Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.036 Introduction to Machine Learning

Midterm exam (March 23, 2017)

SOLUTION KEY

Last name: _____

First name: _____

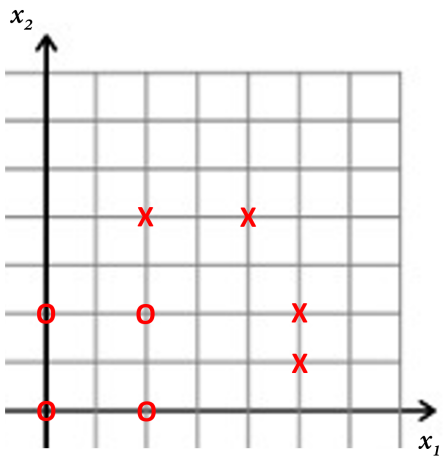
MIT ID: _____

- 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
17	10	12	17	13	69

Problem 1 Consider a labeled training set shown in the table and figure below:

Label	-1	-1	-1	-1	+1	+1	+1	+1
Coordinates	(0,0)	(2,0)	(0,2)	(2,2)	(5,1)	(5,2)	(2,4)	(4,4)
Perceptron mistakes	1	6	1	5	3	1	2	0



(1.1) **(4 points)** We initialize the parameters to all zero values and run the Perceptron algorithm through these points in a particular order until convergence. The number of mistakes made on each point are shown in the table. What is the resulting offset parameter θ_0?

Answer:
Let α_i be the number of mistakes that perceptron makes on the point $x^{(i)}$ with label $y^{(i)}$. The resulting offset parameter is

$$\theta_0 = \sum_{i=1}^{8} \alpha_i y^{(i)} = -7$$

(1.2) **(2 points)** The mistakes that the algorithm makes often depend on the order in which the points were considered. Could the point (4,4) labeled '+1' have been the first one considered? (Y/N) (N)

Reasoning:
When perceptron is initialized to all zeros, the first point considered is always a mistake. Since no mistakes were made on the point (4,4) labeled +1, it could not have been the first point considered.

Suppose that we now find the linear separator that **maximizes** the margin instead of running the perceptron algorithm.

(1.3) **(4 points)** What are the parameters θ and θ_0 corresponding to the maximum margin separator?

Answer:
The margin of a separator is the minimal distance between the separator and any point in the dataset. The equation of the line that maximizes the margin on the given points is $x_1 + x_2 - 5 = 0$. The parameters corresponding to the maximum margin separator are:

$$\theta = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \theta_0 = -5$$

.

(1.4) **(2 points)** What is the value of the margin attained?

Answer:
The support vectors (points closest to the max-margin separator) are (2,2), (2, 4) and (5,1). The distance between any one of these points and the separator is $\frac{\sqrt{2}}{2}$.
Alternatively, we know the margin is $\frac{1}{\|\theta\|} = \frac{1}{\sqrt{2}}$.

(1.5) **(2 points)** What is the sum of Hinge losses evaluated on each example?

Answer:
Since the points are perfectly separated, the hinge loss is 0.
Alternatively, the sum of the hinge losses can be calculated by:

$$\sum_{i=1}^{8} \max\{0, 1 - y^{(i)} \left(\theta \cdot x^{(i)} + \theta_0\right)\} = 0$$

(1.6) **(3 points)** Suppose we modify the maximum margin solution a bit and divide both θ and θ_0 by 2. What is the sum of Hinge losses evaluated on each example for this new separator?

Answer:
The sum of the hinge losses for the new parameters is:

$$\begin{aligned}\sum_{i=1}^{8} \max\left\{0, 1 - y^i\left(\frac{1}{2}\theta x^{(i)} + \frac{1}{2}\theta_0\right)\right\} &= 0 + 0 + 0 \\ &\quad + \left(1 - (-1)\frac{1}{2}[1 \;\; 1] \cdot [2 \;\; 2] - 2.5\right) \\ &\quad + \left(1 - \frac{1}{2}[1 \;\; 1] \cdot [5 \;\; 1] - 2.5\right) \\ &\quad + 0 + \left(1 - \frac{1}{2}[1 \;\; 1] \cdot [2 \;\; 4] - 2.5\right) + 0 \\ &\quad = 1.5\end{aligned}$$

We can also find the hinge loss visually. Since both θ and θ_0 are scaled by the same constant (0.5), the decision boundary stays the same, but the margin is twice what it was before. The points that were right on the margin [i.e. (2,2), (2, 4) and (5,1)] will now have a loss of 1/2 each, resulting in a total loss of 1.5.

