CS 475 Machine Learning: Homework 1
Supervised Classifiers 1
Analytical Problems

Due: Friday, February 21, 2020, 11:59 pm

55 Points Total Version 1.0

Li Xu (lxu41)

Instructions

We have provided this LaTeX document for turning in this homework. We give you one or more boxes to answer each question. The question to answer for each box will be noted in the title of the box.

Other than your name, do not type anything outside the boxes. Leave the rest of the document unchanged.

For written answers, replace the **\TextRequired** (**Place Answer Here**) command with your answer. For the following example *Question 0.1* , you would place your answer where **\TextRequired** (**Place Answer Here**) is located,

Place Answer Here

Do not change the height or title of the box. If your text goes beyond the box boundary, it will be cut off. We have given sufficient space for each answer, so please condense your answer if it overflows. The height of the box is an upper bound on the amount of text required to answer the question - many answers can be answered in a fraction of the space. Do not add text outside of the boxes. We will not read it.

For True/False or Multiple Choice questions, place your answers within the defined table. To mark the box(es) corresponding to your answers, replace **\Unchecked** (☐) commands with the **\Checked** (☒) command. Do not make any other changes to the table. For example, in *Question 0.2* ,

☒	Logistic Regression
☐	Perceptron

For answers that require a single equation, we will provide a specific type of box, such as in the following example *Question 0.3* . Please type the equation where **\EquationRequired** (**Type Equation Here**) without adding any $ signs or **\equation** commands. Do not put any additional text in this field.

$\mathbf{w} =$ **Type Equation Here**

For answers that require multiple equations, such as a derivation, place all equations within the specified box. You may include text short explanations if you wish (as shown in *Question 0.4*). You can put the equations in any format you like (e.g. within $ or $$, the **\equation** environment, the **\align** environment) as long as they stay within the box.

$x + 2$ x is a real number

the following equation uses the variable y

$y + 3$

Do not change any formatting in this document, or we may be unable to grade your work. This includes, but is not limited to, the height of textboxes, font sizes, and the spacing of text and tables. Additionally, do not add text outside of the answer boxes. Entering your answers are the only changes allowed.

We strongly recommend you review your answers in the generated PDF to ensure they appear correct. We will grade what appears in the answer boxes in the submitted PDF, NOT the original latex file.

Notation

$\mathbf{x_i}$ One input data vector. $\mathbf{x_i}$ is M dimensional. $\mathbf{x_i} \in \mathbb{R}^{1\times M}$.
We assume $\mathbf{x_i}$ is augmented with a 1 to include a bias term.

$\mathbf{X}$ A matrix of concatenated $\mathbf{x_i}$'s. There are N input vectors, so $\mathbf{X} \in \mathbb{R}^{N\times M}$

y_i The true label for input vector $\mathbf{x_i}$. In regression problems, y_i is a continuous value.
In general y_i can be a vector, but for now we assume y_i is a scalar. $y_i \in \mathbb{R}^1$.

$\mathbf{y}$ A vector of concatenated y_i's. There are N input vectors, so $\mathbf{y} \in \mathbb{R}^{N\times 1}$

$\mathbf{w}$ A weight vector. We are trying to learn the elements of $\mathbf{w}$.
$\mathbf{w}$ is the same number of elements as $\mathbf{x_i}$ because we will end up computing the dot product $\mathbf{x_i} \cdot \mathbf{w}$.
$\mathbf{w} \in \mathbb{R}^{M\times 1}$. We assume the bias term is included in $\mathbf{w}$.

$E_D(\mathbf{w})$ The loss due to the model fit.

$E_{\mathbf{w}}(\mathbf{w})$ The regularization term.

Notes: In general, a lowercase letter (not boldface), a, indicates a scalar.
A boldface lowercase letter, $\mathbf{a}$, indicates a vector.
A boldface uppercase letter, $\mathbf{A}$, indicates a matrix.

1) Regularization (25 points)

In class, we discussed adding a regularization penalty term to our objective function. This gives us an optimization problem of the following general form

$$\underset{\mathbf{w} \in \mathbb{R}^D}{\operatorname{argmin}} \; E_{\mathbf{D}}(\mathbf{w}) + \lambda \cdot E^q_{\mathbf{w}}(\mathbf{w}) \tag{1}$$

where $E_{\mathbf{D}}(\mathbf{w})$ is our usual loss function, $\lambda \geq 0$, and

$$E^q_{\mathbf{w}}(\mathbf{w}) \stackrel{\text{def}}{=} \sum_{j=1}^{M} |w_j|^q \tag{2}$$

where $q \geq 0$ is a hyper-parameter that can be experimented with. In class, we discussed $q = 2$ and $q = 1$ as they are the most common in practice.

