First Name: ____________ Last Name: ____________

Student Number: ____________

ECE 421F — Introduction to Machine Learning
MidTerm Examination

Wed Oct 16^{th}, 2019
4:10-6:00 p.m.

Instructor: Ashish Khisti

Circle your tutorial section:

- TUT0101 Thu 1-3
- TUT0102 Thu 4-6

Instructions

- Please read the following instructions carefully.
- You have 1 hour fifty minutes (1:50) to complete the exam.
- Please make sure that you have a complete exam booklet.
- Please answer *all* questions. Read each question carefully.
- The value of each question is indicated. Allocate your time wisely!
- No additional pages will be collected beyond this answer book. You may use the reverse side of each page if needed to show additional work.
- This examination is closed-book; One 8.5 × 11 aid-sheet is permitted. A non-programmable calculator is also allowed.
- Good luck!

1. (40 MARKS) Consider a multi-class linear classification problem where the data points are two dimensional, i.e., $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ and the labels $y \in \{1, 2, 3\}$. Throughout this problem consider the data-set with following five points:

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4), (\mathbf{x}_5, y_5)\}$$

where the input data-vectors are given by:

$$\mathbf{x}_1 = (-1, 0)^T, \quad \mathbf{x}_2 = (1, 0)^T, \quad \mathbf{x}_3 = (1, 1)^T, \quad \mathbf{x}_4 = (-1, 1)^T, \quad \mathbf{x}_5 = (0, 3)^T$$

and the associated labels are given by

$$y_1 = 1, \quad y_2 = 2, \quad y_3 = 2, \quad y_4 = 1, \quad y_5 = 3$$

Our aim is to find a linear classification rule that classifies this dataset.

10 marks

(a) Suppose we implement the perceptron learning algorithm for binary classification that finds a perfect classifier separating the data points between the two sets: $\mathcal{S}_1 = \{(\mathbf{x}_1, y_1), (\mathbf{x}_4, y_4)\}$ and $\mathcal{S}_2 = \{(\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3)\}$.
Assume that the initial weight vector $\mathbf{w} = (0, 0, 0)^T$, that each point that falls on the boundary is treated as a mis-classified point and the algorithm visits the points in the following order:

$$\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3 \to \mathbf{x}_4 \to \mathbf{x}_1 \to \mathbf{x}_2 \cdots$$

until it terminates. Show the output of the perceptron algorithm in each step and sketch the final decision boundary when the algorithm terminates. [**Important**: When applying the perceptron update, recall that you have to transform the data vectors to include the constant term i.e., $\mathbf{x}_1 = (-1, 0)^T$ must be transformed to $\tilde{\mathbf{x}}_1 = (1, -1, 0)^T$ etc.]

$x_1, x_4 \rightarrow y = 1$ (+ve class)

$x_2, x_3 \rightarrow y = -1$ (-ve class)

$$\text{Sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Step1: $\hat{y}_1 = \text{Sign}(w^{(0)T}\tilde{x}_1)$

$= \text{Sign}\left([0\ 0\ 0]\begin{bmatrix}1\\-1\\0\end{bmatrix}\right)$

$= \text{Sign}(0)$

$= 0 \rightarrow$ misclassified

$w^{(1)} = w^{(0)} + (y_1)\tilde{x}_1$

$= \begin{bmatrix}0\\0\\0\end{bmatrix} + 1\begin{bmatrix}1\\-1\\0\end{bmatrix} = \begin{bmatrix}1\\-1\\0\end{bmatrix}$

$= \begin{bmatrix}1\\-1\\0\end{bmatrix}$

Step2: $\hat{y}_2 = \text{Sign}(w^{(1)T}\tilde{x}_2)$

$= \text{Sign}\left([1\ -1\ 0]\begin{bmatrix}1\\1\\0\end{bmatrix}\right)$

$= \text{Sign}(0) = 0 \rightarrow$ misclas

$w^{(2)} = w^{(1)} + y_2\tilde{x}_2$

$= \begin{bmatrix}1\\-1\\0\end{bmatrix} + (-1)\begin{pmatrix}1\\1\\0\end{pmatrix}$

$= \begin{bmatrix}0\\-2\\0\end{bmatrix}$;

total/-

[continue part (a) here]

