# CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, October 6, 2023, submit on LEARN (written) and Crowdmark (code).
Include your name and student number!

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

## Exercise 1: Perceptron Questions (5 pts)

1. (3 pts) The perceptron algorithm makes an update every time it witnesses a mistake. What if it makes an update on every point, regardless of whether or not that point is correctly classified? For simplicity, consider a setting where where $b$ is fixed to 0. Give an example of an infinite sequence of points $(x_i, y_i)$ with the following properties:

   - The sequence is strictly linearly separable with $b = 0$ (i.e., the margin is some constant $\gamma > 0$),
   - The sequence has $\max \|x_i\|_2 \leq 1$,
   - The modified perceptron algorithm makes an infinite number of mistakes on this sequence.

   Prove that it has these properties. Note that the perceptron convergence theorem and the first two conditions imply that, at some point, the unmodified perceptron algorithm would only make a finite number of mistakes.

   Ans:

2. (1 pt) Give examples of where the perceptron algorithm (the original algorithm studied in class, not the one from the previous part) converges to an arbitrarily-small margin halfspace, and a separate example where it converges to a maximum margin halfspace. To word the former more precisely: for any $0 < \varepsilon < 1/2$, give a dataset (with margin at least 1) and a order in which to process the points such that the perceptron algorithm halts providing a halfspace with margin $\leq \varepsilon$.

   Ans:

3. (1 pt) Suppose that in each iteration of the perceptron algorithm (the original algorithm studied in class), a point is chosen uniformly at random from the dataset. Show how the perceptron algorithm can be viewed as an instantiation of stochastic gradient descent (SGD). In particular, you must provide a loss function and a learning rate such that SGD with these parameters and perceptron are identical.

   Ans:

## Exercise 2: Regression Implementation (8 pts)

Recall that ridge regression refers to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \overbrace{\underbrace{\frac{1}{2n}\|X\mathbf{w} + b\mathbf{1} - \mathbf{y}\|_2^2}_{\text{error}} + \lambda\|\mathbf{w}\|_2^2}^{\text{loss}}, \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ are the given dataset and $\lambda \geq 0$ is the regularization hyperparameter. If $\lambda = 0$, then this is the standard linear regression problem. Observe the distinction between the error (which does not include the regularization term) and the loss (which does).

1. (1 pt) Show that ridge regression can be rewritten as a non-regularized linear regression problem. That is, prove 1 is equivalent to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\| \begin{bmatrix} X & \mathbf{1}_n \\ \sqrt{2\lambda n} I_d & \mathbf{0}_d \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} - \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_d \end{bmatrix} \right\|_2^2, \tag{2}$$

where $I_d$ is the $d$-dimensional identity matrix, and $\mathbf{0}_k$ and $\mathbf{1}_k$ are zero and one column vectors in $k$ dimensions.

Ans:

2. (1 pt) Show that the derivatives of 1 are

$$\frac{\partial}{\partial \mathbf{w}} = \tfrac{1}{n} X^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}) + 2\lambda \mathbf{w} \tag{3}$$

$$\frac{\partial}{\partial b} = \tfrac{1}{n} \mathbf{1}^\top (X\mathbf{w} + b\mathbf{1} - \mathbf{y}). \tag{4}$$

Ans:

3. (2 pts) Implement ridge regression using the closed form solution for linear regression as derived in lecture. You may find the function `numpy.linalg.solve` useful.

Ans:

4. (2 pts) Implement the gradient descent algorithm for solving ridge regression. The following incomplete pseudo-code may of help. Your training loss should monotonically decrease during iteration; if not try to tune your step size $\eta$, e.g. make it smaller.

Ans:

**Algorithm 1:** Gradient descent for ridge regression.

**Input:** $X \in \mathbb{R}^{n \times d}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{w}_0 = \mathbf{0}_d$, $b_0 = 0$, $\mathsf{max\_pass} \in \mathbb{N}$, $\eta > 0$, $\mathsf{tol} > 0$
**Output:** $\mathbf{w}, b$

```
for t = 1, 2, ..., max_pass do
    w_t ←
    b_t ←
    if ||w_t − w_{t−1}|| ≤ tol then                    // can use other stopping criteria
        break
w ← w_t,  b ← b_t
```

Ans:

5. (3 pts) Test your two implementations on the Boston housing dataset (to predict the median house price, i.e., $y$). Use the train and test splits provided on course website. Try $\lambda \in \{0, 10\}$ and report your training error, training loss and test error for each. You may have trouble getting gradient descent to work if you don't standardize your data (you may not use sklearn to standardize your data, do it yourself). Note that you would have to standardize both your training and test features. (Try gradient descent both without and with standardizing the data and see how it differs.) For the gradient descent algorithm, plot the training loss over iterations. Compare the running time of the two approaches, which is faster? Overall, which approach do you think is better? Explain why.

