ECSE 4840 Introduction to Machine Learning
Term Exam 2

Rensselaer Polytechnic Institute

Due: 2PM on December 11th

Name: Saaif Ahmed

RIN ID: 661925946

Question	**Points**
1	0
2	0
3	0
4	0
Total	0

Instructions:

1. This examination contains 8 pages, including this page.
2. The exam has no **grace days**.
3. Write your answers or type them in latex. We scan this into Gradescope, so **please try to avoid writing on the backs of pages**. If you must do so, please indicate **very** clearly on the front of the page that you have written on the back of the page.
4. You may use notes that you have prepared. You may not use any other electrical resources, other students, other instructors, or other engineers.
5. You may use a calculator. You may not share a calculator with anyone.

I certify that I will neither give nor receive unpermitted aid on this examination.

Signature: Saaif Ahmed

Question 1: Gradient Descent for Linear Regression (30 points)

Consider the linear regression problem of finding $\boldsymbol{\theta}$ that minimizes the following ℓ_2-regularized loss function

$$L(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{\theta}^\top \mathbf{x}_i - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \tag{1}$$

where $\mathcal{D} = \left\{\mathbf{x}_i, y_i\right\}_{i=1}^{N}$ is the training data set with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, $\boldsymbol{\theta} \in \mathbb{R}^d$ is the parameter to be determined, and $\lambda \in \mathbb{R}$ is the regularization coefficient. We plan to implement batch gradient descent algorithms to solve this problem, with a constant learning rate of α.

(a) (10 points) Write the batch gradient descent update at the k^{th} iteration, for solving this problem.

(b) (5 points) Given the dataset in Table 1 and $\lambda = 1$, find the minimizer $\boldsymbol{\theta}^*$ of the problem (1). Hint: Compute the gradient. It will be easier to use the matrix and vector forms of $\mathbf{x}_i$ and y_i.

i	$\mathbf{x}_i$	y_i
1	$(0,\ 1)^\top$	-1
2	$(-1,\ 1)^\top$	1

Table 1: Dataset for linear regression

(c) (10 points) Now consider you are provided with the data in Table 1. For this part let $\alpha = 1$ and $\lambda = 1$. By taking initial parameter $\boldsymbol{\theta}^0 = (0,\ 0)^\top$, find the parameters $\boldsymbol{\theta}^1$, $\boldsymbol{\theta}^2$ obtained after running two batch gradient descent updates.

(d) (5 points) For the batch gradient descent and the same setting in (c), does the loss $L(\boldsymbol{\theta})$ decrease at $\boldsymbol{\theta}^2$ compared with $\boldsymbol{\theta}^0$? If yes, calculate $L(\boldsymbol{\theta}^2) - L(\boldsymbol{\theta}^0)$. Verify if this error satisfies the theory of gradient descent we introduced in the class, that is,

$$L(\boldsymbol{\theta}^T) - L(\boldsymbol{\theta}^*) \leq \frac{2\beta\|\boldsymbol{\theta}^0 - \boldsymbol{\theta}^*\|^2}{T} \tag{2}$$

where β is the smoothness of the objective function $L(\boldsymbol{\theta})$. Hint: Find β by calculating the maximum eigenvalue of the Hessian matrix of $L(\boldsymbol{\theta})$.

Question 2: Kernel Methods (15 points)

Recall that a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a valid kernel function if it is **symmetric and positive semi-definite** function. For the current problem, we assume that the domain $\mathcal{X} = \mathbb{R}$.

(a) (10 points) Consider the function

$$K(x, x') = (xx' + 1)^{2021} \tag{3}$$

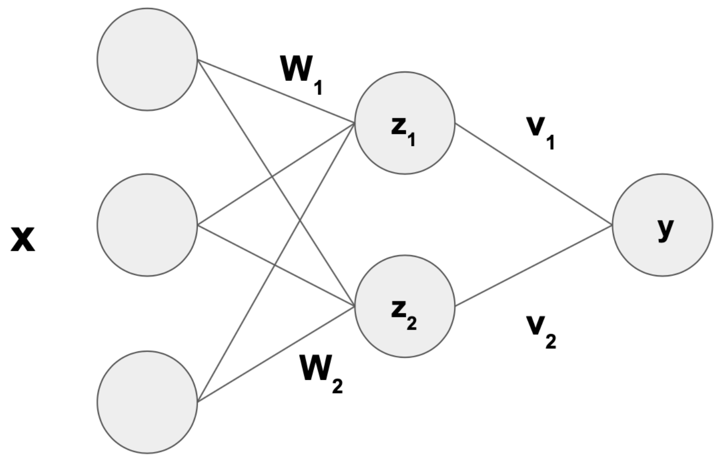
where $x, x' \in \mathbb{R}$. Show that K is a valid kernel function. Hint: Check Homework 5.