Step 3: $\hat{y}_3 = \text{Sign}(w^{(2)T}\hat{x}_3)$

$$= \text{Sign}\left(\begin{bmatrix} 0 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right)$$

$$= \text{Sign}(0-2+0)$$

$$= -1 \longrightarrow \text{classified} \qquad w^3 = w^2$$

Step 4: $\hat{y}_4 = \text{Sign}(w^{(3)T}\hat{x}_4)$

$$= \text{Sign}\left(\begin{bmatrix} 0 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}\right)$$

$$= 1 \rightarrow \text{classified} \qquad w^4 = w^3$$

Step 5: $\hat{y}_1 = \text{Sign}(w^{4T} x)$

$$= \text{Sign}\left(\begin{bmatrix} 0 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}\right)$$

$$= +1 \rightarrow \text{classified}$$

$h_w(x) = y = \text{Sign}(w_0 + w_1x_1 + w_2x_2)$

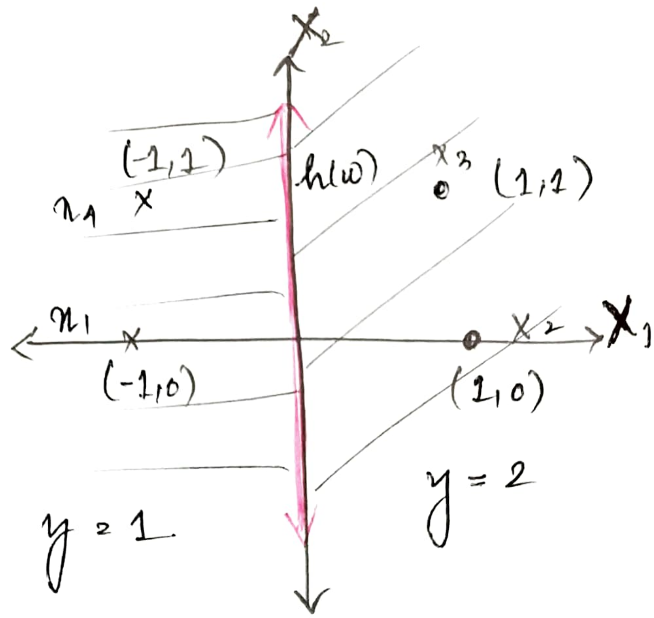
$= \text{sign}(-2x_1)$

$w^5 = w^4$

Step 6: $\hat{y}_2 = \text{Sign}(w^{5T}\hat{x}_2)$

$$= \text{Sign}\left(\begin{bmatrix} 0 & -2 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}\right)$$