Problem 2 Stochastic gradient descent (SGD) is a simple but widely applicable optimization technique. For example, we can use it to train a Support Vector Machine. The objective function in this case is given by (here without offset)

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

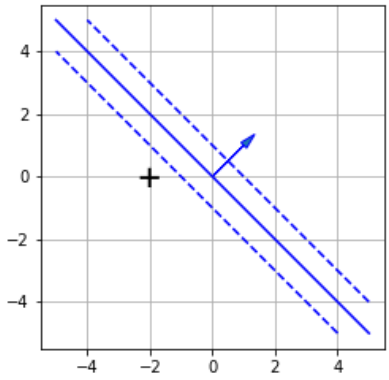
where $\text{Loss}_h(z) = \max\{0, 1-z\}$ is the Hinge loss.

(2.1) **(4 points)** Let θ be the current parameters. Provide an expression for the stochastic gradient update. Your expression should involve only θ, points, labels, λ, and the learning rate η.

Answer:
By definition, the update step for a random point $x^{(i)}$ with label $y^{(i)}$ is

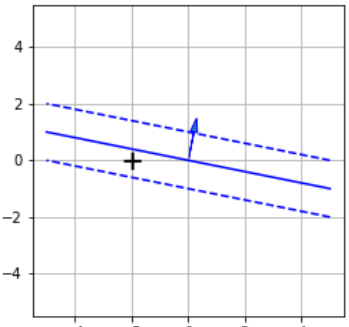
$$\begin{aligned}
\theta &\leftarrow \theta - \eta\nabla_\theta \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)}) + \frac{\lambda}{2}\|\theta\|^2\right] \\
&= \theta - \eta\nabla_\theta \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)})\right] - \nabla_\theta \left[\frac{\lambda}{2}\|\theta\|^2\right] \\
&= \theta - \eta\nabla_\theta \left[\text{Loss}_h(y^{(i)}\theta \cdot x^{(i)})\right] - \eta\lambda\theta \\
&= (1-\eta\lambda)\theta + \eta \begin{cases} y^{(i)}x^{(i)} & \text{if } y^{(i)}\theta \cdot x^{(i)} \le 1 \\ 0 & \text{o.w.} \end{cases}
\end{aligned}$$

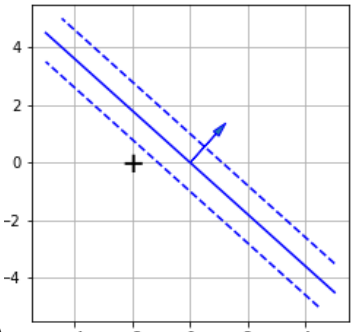
(2.2) **(6 points)** Consider the situation shown in this figure:

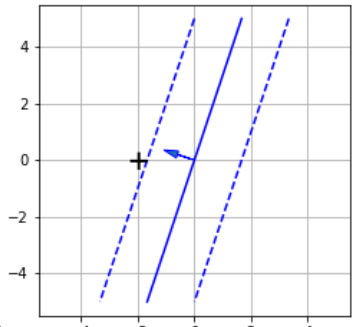


Associate the figures (A,B,C) below to a single SGD update made in response to the point labeled '+1' when we use

(B) small λ, small η, (A) small λ, large η, (C) large λ, large η.

A) 

B) 

C) 

Reasoning:
With small η, we should see very little change to the classifier; thus, this corresponds to figure B. With large λ and large η, we should see both an increase in the margin and large update to the classifer. This matches figure C, leaving figure A as small λ and large η. Also note that in figure A, the new θ can be visually obtained by adding a fraction of vector x (the point) to the previous θ. As a result, λ has to be small.

Problem 3 Suppose our rating matrix is just a 2x2 matrix and we are looking for a rank-1 solution UV^T so that user and movie features U and V are both 2x1 matrices. The observed rating matrix has only a single known entry

$$Y = \begin{bmatrix} ? & 1 \\ ? & ? \end{bmatrix} \tag{2}$$

In order to learn user/movie features, we minimize

$$J(U,V) = \frac{1}{2} \sum_{(a,i)\in D} (Y_{ai} - [UV^T]_{ai})^2 + \lambda(U_1^2 + V_1^2) \tag{3}$$

where the set D is just the observed entries of the matrix Y, in this case just $(1,2)$. *Note that the regularization we use applies only to the first coordinate of user/movie features.* We will see how things get a bit tricky with this type of partial regularization.

(3.1) **(3 points)** If we initialize $U = [u, 1]^T$, for some $u > 0$, what is the solution to V as a function of λ and u?