(b) (5 points) If K_1 is a valid kernel function and f is a function $f : \mathbb{R} \to \mathbb{R}$, check if the following function is a valid kernel function

$$K(x, x') = f(x)K_1(x, x')f(x'). \tag{4}$$

Question 3: Neural Networks (30 points)

Consider a one-hidden-layer neural network as shown below. The input $\mathbf{x} := [x^{(1)}, x^{(2)}, x^{(3)}]^\top \in \mathbb{R}^3$ is a vector with dimension 3, and the output $y \in \mathbb{R}$ is a scalar. The weight vector $\mathbf{w}_1 := [w_1^{(1)}, w_1^{(2)}, w_1^{(3)}]^\top \in \mathbb{R}^3$ concatenates the weights to z_1 and $\mathbf{w}_2 := [w_2^{(1)}, w_2^{(2)}, w_2^{(3)}] \in \mathbb{R}^3$ concatenates the weights to z_2. Only two neurons in the hidden layer have nonlinear activation functions σ.



(a) (10 points) Represent y as a function of $\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2, v_1, v_2$.

(b) (10 points) Initialize $\mathbf{w}_1 = \mathbf{0}$, $\mathbf{w}_2 = \mathbf{0}$, $v_1 = 0$ and $v_2 = 0$. We want to run backpropagation over the network for training. Assume the stepsize is $\eta = 1$. After 1 iteration of backpropagation, what are the values of $\mathbf{w}_1$? Hint: No need to know the values of feature $\mathbf{x}$.

(c) (5 points) After n iterations of backpropagation, what is the relationship between $\mathbf{w}_1$ and $\mathbf{w}_2$? Explain why. Hint: No heavy calculation is needed.

(d) (5 points) Is this an effective initialization of $\mathbf{w}_1, \mathbf{w}_2, v_1, v_2$ to optimize neural networks? Why?

Question 4: Constrained Optimization (25 points)

Consider the problem of solving a constrained quadratic optimization problem over $\mathbf{x} \in \mathbb{R}^d$ given as follows:

$$\min_{\mathbf{x}} \ \mathbf{x}^\top P \mathbf{x} + q^\top \mathbf{x} \quad \text{such that} \quad A\mathbf{x} = \mathbf{a}, \tag{5}$$

where $P \in \mathbb{R}^{d \times d}$ is positive definite, $q \in \mathbb{R}^d$, $A \in \mathbb{R}^{k \times d}$, $\mathbf{a} \in \mathbb{R}^k$.

(a) (10 points) What is the Lagrangian for the optimization problem? Clearly state the dimensionality and constraints on the Lagrange multiplier $\boldsymbol{\lambda}$ (if any) involved. Hint: Check the notes on SVM.

(b) (10 points) Can the Lagrange dual for the optimization problem be written as a closed form expression in terms of the Lagrange multipliers and the given matrices and vectors? If yes, clearly state the closed form for the Lagrange dual; if no, clearly explain why a closed form is not possible.

(c) (5 points) Clearly present an ADMM[1] algorithm for solving the optimization problem. In particular, say how additional variables need to be introduced, and then give the key steps for the algorithm. The key steps have to be presented as closed form, not as minimization over functions.