$$= -1 \rightarrow \text{classified}$$

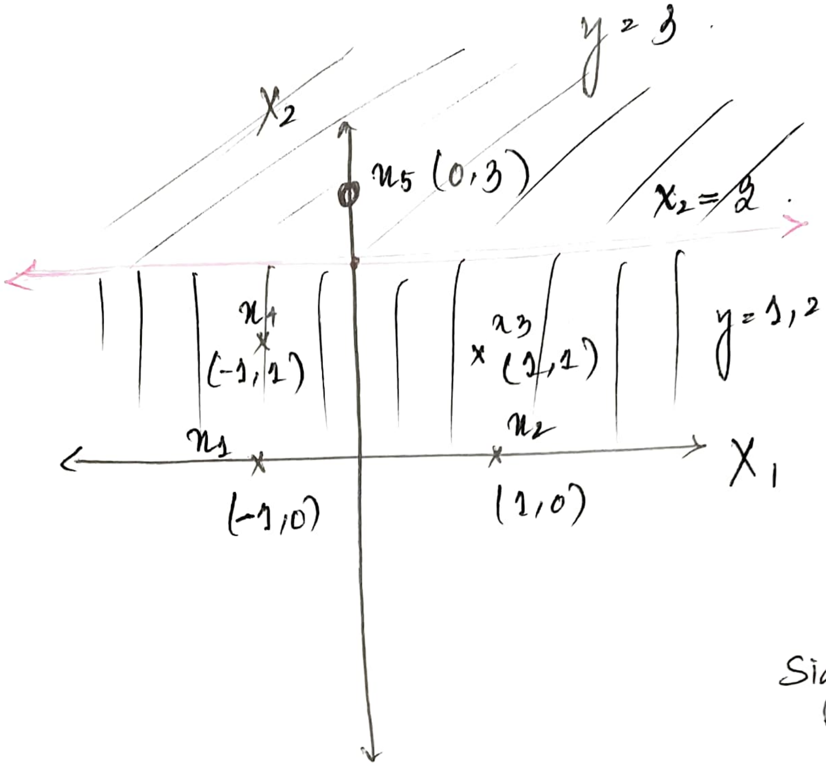


$$w^6 = w^5 = \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix} = w^*$$

total/10

5 marks

(b) Find any linear classification rule that perfectly separates the data points between the two sets: $\mathcal{S}_{12} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)\}$ and $\mathcal{S}_3 = \{(\mathbf{x}_5, y_5)\}$. Draw your decision boundary and clearly mark the labels for all the decision regions. You need not use a perceptron algorithm to find the classification rule.



(one of the many possible linear seperators)

$x_2 = 2$

$\text{Sign}(2 + 0x_1 - x_2) = h(x)$

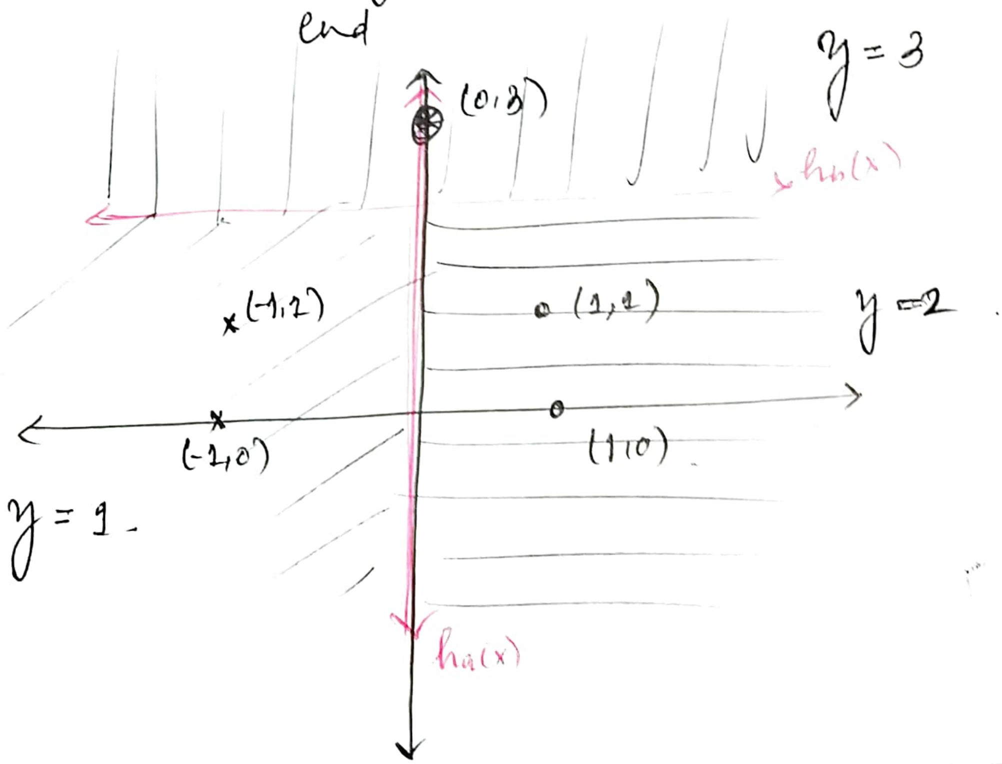
$$w^* = \begin{bmatrix} 2 \\ 0 \\ -1 \end{bmatrix} \text{ or } \begin{bmatrix} -2 \\ 0 \\ 1 \end{bmatrix}$$

total/10

10 marks (c) Explain how to combine parts (a) and (b) to develop a classification rule that given any input $\mathbf{x} \in \mathbb{R}^2$ outputs a label $\hat{y} \in \{1, 2, 3\}$. Your classification rule must achieve perfect classification on the training set. Sketch your decision boundaries in $\mathbb{R}^2$ and show the labels associated with each decision region.

