Winter 2026 DSC 240: Introduction to Machine Learning
Homework 2

Due: **Monday, Feb 9th, 11:59 pm PST**

Notes:

1. **This assignment is to be done individually.** You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the question, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.

2. Be aware of the late policy in the course syllabus – i.e., *no late days for all the assignments,* so **it is your responsibility to turn in your assignment to Gradescope by the due time.**

3. Justify every answer you give – **show the work** that achieves the answer or **explain** your response.

4. Any updates or corrections will be posted on Piazza, so check there occasionally.

5. It is recommended to typeset the homework in LaTeX or markdown for the submission. However, neatly, hand-written homework are also allowed.

6. This assignment requires **two parts of the submission**. Both of these submissions will be available on Gradescope, and **both submissions are MANDATORY** for receiving full credit for the assignment:

   (a) Written form of the submission as a **PDF** file format. This needs to be submitted in **Homework 2 - Report** on Gradescope.

   (b) Code submission as a **.py** python script or a **.ipynb** jupyter notebook file. This needs to be submitted in **Homework 2 - Code** on Gradescope.

7. You can complete the assignment in a Jupyter notebook, which allows you to write both the text (using LaTeX for math) and the code in one file. Once done, convert the notebook to a PDF and submit it to the **Homework 2 - Report** on Gradescope. Also, submit the .ipynb notebook file (with the code) to the **Homework 2 - Code** on Gradescope. This is just a suggestion – you can also submit the report using LaTeX/Markdown/handwriting and the code in a separate .py/.ipynb file if you prefer.

8. Be sure to re-read the **"Academic Integrity"** on the course syllabus. You must complete the section below. If you answered Yes to either of the following two questions, give corresponding full details.

Did you receive any help whatsoever from anyone in solving this assignment?
Did you give any help whatsoever to anyone in solving this assignment?

Note: Please upload your written part as a PDF, and please upload your code as a separate file in (".py" or ".ipynb") form. Both files should be uploaded to Gradescope.

1. **(15 points)** The Hinge loss for a linear classifier is $\ell(w^T x, y) := \max\{0, 1 - yw^T x\}$. The shifted Hinge loss (aka Perceptron loss) for a linear classifier is $\ell(w^T x, y) := \max\{0, -yw^T x\}$.

   (a) Calculate the gradient of hinge loss with respect to w.
   (you can ignore the case when $w^T x = 1$)

   (b) Calculate the gradient of Perceptron loss w.r.t w
   (You may ignore the case when $w^T x = 0$)

   (c) What is the main difference in the SGD algorithm with the two losses above?

2. **(20 points)** Figure 1 shows training data with two features, with each example labeled as being in the positive (filled-in points) or negative (open points) class. Proposed linear classifier functions (C1 through C6) are shown as dotted lines, each one indicating a different classifier for this data. Each classifier classifies points to the upper-right of its dotted line as positive and points to the lower-left of its dotted line as negative.

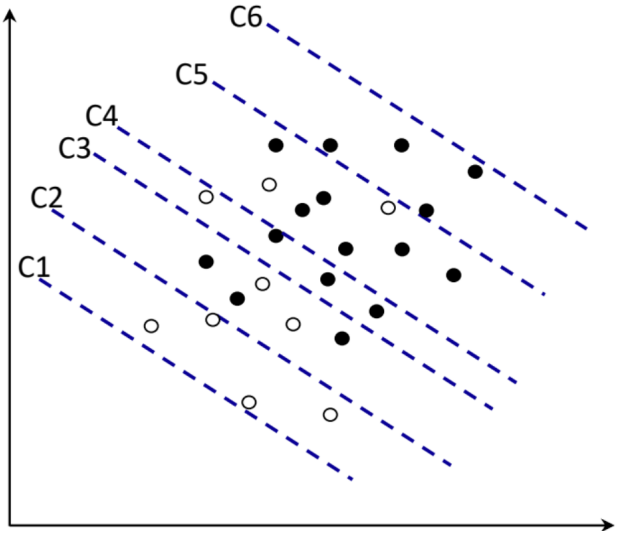

Figure 1: Training data with two features, with each example labeled as being in the positive (filled-in points) or negative (open points) class.

   (a) Draw the ROC plot and label the classifiers on the plot.

   (b) Which classifiers have the highest and lowest accuracy?

   (c) Which classifiers have the highest and lowest precision?

   (d) What are F1 scores for classifiers C1-C6?

3. **(15 points)** The decision boundary for a binary classifier that takes an input with three features is the plane defined by the equation $3x_1 + 2x_2 + 4x_3 = 18$, where the features are x_1, x_2 and x_3. The plane separates the positive predictions ($3x_1 + 2x_2 + 4x_3 \geq 18$) from the negative predictions ($3x_1 + 2x_2 + 4x_3 < 18$).

