}, x^{(i)}) + \theta_0) \leq 0$ then

update $\alpha_i \leftarrow \alpha_i + 1, \quad \theta_0 \leftarrow \theta_0 + y^{(i)}$

(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.

The radial basis kernel includes all orders of polynomial features. Kernel perceptron with the radial basis kernel can therefore find a separating solution for any set of distinct training points.

(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.

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

The points could also be all negative.

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 <u>support vectors</u>. Check all that apply.

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

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

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

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

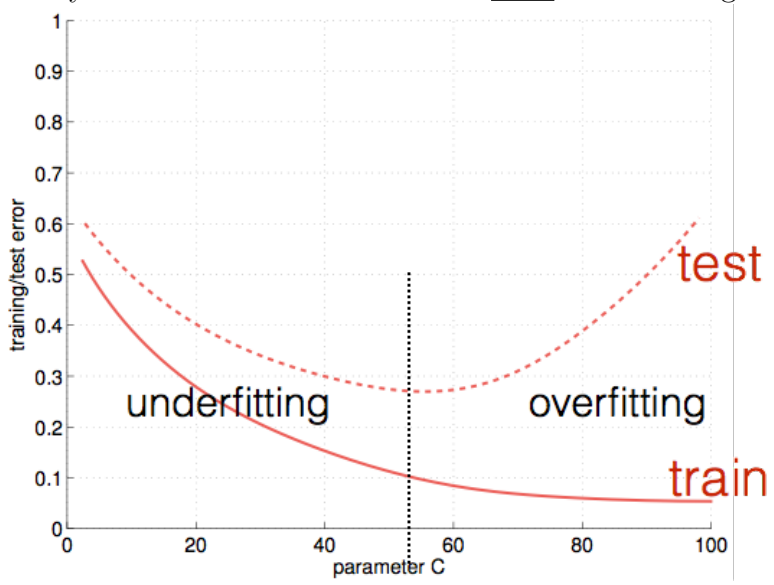
(b) **(2 points)** What is the orthogonal distance from the decision boundary to either of the margin boundaries?

distance $= 1/\|\theta\| = 1/\sqrt{5}$

(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.

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

(X) $\beta_{\hat{w}_2|z=1}\theta_{z=1|\hat{w}_1} / \big(\sum_{z'=1}^{k} \beta_{\hat{w}_2|z'}\theta_{z'|\hat{w}_1} \big)$ (9)

( ) $\beta_{\hat{w}_2|z=1}\theta_{z=1|\hat{w}_1} / \big(\sum_{w\in\mathcal{W}} \beta_{\hat{w}_2|z=1}\theta_{z=1|w} \big)$ (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 $\theta_{z|w}$.

$\sum_{t=1}^{T-1} \log \left[\sum_{z=1}^{k} \beta_{\hat{w}_{t+1}|z} \theta_{z|\hat{w}_t} \right]$

(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.

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

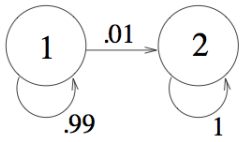
( ) $\hat{n}(z, w') = \sum_{t=1}^{T-1} P(z_t = z|\hat{w}_t) [\![\hat{w}_t = w']\!]$

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

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

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

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.

$x_1 = A$, $x_2 = G$

(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?

state = 2 because the model will eventually transition into state 2 but cannot transition out of it.

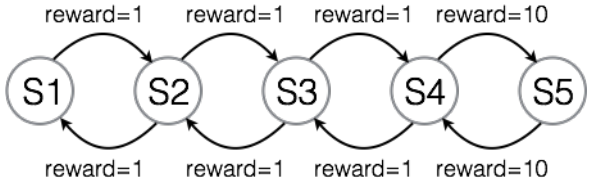
(c) **(4 points)** If we observe $x_1 = T$ and $x_2 = T$ (first two symbols). What is the most likely underlying hidden state sequence, i.e., s_1 and s_2, that generated these observations?

We can systematically evaluate the joint probability of events that result in $x_1 = T$ and $x_2 = T$:

$$
\begin{aligned}
& & & \overbrace{}^{\text{states}} \overbrace{}^{\text{outputs}} \\
P(s_1 = 1, s_2 = 1, x_1 = T, x_2 = T) &= (0.5\,0.99)\,(0.8\,0.8) \\
P(s_1 = 1, s_2 = 2, x_1 = T, x_2 = T) &= (0.5\,0.01)\,(0.8\,0.7) \\
P(s_1 = 2, s_2 = 1, x_1 = T, x_2 = T) &= (0.5\,0.0)\,(0.7\,0.8) \\
P(s_1 = 2, s_2 = 2, x_1 = T, x_2 = T) &= (0.5\,1.0)\,(0.7\,0.7)
\end{aligned}
$$

clearly, $P(s_1 = 1, s_2 = 1, x_1 = T, x_2 = T)$ is the highest.

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.

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

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

(2 points) one value iteration

1

(2 points) two value iterations

$1 + 0.5\,10 = 6.0$

(3 points) ∞ number of value iterations

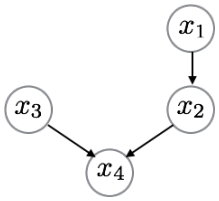
$1 + (\sum_{i=1}^{\infty} 0.5^i)10 = 11$

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

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

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.

$$\begin{aligned} P(x_1,x_3) &= \sum_{x_2,x_4} P_1(x_1)P_2(x_2|x_1)P_3(x_3)P_4(x_4|x_2,x_3) \\ &= P_1(x_1)P_3(x_3)\sum_{x_2} P_2(x_2|x_1)\sum_{x_4} P_4(x_4|x_2,x_3) \\ &= P_1(x_1)P_3(x_3) \end{aligned}$$

(c) **(4 points)** Let's introduce an alternative model, also fully parameterized, given by

$$\text{BN 2}: \qquad P(x_1,x_2,x_3,x_4) = P_1(x_1|x_2)P_2(x_2|x_3)P_3(x_3)P_4(x_4|x_2) \tag{14}$$

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.

We would prefer BN 2 because it has fewer parameters (fewer degrees of freedom) yet performs equally well.