Ans:

1. Pass through classifier in part (b) ; $h_b(x)$
2. If $\hat{y} = -1$ then:
   $y = 3$
   end
3. else if $\hat{y} = 1$ then:
4. Pass through classifier in part (a) ; $h_a(x)$
5. If $\hat{y} = -1$:
   $y = 2$
6. else if $\hat{y} = +1$
   $y = 1$
   end



total/10

10 marks

(d) Suppose we wish to implement a multi-class logistic regression model for classifying the training set $\mathcal{D}$. Let $\Omega = \{\mathbf{w}(1), \mathbf{w}(2), \mathbf{w}(3)\}$ denote the model parameters of your logistic regression model where $\mathbf{w}(i) \in \mathbb{R}^3$ is the weight vector associated with class label $y = i$. Given an input data vector $\mathbf{x} = (x_1, x_2)^T$ the model outputs is a probability vector:

$$\hat{p}_\Omega(i|\mathbf{x}) = \frac{e^{[\mathbf{w}^T(i)\cdot\tilde{\mathbf{x}}]}}{\sum_{j=1}^{3} e^{[\mathbf{w}^T(j)\cdot\tilde{\mathbf{x}}]}}, \quad i = 1, 2, 3$$

where $\tilde{\mathbf{x}} = (x_0 = 1, x_1, x_2)^T \in \mathbb{R}^3$ is the augmented vector of $\mathbf{x}$ as discussed in class. We assume a standard log-loss function for the training error, i.e.,

$$E_{\text{in}}(\Omega) = \frac{1}{5}\sum_{n=1}^{5} e_n(\Omega), \qquad e_n(\Omega) = -\log \hat{p}_\Omega(y_n|\mathbf{x}_n)$$

Assuming that we select $\mathbf{w}(1) = (1,0,0)^T$, $\mathbf{w}(2) = (0,1,0)^T$ and $\mathbf{w}(3) = (0,0,1)^T$ numerically evaluate $\nabla_{\mathbf{w}(j)}\{e_1(\Omega)\}$ for $j = 1, 2, 3$.

$$\nabla_{w(j)}\{e_{n}(\Omega)\} = \nabla_{w(j)}\left(-\log p_\Omega\left(y_n = \ell / \hat{x}_n\right)\right)$$

$$= \nabla_{(w(j))}\left[-\log\left(\frac{e^{w^T(\ell)\hat{x}_n}}{\sum_{k=1}^{3} e^{w^T(k)\hat{x}_n}}\right)\right]$$

$$= -\nabla_{w(j)}\left(w^T(\ell)\hat{x}_n\right) + \frac{\nabla_{w(j)}\left(\sum_{k=1}^{3} e^{w^T(k)\hat{x}_n}\right)}{\sum_{k=1}^{3} e^{w^T(k)\hat{x}_n}}$$

$$= -\hat{x}_n\delta(\ell = j) + \frac{\hat{x}_n e^{w^T(j)\hat{x}_n}}{\sum_{k=1}^{3} e^{w^T(k)\hat{x}_n}}$$

for $n = 1$; $y = \ell = 1$;

for $j = 1$;

$$\nabla_{w(1)}\{e_1(\Omega)\} = -\hat{x}_1 + \frac{\tilde{x}_1 e^{w^T(1)\hat{x}_1}}{\sum_{k=1}^{3} e^{w^T(k)\hat{x}_1}}$$

$$= -\hat{x}_1 + \frac{\tilde{x}_1 e^{w^T(1)\hat{x}_1}}{\sum_{k=1}^{3} e^{w^T(k)\hat{x}_1}}; \quad \longleftarrow \ 1$$

total/-

[continue part (d) here]

for $j=2$:

$$\nabla_{\omega(2)}\{e_1(\Omega)\} = \frac{\hat{x}_1 e^{\omega^T(2)\hat{x}_1}}{\sum_{k=1}^{3} e^{\omega^T(k)\hat{x}_1}} \quad (2)$$

for $j=3$:

$$\nabla_{\omega(3)}\{e_1(\Omega)\} = \frac{\hat{x}_1 e^{\omega^T(3)\hat{x}_1}}{\sum_{k=1}^{3} e^{\omega^T(k)\hat{x}_1}} \quad (3)$$