(1) (8 points) Derive $\frac{d}{d\mathbf{w}} E^q_{\mathbf{w}}(\mathbf{w})$ for L1 and L2 regularization. You may assume all elements of $\mathbf{w}$ are nonzero.

L1 : $\frac{d}{d\mathbf{w}} E^q_{\mathbf{w}}(\mathbf{w}) = \frac{d}{d\mathbf{w}} |\mathbf{w}| = \text{sign}(\mathbf{w})$

L2 : $\frac{d}{d\mathbf{w}} E^q_{\mathbf{w}}(\mathbf{w}) = 2\mathbf{w}$

(2) (5 points) Using limits and the derivatives above, show why the L1 norm encourages sparser **w**'s than L2 norm regularization.

Sparser means more w_i=0 in $\mathbf{w}$
When the loss function $E_{\mathbf{D}}(\mathbf{w}) + \lambda \cdot E^q_{\mathbf{w}}(\mathbf{w})$ converges, $\frac{d}{dw} E_{\mathbf{D}}(\mathbf{w}) = 0$
For L1 norm, $\lim_{w \to 0} \frac{d}{d\mathbf{w}} \lambda |\mathbf{w}| = \text{sign } \lambda$,
update formula is w:= w - $\lambda \frac{d}{d\mathbf{w}}$
when $\lim_{w \to 0^+}$, $w := w - \lambda$;
else when $\lim_{w \to 0^-}$, $w := w + \lambda$,
as λ is positive, which makes w easy to turn to 0

However,for L2 norm, $\lim_{x \to 0} \frac{d}{d\mathbf{w}} \lambda |\mathbf{w}| = 2\lambda \cdot \mathbf{w}$
w:= w + $2\lambda \cdot w$, which makes the sign of w always the same, meaning impossible to turn w to 0.

(3) (3 points) Which norm (L1 or L2) has larger penalties for large weights?

L2. Larger absolute value on regularization loss.

(4) (6 points) Suppose you are a Data Scientist working in a nuclear lab where a spring moves a critical component.[1] Spring movement can be modeled with a second order differential equation:

$$\frac{d^2x}{dt^2} = c + b\frac{dx}{dt} + ax$$

Based on the material of the spring, you know $c \approx 1$, $b \approx 5$, and $a \approx 10$. Because of the dangerous nature of the nuclear experiment, your advisor asked you to record motion data of the spring and to learn the exact coefficients for your spring. Using $\mathbf{x_i} = [1, \frac{dx_i}{dt}, x_i]$, we want to learn $\mathbf{w} = [w_1, w_2, w_3]$ so that $\frac{d^2x}{dt^2} = \mathbf{x_i w}$.

Write a regularization term that reflects your prior knowledge of the spring.
Note: you do not need to understand differential equations to answer this question!

$(w_1 - 1)^2 + (w_2 - 5)^2 + (w_3 - 10)^2$

(5) (3 points) In lecture, we learned that larger weights can indicate overfitting. Why won't our regularization scheme in (4) cause our model to overfit? (Explain in words.)

What I add to the loss function is regularization term without prior information on train data. Adding difference means that $\mathbf{w}$ is lower to zero when calculate the regularization, which prevents overfitting.

[1] This is a fictitious example. We are in no way endorsing springs in nuclear experiments or claiming this is a reasonable way to run nuclear labs.

2) Maximizing Likelihood vs Minimizing the Loss (15 points)

In a statistical view of linear regression, $y_i = \mathbf{x_i w} + \upsilon$, $\upsilon \sim \mathcal{N}(\upsilon; 0, \sigma^2)$, where υ accounts for noise not captured by $\mathbf{x_i w}$. We can define the label y_i as a random variable such that

$$p(y_i|\mathbf{x_i}; \mathbf{w}, \sigma) = \mathcal{N}(y_i; \mathbf{x_i w}, \sigma^2)$$

(1) (10 points) Demonstrate that maximizing the log-likelihood of $\mathbf{w}$ given the observed data $\mathbf{y}$ and $\mathbf{X}$ is equivalent to minimizing the sum of squared prediction error. That is, show

$$\arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y_i|\mathbf{x_i}; \mathbf{w}, \sigma)) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{x_i w})^2$$

Hint: You can show this by deriving the closed form solution for $\mathbf{w}$ *for each of the two formulations.*