   A dataset set S consists of the following labeled examples:

- $(2, 2, 3)+$ (positive class)
- $(3, 3, 2)+$
- $(1, 2, 3)+$
- $(1, 4, 1)+$
- $(4, 4, 4)+$
- $(2, 2, 2)+$
- $(3, 3, 1)-$ (negative class)
- $(1, 1, 1)-$
- $(3, 2, 2)-$
- $(0, 4, 2)-$
- $(4, 0, 0)-$
- $(0, 0, 3)-$

For each data point, provide:

(a) Its margin (distance from the decision boundary)
(b) Its 0-1 loss
(c) Its hinge loss
(d) Its squared loss

(You may either calculate these quantities manually or write code to compute them.)

4. **(10 points)** Recall the objective function for linear regression can be expressed as

$$E(\mathbf{w}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2.$$

Minimizing this function with respect to $\mathbf{w}$ leads to the optimal $\mathbf{w}^*$ as $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. This solution holds only when $\mathbf{X}^T\mathbf{X}$ is nonsingular (i.e., full rank or invertible). To overcome this problem, the following objective function is commonly minimized instead:

$$E_2(\mathbf{w}) = \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2,$$

where $\lambda > 0$ is a user-specified parameter. Please do the following:

(a) (5 points) Derive the optimal $\mathbf{w}^*$ that minimizes $E_2(\mathbf{w})$.
(b) (5 points) Explain how this new objective function can overcome the singularity problem of $\mathbf{X}^T\mathbf{X}$. [Hint: Think about positive definite and positive semi-definite definitions.]

5. **(20 points)** In a linear regression task, implement a gradient solver and a stochastic gradient solver for (a) least squares regression (lsqr) and (b) ridge regression (least squares with ℓ_2-norm regularization) the given dataset files `data_X_Q5Q6.npy` and `data_y_Q5Q6.npy`. For both lsqr and ridge, compare the result models from the two optimizers (gradient descent and stochastic gradient descent) and the closed-form solutions (where we set gradient to zero and solved the equation), using the ℓ_2-norm distance to measure the difference between the two solutions. Please refer to the **Appendix** section for instructions on how to load the dataset files in python.

Deliverables:

Code: Your implementation of gradient solver, stochastic gradient solver, and closed form solver. Your code then load the given dataset(data_*.npy), and your code that computes solutions using 3 solvers. Finally, your code that compares weights and reports ℓ_2-norm distance for each pair. Repeat this whole process with and without regularization.

PDF: Result of the ℓ_2-norm distance of weights between 3 comparisons: GD vs SGD, GD vs closed-form, and SGD vs closed-form. Repeat this whole process with and without regularization.

Hyper-parameters and Dataset: Load the data to use in the questions from the directory. For regularization parameters, we suggest $1e-6$.

6. **(10 points)** Use the given dataset files `data_X_Q5Q6.npy` and `data_y_Q5Q6.npy` to compute the objective functions (lsqr and ridge, respectively). Plot the **learning curve** (x-axis: iteration number, y-axis: objective function). Note that when computing the objective function, $E(w(t))$, you should use all the data points, for both gradient descent and stochastic gradient descent. Please refer to the **Appendix** section for instructions on how to load the dataset files in python.
Deliverables:

   Code: Using previous implemented solvers and given datasets to generate the curves (with and without regularization; gradient descent and stochastic gradient descent).

   PDF: The result as curves.

7. **(10 points)** Previously, stochastic gradient descent we introduced uses 1 data point in each iteration to estimate the gradient. In fact, the stochastic gradient descent can use a batch of data points to estimate/compute the gradient to accelerate the convergence speed. Implement batched version gradient descent for ridge regression. For this question, use the corresponding dataset files `data_X_Q7.npy` and `data_y_Q7.npy`. Plot the learning curve (again as in Q6, this is objective function against number of iterations) of stochastic gradient descent with batch size of $[1, 10, 100]$. Please refer to the **Appendix** section for instructions on how to load the dataset files in python.
Deliverables:

   Code: Stochastic gradient descent with different batch sizes.

   PDF: 3 Convergence graphs with different batch sizes.

8. (**Bonus 12 points**) Consider the hat matrix $\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$, where $\mathbf{X}$ is an N by $d+1$ matrix, and $\mathbf{X}^T\mathbf{X}$ is invertible.

   (a) (3 points) Show that $\mathbf{H}$ is symmetric.

   (b) (3 points) Show that $\mathbf{H}^K = \mathbf{H}$ for any positive integer K.

   (c) (3 points) If $\mathbf{I}$ is the identity matrix of size N, show that $(\mathbf{I}-\mathbf{H})^K = \mathbf{I}-\mathbf{H}$ for any positive integer K.

   (d) (3 points) Show that $\text{trace}(\mathbf{H}) = d+1$, where the trace is the sum of diagonal elements. [Hint: $\text{trace}(\mathbf{AB}) = \text{trace}(\mathbf{BA})$ for any matrices A, B of compatible sizes]

Appendix

To load a `.npy` file (for e.g., `data_X_Q5Q6.npy`) in Python, you can use the following lines of code:

```
import numpy as np
X = np.load("data_X_Q5Q6.npy")
```