Substituting $\hat{x}_1 = [1\ -1\ 0]^T$ and $\omega(1) = [1\ 0\ 0]^T$ in (1)

$$\nabla_{\omega(1)}\{e_1(\Omega)\} = -\begin{pmatrix}1\\-1\\0\end{pmatrix}\left(1 - \frac{e^{[1\,0\,0]\begin{bmatrix}1\\-1\\0\end{bmatrix}}}{e^{[1\,0\,0][1\,-1\,0]^T} + e^{[0\,1\,0][1\,-1\,0]^T} + e^{[0\,0\,1][1\,-1\,0]^T}}\right)$$

$$= -\begin{pmatrix}1\\-1\\0\end{pmatrix}\left(1 - \frac{e^1}{e^0+e^1+e^{-1}}\right)$$

$$= \begin{pmatrix}-0.334\\0.334\\0\end{pmatrix}$$

Sub $\hat{x}_1$ and $\omega(2)$ in (2)

$$\nabla_{\omega(2)}\{e_1(\Omega)\} = \begin{bmatrix}1\\-1\\0\end{bmatrix}\left(\frac{e^{[0\,1\,0][1\,-1\,0]^T}}{e^0+e^1+e^{-1}}\right) = \begin{pmatrix}1\\-1\\0\end{pmatrix}\left(\frac{e^{-1}}{e^0+e^1+e^{-1}}\right) = \begin{pmatrix}0.09\\-0.09\\0\end{pmatrix}$$

Sub $\hat{x}_1$ and $\omega(3)$ in (3):

$$\nabla_{\omega(3)}\{e_1(\Omega)\} = \begin{bmatrix}1\\-1\\0\end{bmatrix}\left(\frac{e^{[0\,0\,1][1\,-1\,0]^T}}{e^0+e^1+e^{-1}}\right) = \begin{bmatrix}1\\-1\\0\end{bmatrix}\left(\frac{e^0}{e^0+e^1+e^{-1}}\right) = \begin{pmatrix}0.245\\-0.245\\0\end{pmatrix}$$

total/10

5 marks (e) For the problem in part (d), find the output of one-step update of the stochastic gradient descent (SGD) algorithm when the selected training example is $n = 1$, and ϵ is selected as the learning rate.

Update with SGD

$$w(1) = w(1) - \varepsilon\left(\nabla_{w(1)}\{e_1(\Omega)\}\right)$$

$$= \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} - \varepsilon\begin{pmatrix} -0.334 \\ 0.334 \\ 0 \end{pmatrix} = \begin{pmatrix} 1+\varepsilon 0.334 \\ -\varepsilon 0.334 \\ 0 \end{pmatrix}$$

$$w(2) = w(2) - \varepsilon\{\nabla_{w(2)}\{e_1(\Omega)\}\}$$

$$= \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} - \varepsilon\begin{pmatrix} 0.09 \\ -0.09 \\ 0 \end{pmatrix} = \begin{pmatrix} 0-\varepsilon 0.09 \\ 0+\varepsilon 0.09 \\ 0 \end{pmatrix}$$

$$w(3) = w(3) - \varepsilon\{\nabla_{w(3)}\{e_1(\Omega)\}\}$$

$$= \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} - \varepsilon\begin{pmatrix} 0.245 \\ -0.245 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} \cancel{0}-\varepsilon 0.245 \\ +\varepsilon 0.245 \\ 1 \end{pmatrix}$$

total/5

2. (40 MARKS) Consider a linear regression model where the training set is specified by

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\},$$

with $\mathbf{x}_i \in \mathbb{R}^{d+1}$ and $y_i \in \mathbb{R}$. We assume that each data vector is in the augmented dimension i.e., $\mathbf{x}_i = (x_{i,0} = 1, x_{i,1}, \ldots x_{i,d})$. We aim to find a weight vector $\mathbf{w} \in \mathbb{R}^{d+1}$ that aims to minimized a **weighted** squared error loss function

$$E_{\text{in}}^{\Lambda}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \lambda_i \cdot (\mathbf{w}^T \mathbf{x}_i - y_i)^2,$$

where $\mathbf{\Lambda} = (\lambda_1, \ldots, \lambda_N)^T$ is a pre-specified vector of **non-negative** constants that determine the importance of each sample.