Answer:
Notice that J only regularizes on the first coordinate. Thus, we only want to minimize $J(v_1, v_2) = \frac{1}{2}(1 - uv_2)^2 + \lambda v_1^2$ given that $V = [v_1, v_2]^T$. We can see that J is minimized when $v_1 = 0$, $v_2 = \frac{1}{u}$. Therefore,

$$V = \left[0, \frac{1}{u}\right]^T$$

(3.2) **(3 points)** What is the resulting value of $J(U,V)$ as a function of λ and u?

Answer:
Notice that J only regularizes on the first coordinate. Therefore,

$$\begin{aligned} J(U,V) &= \frac{1}{2} \sum_{(a,i)\in D} (Y_{ai} - [UV^T]_{ai})^2 + \lambda(U_1^2 + V_1^2) \\ &= \frac{1}{2}(1 - u_1 v_2)^2 + \lambda(u_1^2 + v_1^2) \\ &= \frac{1}{2}\left(1 - u \cdot \frac{1}{u}\right)^2 + \lambda(u^2 + 0^2) \\ &= \lambda u^2 \end{aligned}$$

(3.3) **(3 points)** If we continue to iteratively solve for U and V, what would U and V converge to?

Answer:
The regularization error is minimized when u_1 and v_1 are 0. Over many iterations, u_1 will eventually converge to zero. The squared error term $\frac{1}{2}(1 - u_1 v_2)^2$ is minimzed when $u_1 v_2 = 1$. Since u_1 converges to 0, v_2 diverges to ∞. In the limit, we conclude:

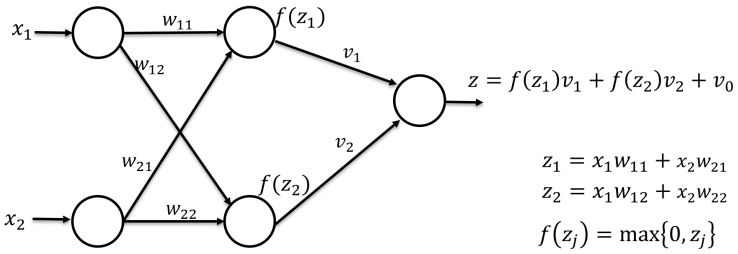
$$U \to [0, 1]^T$$
$$V \to [0, \infty]^T$$

(3.4) **(3 points)** We often need more features for each user/item. Not all rating matrices can be reproduced by UV^T when we restrict the dimensions of U and V to be 2x1. For each matrix, answer Y or N according to whether it can be reproduced by such U and V.

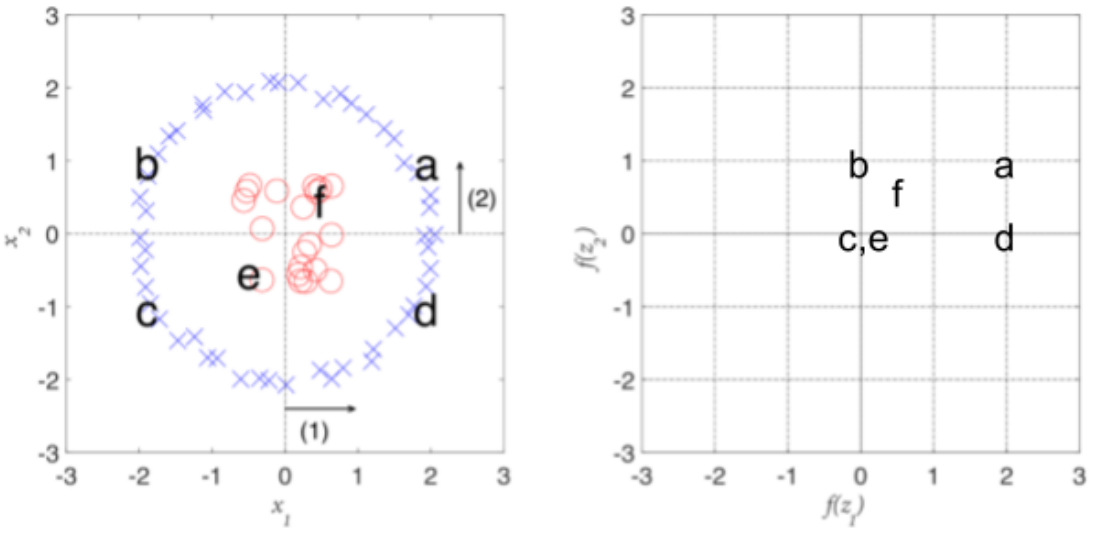
$$(\text{ Y }) \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (\text{ N }) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (\text{ Y }) \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \tag{4}$$

Reasoning:
In order for a matrix M to be reproduced by UV^T, we must have $[u_1, u_2]^T [v_1, v_2] = M$. For the second matrix, this would require $u_1 v_1 = 1$, $u_1 v_2 = 0$, $u_2 v_1 = 0$, $u_2 v_2 = 1$, which has no solution. The first matrix can be represented as $\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix}$ and the third can be represented as $\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$.