[1] Reading material: `https://www.stat.cmu.edu/~ryantibs/convexopt/lectures/admm.pdf`

Saaif Ahmed Exam 2

Problem 1

A:

$$\nabla_\theta = \frac{1}{2}\sum_{i=1}^{N} 2\left(\vec{\theta}^T\overrightarrow{x_i} - y_i\right) * (\overrightarrow{x_i}) + \frac{\lambda}{2} 2\left(\left|\left|\vec{\theta}\right|\right|\right) * \frac{\vec{\theta}}{||\vec{\theta}||}$$

$$= \frac{1}{2}\sum_{i=1}^{N} 2\left(\vec{\theta}^T\overrightarrow{x_i} - y_i\right) * (\overrightarrow{x_i}) + \lambda\vec{\theta}$$

Thus at some iteration k

$$\vec{\theta}^k = \vec{\theta}^{(k-1)} - \alpha\left(\frac{1}{2}\sum_{i=1}^{N} 2\left(\vec{\theta}^T\overrightarrow{x_i} - y_i\right) * (\overrightarrow{x_i}) + \lambda\vec{\theta}\right)$$

B:

$$L(\vec{\theta}) = \frac{1}{2}\left(X\vec{\theta} - \vec{y}\right)^T\left(X\vec{\theta} - \vec{y}\right) + \frac{\lambda}{2}\left|\left|\vec{\theta}\right|\right|^2$$

$$\nabla_{\vec{\theta}} = X^T X\vec{\theta} - X^T\vec{y} + \lambda\vec{\theta}$$

$$\vec{\theta}^* = (X^T X)^{-1}X^T\vec{y} + \lambda\vec{\theta}$$

$$X = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} ; y = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$\vec{\theta}^* = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}^{-1} * \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}^T * \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

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

$$= \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

$$\vec{\theta}^* = \begin{bmatrix} -2 \\ -1 \end{bmatrix}$$

Problem 1

C:

$$\vec{\theta}^{k} = \vec{\theta}^{(k-1)} - \alpha\left(\frac{1}{2}\sum_{i=1}^{N} 2\left(\vec{\theta}^{T}\vec{x_i} - y_i\right) * (\vec{x_i}) + \lambda\vec{\theta}\right)$$

$$\theta^1 = \begin{bmatrix}0\\0\end{bmatrix} - \left(\left(\begin{bmatrix}0\\0\end{bmatrix}^T \begin{bmatrix}0\\1\end{bmatrix} + 1\right) * \begin{bmatrix}0\\1\end{bmatrix} + \left(\begin{bmatrix}0\\0\end{bmatrix}^T \begin{bmatrix}-1\\1\end{bmatrix} - 1\right) * \begin{bmatrix}-1\\1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\right)$$

$$\theta^1 = \begin{bmatrix}0\\0\end{bmatrix} - \left(\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\-1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\right)$$

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

$$\theta^2 = \begin{bmatrix}-1\\0\end{bmatrix} - \left(\left(\begin{bmatrix}-1\\0\end{bmatrix}^T \begin{bmatrix}0\\1\end{bmatrix} + 1\right) * \begin{bmatrix}0\\1\end{bmatrix} + \left(\begin{bmatrix}-1\\0\end{bmatrix}^T \begin{bmatrix}-1\\1\end{bmatrix} - 1\right) * \begin{bmatrix}-1\\1\end{bmatrix} + \begin{bmatrix}-1\\0\end{bmatrix}\right)$$

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

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

D:

$$L(\vec{\theta}^2) = \frac{1}{2}\left(\left(\begin{bmatrix}0\\-1\end{bmatrix}^T \begin{bmatrix}0\\1\end{bmatrix} + 1\right)^2 + \left(\begin{bmatrix}0\\-1\end{bmatrix}^T \begin{bmatrix}-1\\1\end{bmatrix} - 1\right)^2\right) + \frac{1}{2}\left\|\begin{bmatrix}0\\-1\end{bmatrix}\right\|^2$$

$$= \frac{1}{2}(0+4) + \frac{1}{2}$$

$$= \frac{5}{2}$$

$$L(\vec{\theta}^0) = \frac{1}{2}\left(\left(\begin{bmatrix}0\\0\end{bmatrix}^T \begin{bmatrix}0\\1\end{bmatrix} + 1\right)^2 + \left(\begin{bmatrix}0\\0\end{bmatrix}^T \begin{bmatrix}-1\\1\end{bmatrix} - 1\right)^2\right)$$

$$= \frac{1}{2}(1+1)$$

$$= 1$$

In this case the loss of $L(\vec{\theta}^2) > L(\vec{\theta}^0)$. This is due to the learning rate being too large since $\alpha = 1$