$\arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y_i|\mathbf{x_i}; \mathbf{w}, \sigma))$
$= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-(y_i - x_i w)^2}{2\sigma^2})$
$= \sum_{i=1}^{N} \log(2\pi\sigma)^{\frac{-1}{2}} + \sum_{i=1}^{N} \log\exp(\frac{-(y_i - x_i w)^2}{2\sigma^2})$
$= \sum_{i=1}^{N} \frac{-(y_i - x_i w)^2}{2\sigma^2} + \frac{N}{2} \log(2\pi\sigma^2)$
$= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i w)^2 + \frac{N}{2} \log(2\pi\sigma^2)$
$\because \frac{N}{2} \log(2\pi\sigma^2)$ and $-\frac{1}{2\sigma^2}$ is constant,
$\therefore \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log(p(y_i|\mathbf{x_i}; \mathbf{w}, \sigma)) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - \mathbf{x_i w})^2$

(2) (5 points) Suppose we now use the loss function $\sum_{i=1}^{N} |y_i - \mathbf{x_i w}|$. Does a closed-form solution exist for $\mathbf{w}$? If it does, derive it. If it does not, explain why. What are advantages of minimizing this loss function for our model over the usual Sum of Squared Error function above?

It does not.
$\because$ Loss function is not derivative when $|y_i - x_i w| = 0$.
Sum of Squared Error function is $\sum_{i=1}^{N}(y - x_i w)^2$ which is derivative.
When the loss function is derivative, gradient decent is possible, $\mathbf{w}$ can be updated to make loss function converge.

3) Support Vector Machines (10 points)

In Support Vector Machines (SVMs) the margin γ_i for input $\mathbf{x_i}$ is the distance between $\mathbf{x_i}$ and the decision boundary.

$$\gamma_i = y_i(\mathbf{x_i w})$$

Where $y_i \in \{-1, 1\}$

(1) (7 points) Prove that the margin cannot be negative if your training data is linearly separable.

Linear separable means that
$\forall(y_i, \mathbf{x_i})$, if $y_i > 0$, $(\mathbf{x_i w}) > 0$;
if $y_i < 0$, $(\mathbf{x_i w}) < 0$
so that $\gamma_i > 0$
If $\exists(y_i, \mathbf{x_i})$, $\gamma_i = y_i(\mathbf{x_i w}) < 0$, data is not linearly separable.

There is one special case that
$\forall y_i > 0$, $\mathbf{x_i w} < 0$;
$\forall y_i < 0$, $\mathbf{x_i w} < 0$
if we reverse the sign of $\mathbf{w}$, all the margin turns to positive that the training data is linearly separable.

(2) (3 points) What would a negative margin indicate? In what scenario would some margins be negative? How do we address this? (Explain in words.)

When $y_i < 0$, $\mathbf{x_i w} > 0$ or When $y_i > 0$, $\mathbf{x_i w} < 0$ We need to consider such cases to add to loss function $\sum_{i=1}^{N} L_w(y_i) = max(0, -y_i \mathbf{x_i w})$
when negative margin happens, $-y_i \mathbf{x_i w} > 0$, $\sum_{i=1}^{N} L_w(y_i) > 0$ then do gradient decent $\frac{\delta L_w}{\delta \mathbf{w}}$ to update a better $\mathbf{w}$ for margin.

Add punishment on that or kernel function mentioned on lecture4/5 to get a soft margin.

4) Sensitivity vs Specificity (5 points)

A binary label has two possible values, $y \in \{0, 1\}$, for the output.[2] $y = 0$ is called a negative example, and $y = 1$ is called a positive example. A binary classifier has four possible outcomes: True Positive (TP): the true and predicted label are both positive, False Positive (FP): the true label is negative but the predicted label is positive, True Negative (TN): the true and predicted label are both negative, and False Negative: the true label is positive but the predicted label is negative. Accuracy measures the fraction of True Positives and True Negatives over all the predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

In some cases, it is more important to measure False Positives or False Negatives than the Accuracy. Sensitivity and Specificity are two metrics used in these cases. Sensitivity measures the fraction of positive elements that are detected.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

Specificity measures the fraction of negative elements that are correctly detected.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

In your programming assignment, you will experiment with training binary classifiers that weight positive and negative examples differently in training.

1. If you weight positive examples more highly in training than negative examples, do you expect a higher Sensitivity or Specificity? What is one real world scenario where this kind of classifier would be useful?

2. Conversely, if you weight negative examples more highly in training than positive examples, do you expect a higher sensitivity or specificity? What is one real world scenario where this kind of classifier would be useful?

(a).Higher sensitivity. In scenario where we don't want to miss the detection of true positive like malignant tumour detection.
(b).Higher specificity. In scenario where we don't want to misdiagnosis like some benign tumour.

[2] or $y \in \{-1, 1\}$