Problem 4 Consider a 2-layer feed-forward neural network that takes in $x \in \mathbb{R}^2$ and has two ReLU hidden units. *Note that the hidden units have no offset parameters.*



(4.1) **(6 points)** The values of the weights in the hidden layer are set such that they result in the z_1 and z_2 "classifiers" as shown in the figure by the decision boundaries and the corresponding normal vectors marked as (1) and (2). Approximately sketch on the right how the input data is mapped to the 2-dimensional space of hidden unit activations $f(z_1)$ and $f(z_2)$. Only map points marked 'a' through 'f'. Keep the letter indicators.



(4.2) **(2 points)** If we keep these hidden layer parameters fixed but add and train additional hidden layers (applied after this layer) to further transform the data, could the resulting neural network solve this classification problem? (Y/N) (N)

Reasoning: Since points c and e are mapped to the origin but have opposite label, it is impossible to classify them both correctly.

(4.3) **(3 points)** Suppose we stick to the 2-layer architecture but add lots more ReLU hidden units, all of them without offset parameters. Would it be possible to train such a model to perfectly separate these points? (Y/N) **(** Y **)**

(4.4) **(3 points)** Initialization of the parameters is often important when training large feed-forward neural networks. Which of the following statements is correct? Check T or F for each statement.

(T **)** If we use tanh or linear units and initialize all the weights to very small values, then the network initially behaves as if it were just a linear classifier

(T **)** If we randomly set all the weights to very large values, or don't scale them properly with the number of units in the layer below, then the tanh units would behave like sign units.

(T **)** A network with sign units cannot be effectively trained with stochastic gradient descent

(4.5) **(3 points)** There are many good reasons to use convolutional layers in CNNs as opposed to replacing them with fully connected layers. Please check T or F for each statement.

(T **)** Since we apply the same convolutional filter throughout the image, we can learn to recognize the same feature wherever it appears.

(T **)** A fully connected layer for a reasonably sized image would simply have too many parameters

(F **)** A fully connected layer can learn to recognize features anywhere in the image even if the features appeared preferentially in one location during training

Problem 5 Consider a simple recurrent neural network model given by

$$s_t \;=\; \text{sign}_0(W^{s,s}s_{t-1} + W^{s,x}x_t), \quad t = 1, 2, \ldots \tag{5}$$

where we have omitted any offset parameters (setting them to zero) and the non-linear activation function, applied element-wise, is a sign function that also returns zero: $\text{sign}_0(z) = 1$ if $z > 0$, $\text{sign}_0(0) = 0$, and -1 otherwise. In our problem here, $s \in \mathbb{R}^2$ and $x \in \mathbb{R}$. The parameters are set (not learned) as follows:

$$W^{s,s} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad W^{s,x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad s_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{6}$$

We will use the RNN model to translate variable length input sequences to corresponding state vectors. In other words, a sequence x_1, x_2 will be represented by the resulting s_2; similarly, x_1, x_2, x_3 will be represented by s_3, and so on.

Once each sequence has an associated state vector, we can use them, e.g., to train a linear classifier. To this end, our three training examples are binary sequences

$$x^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1, 0), \quad x^{(2)} = (x_1^{(2)}, x_2^{(2)}, x_3^{(2)}) = (0, 0, 1), \quad x^{(3)} = (x_1^{(3)}) = (0) \tag{7}$$

(5.1) **(4 points)** What is the sequence of first two state vectors s_1 and s_2 resulting from feeding the RNN model with the input sequence $x^{(1)}$?

$$s_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad s_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{8}$$

(5.2) **(3 points)** Map all the sequences to their corresponding state vector representations

$$x^{(1)} \to \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad x^{(2)} \to \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x^{(3)} \to \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9}$$

(5.3) **(2 points)** Suppose we train a linear classifier that operates on the state vector representations of these sequences $x^{(1)}$, $x^{(2)}$, and $x^{(3)}$ (run through our RNN). Could the linear classifier separate these three examples regardless of how they were labeled? (Y/N) **(N)**

(5.4) **(4 points)** We can think of the RNN model as giving rise to an equivalent feed-forward neural network for any input sequence $(x_1, x_2, \ldots, x_n)$ run through it. Mark T or F for each statement based on whether it is correct for this feed-forward interpretation:

(T) The parameters of each layer are shared (same transformation)

(T) Each input x_i is fed into a different hidden layer

(T) The number of layers is proportional to the length of the input sequence

(T) Each hidden layer would have two units in case of our RNN above