With a smaller learning rate it the loss decreases. Such as $\alpha = 0.1$ which I have hard computed. using the code below

```
x1 = np.array([0,1])
x2 = np.array([-1,1])
y1=-1
y2=1
theta = np.array([-0.1,0])

new = theta - 0.1*((np.inner(theta,x1)-y1)+(np.inner(theta,x2)-y2)+theta)
print(new)

loss = 0.5 * ((np.inner(new,x1)-y1)**2 +(np.inner(new,x2)-y2)**2) + 0.5 * (np.linalg.norm(new))**2

print(loss)
```

$* \, Note: \vec{\theta}^1 = \begin{bmatrix}-\frac{1}{10}\\0\end{bmatrix}$ using the work from part c with $\alpha = 0.1$

Because the loss is not reduced after iterations it does not satisfy the theory of gradient descent

Problem 2:

A:

Proof of symmetry:

We have that $K(x,x') = (xx'+1)^{2021}$

And we have that $K(x',x) = (x'x+1)^{2021} = (xx'+1)^{2021}$

Thus symmetric

Proof of Positive Semi Definite

Define $\bar{K} = (xx'+1)$

Since $K = <\phi(x), \phi(x')>$

Where $\phi(x) = \begin{bmatrix} x \\ 1 \end{bmatrix}$; $\phi(x') = \begin{bmatrix} x' \\ 1 \end{bmatrix}$

We have that $\bar{K}$ is a valid kernel function by definition

We can build K^n by defining $K_{m+1} = K_m \odot \bar{K}$

We know that a kernel function created from the Hadamard product is a valid kernel function.

Thus by induction we can see that K_{m+1} is a valid kernel function

$K_1 = \bar{K}$ By induction we have that $K(x,x') = (xx'+1)^{2021}$ is a valid kernel function.

B:

Proof of symmetry

$K(x,x') = f(x)K(x,x')f(x')$

Consider

$K(x,x') = f(x')K(x',x)f(x) = f(x)K(x,x')f(x')$

Since $f(x) = R \rightarrow R$

Thus symmetric

Proof of PSD

$f(\vec{x})$ is a vector of $f(x)$ being applied to each component in $\vec{x}: \vec{x} \in R^n$

Consider $\vec{y}^T K \vec{y}: \vec{y} \in R^n$

$\vec{y}^T K \vec{y} = Tr\big(diag(\vec{y}) * K * diag(\vec{y})\big)$

$K = diag\left(f(\vec{x})\right) * K_1 * diag\left(f(\vec{x}')\right)$

$\vec{y}^T K \vec{y} = Tr\left(diag(\vec{y}) * diag\left(f(\vec{x})\right) * K_1 * diag\left(f(\vec{x}')\right) * diag(\vec{y})\right)$

K is PSD so consider square roots

$Tr\left(diag(\vec{y}) * diag\left(f(\vec{x})\right) * K_1^{\frac{1}{2}} * K_1^{\frac{1}{2}} * diag\left(f(\vec{x}')\right) * diag(\vec{y})\right)$

$Tr\left(K_1^{\frac{1}{2}} * diag(\vec{y}) * diag\left(f(\vec{x})\right) * K_1^{\frac{1}{2}} * diag\left(f(\vec{x}')\right) * diag(\vec{y})\right)$

$Tr(Q^T Q) \geq 0$

Where Tr denotes the matrix trace. Q is a PSD since Q is symmetric since K_1 is symmetric and all other diagonal matrices are symmetric.

Thus positive semi definite.

Problem 3

A:

$$z_1 = \sigma\left(\overrightarrow{w_1}^T \vec{x}\right)$$
$$z_2 = \sigma\left(\overrightarrow{w_2}^T \vec{x}\right)$$
$$y = v_1 z_1 + v_2 z_2$$
$$y = v_1 * \left(\sigma\left(\overrightarrow{w_1}^T \vec{x}\right)\right) + v_2 * \left(\sigma\left(\overrightarrow{w_2}^T \vec{x}\right)\right)$$

B:

LMS loss function update

$$W_1 = W + \eta(-\Delta j \vec{x})$$
$$= \begin{bmatrix}0\\0\end{bmatrix} + \ -\left(\sigma'\left(\overrightarrow{w_1}^T \vec{x} + \overrightarrow{w_2}^T \vec{x}\right) * \left(v_1 \sigma\left(\overrightarrow{w_1}^T \vec{x}\right)(y - \hat{y}) + v_2 \sigma\left(\overrightarrow{w_2}^T \vec{x}\right)(y - \hat{y})\right)\right) * \vec{x}$$
$$= \begin{bmatrix}0\\0\end{bmatrix} + \ \vec{x}$$
$$W_1 = (\vec{x})$$

After 1 iteration

C:

Looking at the update function with $v_1 = v_2 = 0$ We see that most of the $-\Delta j$ element goes away. Which leaves the update to be multiples of $\vec{x}$ as it is $-\Delta j \ * \vec{x}$

For example W_2 after 1 iteration would also be $(\vec{x})$ according to the above.