Suppose that $\mathbf{w}^\star$ minimizes the weighted squared error loss i.e.,

$$\mathbf{w}^\star = \arg\min_{\mathbf{w} \in \mathbb{R}^{d+1}} E_{\text{in}}^{\Lambda}(\mathbf{w}). \tag{1}$$

10 marks (a) The optimal solution $\mathbf{w}^\star$ can be expressed as a solution to the following expression: $\mathbf{M} \cdot \mathbf{w}^\star = \mathbf{U} \cdot \mathbf{y}$ where $\mathbf{M}$ and $\mathbf{U}$ are matrices of dimension $(d+1) \times (d+1)$ and $(d+1) \times N$ respectively and $\mathbf{y} = [y_1, \ldots, y_N]^T$ is the observation vector. Provide an expression for $\mathbf{M}$ and $\mathbf{U}$ in terms of the following matrices:

$$\mathcal{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times (d+1)} \qquad \mathcal{L} = \begin{bmatrix} \sqrt{\lambda_1} & 0 & \cdots & 0 \\ 0 & \sqrt{\lambda_2} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\lambda_N} \end{bmatrix} \in \mathbb{R}^{N \times N}.$$

Please note that $\mathcal{L}$ is a diagonal matrix where the j–th diagonal entry is $\sqrt{\lambda_j}$.

Writing in Matrix and Vector form:

$$w^* = \arg\min_{w} \| \mathcal{L}(\mathcal{X}w - y)\|^2$$

$$0 = \nabla_w \left((\mathcal{L}(\mathcal{X}w - y))^T (\mathcal{L}(\mathcal{X}w - y)) \right)$$

$$0 = \nabla_w \left((\mathcal{X}w - y)^T \mathcal{L}^T \mathcal{L} (\mathcal{X}w - y) \right)$$

$$0 = \nabla_w \left((w^T \mathcal{X}^T - y^T) \mathcal{L}^T \mathcal{L} (\mathcal{X}w - y) \right)$$

$$0 = \nabla_w \left((w^T \mathcal{X}^T \mathcal{L}^T \mathcal{L} - y^T \mathcal{L}^T \mathcal{L})(\mathcal{X}w - y) \right)$$

$$0 = \nabla_w \Big(w^T (\mathcal{X}^T \mathcal{L}^T \mathcal{L} \mathcal{X}) w - w^T (\mathcal{X}^T \mathcal{L}^T \mathcal{L} y)$$
$$- y^T \mathcal{L}^T \mathcal{L} y - y^T (\mathcal{L}^T \mathcal{L}) \mathcal{X} w$$

$$0 = 2\mathcal{X}^T \mathcal{L}^T \mathcal{L} \mathcal{X} w - \mathcal{X}^T \mathcal{L}^T \mathcal{L} y - \mathcal{X}^T \mathcal{L}^T \mathcal{L} y$$

total/-

[continue part (a) here]

$$2(X^T L^T L X)\, w = 2(X^T L^T L)\, y$$

$$(X^T L^T L X)\, w = (X^T L^T L)\, y$$

$$\therefore M = X^T L^T L X \in R^{(d+1 \times d+1)}$$

$$U = X^T L^T L \in R^{(d+1) \times N}$$

total/10

10 marks (b) Provide an expression for $\nabla_{\mathbf{w}}\left(E_{\text{in}}^{\Lambda}(\mathbf{w})\right)$ and use it to provide a (full) gradient descent algorithm for numerically computing the optimal solution $\mathbf{w}^{\star}$. Assume that a constant learning rate of ϵ is used in the algorithm.

For full grad descent:

$$\nabla_w\left(\hat{E}_{in}(w)\right) = \nabla_w\left(\frac{1}{N}\sum_{i=1}^{N}\lambda_i\left(w^T x_i - y_i\right)^2\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N}\lambda_i \nabla_w\left(w^T x_i - y_i\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\lambda_i\left(2\left(w^T x_i - y_i\right)\nabla_w\left(w^T x_i - y_i\right)\right)$$

$$= \frac{2}{N}\sum_{i=1}^{N}\lambda_i\left(w^T x_i - y_i\right)x_i$$

Update expression:

$$w(t+1) = w(t) - \epsilon\nabla_w\left(E_{\hat{in}}(w)\right)$$

$$w(t+1) = w(t) - \frac{2\epsilon}{N}\sum_{i=1}^{N}\lambda_i\left(w^T(t)x_i - y_i\right)x_i ;$$

total/10

5 marks

(c) Using gradient analysis in part (b), provide a stochastic gradient descent (SGD) algorithm for numerically computing the optimal solution $\mathbf{w}^\star$. Assume that a constant learning rate of ϵ is used in the algorithm.

① Initialize weights at $t=0$ to $w(0)$

~~for t=1,2,... epochs do:~~ for $t=1,2,\ldots$ epochs do:

② for $n \sim$ uniform $\{1,2,\ldots N\}$ do:

③ Compute gradient $g_n(t) = \nabla e_n(w)$

$$= 2\lambda_n (w^T x_n - y_n) x_n$$

④ Update weights $w(t+1) = w(t) - \eta g_n(t)$

⑤ Return final weights

(can have diff stopping criteria. eg: no. of epochs or $E_{in} = 0$ etc)

5 marks

(d) List any two advantages of using the SGD algorithm over the solution in part (a)

① Does not need to compute $(X^TX)^{-1}$ of input matrix, saves memory, computationally efficient.

② Not sensitive to outliers.

③ Does not easily overfit to training data, generalizing well.

total/10

10 marks

(e) Suppose we wish to find $\mathbf{w}^\star_\beta$ minimizes the following:

$$\mathbf{w}^\star_\beta = \arg\min_{\mathbf{w}\in\mathbb{R}^{d+1}} \left\{ E_{\text{in}}^{\mathbf{\Lambda}}(\mathbf{w}) + \beta \cdot \|\mathbf{w}\|^2 \right\}, \tag{2}$$

where $E_{\text{in}}^{\mathbf{\Lambda}}(\mathbf{w})$ is the weighted squared error loss as in part (a), $\|\mathbf{w}\|^2$ is the squared Eucledian norm of $\mathbf{w}$ and $\beta > 0$ is the regularization constant. Provide an analytical closed-form expression for $\mathbf{w}^\star_\beta$ in terms of $\mathcal{X}$, $\mathcal{L}$, $\mathbf{y}$, the identity matrix, and the constant β.

$$w_\beta^* = \arg\min_{w} \{ \| \mathcal{L}(Xw - y)\|^2 + \beta \|w\|^2 \}.$$

$$0 = \nabla_w \left((\mathcal{L}(Xw-y))^T (\mathcal{L}(Xw-y)) + \beta w^T w \right)$$

$$= \nabla_w (w^T (X^T \mathcal{L}^T \mathcal{L} X) w - w^T (X^T \mathcal{L}^T \mathcal{L} y) - y^T \mathcal{L}^T \mathcal{L} X w$$

$$+ y^T \mathcal{L}^T \mathcal{L} y + \beta w^T w)$$

$$0 = 2(X^T \mathcal{L}^T \mathcal{L} X) w^* - 2(X^T \mathcal{L}^T \mathcal{L}) y + 2\beta w^*$$

$$\cancel{(X^T \mathcal{L}^T \mathcal{L} X)(w) + \beta w = (X^T \mathcal{L}^T \mathcal{L}}$$

$$(X^T \mathcal{L}^T \mathcal{L} X) w + (\beta)(w^*) = (X^T \mathcal{L}^T \mathcal{L}) y$$

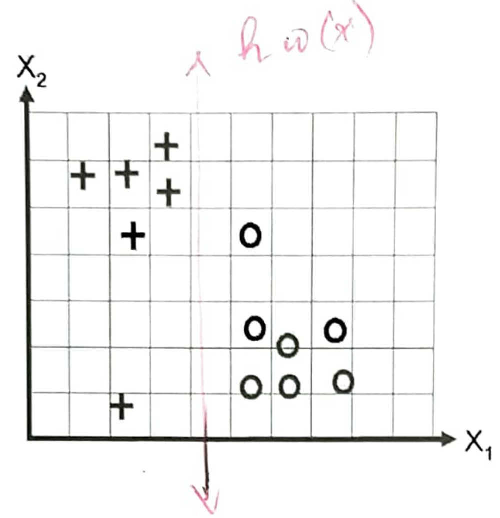
$$(X^T \mathcal{L}^T \mathcal{L} X + \beta I)(w^*) = (X^T \mathcal{L}^T \mathcal{L}) y$$

$$w^* = \underline{(X^T \mathcal{L}^T \mathcal{L} X + \beta I)^{-1} (X^T \mathcal{L}^T \mathcal{L}) y}$$

total/10

20 marks

3. Consider a binary linear classification problem where $\mathbf{x} \in \mathbb{R}^2$ and $y \in \{-1, +1\}$. We illustrate the training dataset below. The '+' label refers to $y = +1$ and the 'o' label refers to $y = -1$. We would like to construct a classifier $h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2)$ where $\text{sign}(\cdot)$ is the *sign* function as discussed in class.