Thus the relationship is that $\overrightarrow{w_1} = \overrightarrow{w_2}$ after n iterations.

D:

This is a good way to initialize these parameters. This is because the intuition of update is that with positive error we increase the weights. We increase the weights each time by the respective feature in the $\vec{x}$ vector. It does require a better learning parameter. Since this update currently may be too drastic to begin converging.
A smaller η should be chose if the parameters are initialized this way.

Problem 4

A:

$L(\vec{x};\vec{\lambda},\vec{q},P)=\vec{x}^T P\vec{x}+\vec{q}^T\vec{x}+\vec{\lambda}^T(A\vec{x}-\vec{a})$

$\nabla_{\vec{x}}=2P\vec{x}+\vec{q}+A^T\vec{\lambda}=0$

$2P\vec{x}=-\left(\vec{q}+A^T\vec{\lambda}\right)$

$$\vec{x}=-\frac{\left(P^{-1}\left(\vec{q}+A^T\vec{\lambda}\right)\right)}{2}$$

$$\vec{x}=-\frac{1}{2}\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)$$

$D(\vec{\lambda})=\frac{1}{4}\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)P\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)-\frac{1}{2}\vec{q}^T\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)-\frac{1}{2}\vec{\lambda}^T A\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)-\vec{\lambda}\vec{a}$

$$D(\vec{\lambda})=\frac{1}{4}\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)\left(\vec{q}+A^T\vec{\lambda}\right)-\frac{1}{2}\vec{q}^T\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)-\frac{1}{2}\vec{\lambda}^T A\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)-\vec{\lambda}\vec{a}$$

$$D(\vec{\lambda})=\frac{1}{4}\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)\left(\left(\vec{q}+A^T\vec{\lambda}\right)-2\vec{q}^T-2\vec{\lambda}^T A\right)-\vec{\lambda}\vec{a}$$

The LaGrange multiplier $\vec{\lambda}$ is the $\arg\max_{\vec{\lambda}} D(\lambda)$ of which $D(\vec{\lambda})$ is shown above. $\vec{\lambda}\in \mathrm{R}^{\mathrm{k}}$

B:

A closed form is possible. The closed form is the $\min_{\vec{x}} L(\vec{x};\vec{\lambda})=D(\vec{\lambda})$

Which we have solved in the previous problem

$$D(\vec{\lambda})=\frac{1}{4}\left(P^{-1}\vec{q}+P^{-1}A^T\vec{\lambda}\right)\left(\left(\vec{q}+A^T\vec{\lambda}\right)-2\vec{q}^T-2\vec{\lambda}^T A\right)-\vec{\lambda}\vec{a}$$

C:

Given a data set of features $\{\overrightarrow{x_1},\overrightarrow{\mathrm{x}_2},\dots\}$ and labels $\vec{y}$

Initialize weights $\vec{w}$ and constant b

Parameters $\vec{\alpha}$

*Assume P exists for the algorithm

Take the Lagrangian Function $L(\vec{x};\vec{\lambda},\vec{q},P)$ and substitue in

$L(\vec{w},\vec{b};\vec{\alpha})=b^2\vec{w}^T P\vec{w}+\vec{\alpha}^T\vec{w}+\vec{\alpha}$

Now make the $D(\vec{\alpha})=\frac{1}{4}(P^{-1}\vec{\alpha})\left(||\vec{w}||\right)^2\left(||\vec{\alpha}||\right)^2-\vec{\alpha}$

Directly compute the slackness

- if $\alpha_i>0\rightarrow y_i(\vec{w}^T\overrightarrow{x_i}+b)=1$
  - $(\overrightarrow{x_i},y_i)$ is a support vector
- if $\alpha_i=0\rightarrow y_i(\vec{w}^T\overrightarrow{x_i}+b)>1$
  - $(\overrightarrow{x_i},y_i)$ is not a support vector