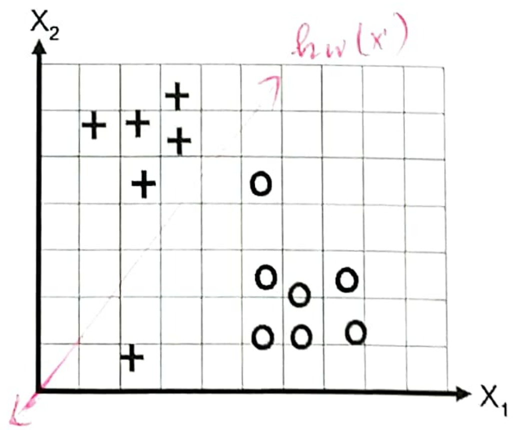
In the figure above, the adjacent vertical (and horizontal) lines are 1 unit apart from each other. Assume that the training points are above are $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ (with $N = 13$). We consider the classification loss

$$L(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(y_i \neq h_{\mathbf{w}}(\mathbf{x}_i))$$

where $\mathbb{I}$ denotes the indicator function.

2 marks (a) Draw a decision boundary in the figure above that achieves zero training error.

6 marks (b) Suppose that we attempt to minimize the following loss function over $\mathbf{w}$: $J(\mathbf{w}) = L(\mathbf{w}) + \lambda w_0^2$, where $\lambda = 10^7$ is a huge constant. Sketch a possible decision boundary in the figure below. How many points are mis-classified.



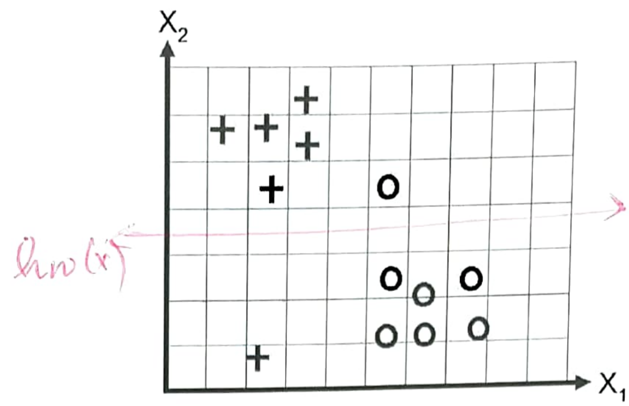
here:
When $\lambda \to \infty$
$w_0 \to 0$
(i.e, decision boundary is ~~pla~~ passing through origin)

1 point is missclassified

total/8

6 marks

(c) Suppose that we attempt to minimize the following loss function over $\mathbf{w}$: $J(\mathbf{w}) = L(\mathbf{w}) + \lambda w_1^2$, where $\lambda = 10^7$ is a huge constant. Sketch a possible decision boundary in the figure below. How many points are mis-classified.

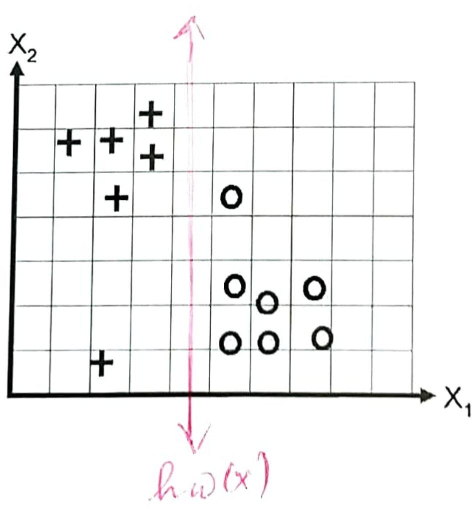


here:
when $\lambda \to \infty$
$w_1 \to 0$
(decision boundary is parallel to x_1)

2 points misclassified.

6 marks

(d) Suppose that we attempt to minimize the following loss function over $\mathbf{w}$: $J(\mathbf{w}) = L(\mathbf{w}) + \lambda w_2^2$, where $\lambda = 10^7$ is a huge constant. Sketch a possible decision boundary in the figure below. How many points are mis-classified.



here:
when $\lambda \to \infty$
$w_2 \to 0$
(decision boundary parallel to x_2)

No points misclassified.

total/12