Ans:

**Exercise 3: Playing with Regression (3 pts)**

You may use the Python package scikit-learn for this exercise (and only for this exercise).

Train (unregularized) linear regression, ridge regression, and lasso on the mystery datasets A, B, and C on the course website (using X_train and Y_train for each dataset). For ridge regression and lasso, use k-fold cross validation to determine appropriate choices of the hyperparameters (you may NOT use functions like RidgeCV and LassoCV, you must implement it yourself). Report the average mean squared error on the test set for each method, as well as the selected regularization parameters. Which approach performs best in each case? For each dataset (A, B, C), do the following: Create a histogram where the x-axis is divided into bins corresponding to the values of coordinates of the parameter vector, and the y-axis is the number of coordinates of the parameter vector which fall into each bin. Plot the three parameter vectors (one for unregularized, one for ridge, one for lasso) on the same histogram, so they're all visible at once (change the opacity setting for

the bars if necessary). [There should be three total histograms, each with three parameter vectors overlayed.]

Ans:

**Exercise 4: Nearest Neighbour Regression (7 pts)**

1. (3 pts) Implement $k$-nearest neighbour regression, for a dataset of $n$ $X, y$ pairs where $X \in \mathbb{R}^d$ and $y \in \mathbb{R}$. This is similar to $k$-nearest neighbour classification, but instead of aggregating labels by taking the majority, we average the $y$ values of the $k$ nearest neighbours. Use $\ell_2$ distance as the distance metric, that is, the distance between points $X_1$ and $X_2$ is $\|X_1 - X_2\|_2$. Ensure that your implementation is $O(nd)$ time for all values of $k$, and argue that this is the case.

   Ans:

2. (2 pts) For the training sets of the $d = 1$ mystery datasets D and E on the course website, compute a) the (unregularized) least squares linear regression solution and b) the $k$-nearest neighbour regression solution with each integer $k$ from 1 to 9. For each dataset, on one plot with the $x$ and $y$ axes corresponding to the $x$ and $y$ values, display the least squares solution, the 1-nearest neighbour solution, and the 9-nearest neighbour solution. Be sure to plot these solutions for all points between the minimum and maximum $x$ values, not just for the points in the dataset. On another plot, with $k$ on the $x$ axis and test mean-squared error on the $y$ axis, display the error of the $k$-NN solution for each value of $k$. On the same plot, include the test error of the least squares solution as a horizontal line. When does each approach perform better, and why?

   Ans:

3. (2 pts) For the training set of the $d = 20$ mystery dataset F on the course website, compute a) the (unregularized) least squares linear regression solution and b) the $k$-nearest neighbour regression solution with each integer $k$ from 1 to 9. Plot, with $k$ on the $x$ axis and test mean-squared error on the $y$ axis, display the error of the $k$-NN solution for each value of $k$. On the same plot, include the test error of the least squares solution as a horizontal line. Which approach performs better, and why? Hint: consider inspecting distances between the test points and their $k$ nearest neighbours in the training set.

   Ans:

**Exercise 5: Poisson Regression (4 pts)**

Recall that in logistic regression we assumed the *binary* label $Y_i \in \{0, 1\}$ follows the Bernoulli distribution: $\Pr(Y_i = 1|X_i) = p_i$, where $p_i$ also happens to be the mean. Under the independence assumption we derived the log-likelihood function:

$$\sum_{i=1}^{n} (1 - y_i) \log(1 - p_i) + y_i \log(p_i). \tag{5}$$

Then, we parameterized the mean parameter $p_i$ through the logit transform:

$$\log \frac{p_i}{1 - p_i} = \mathbf{w}^\top \mathbf{x}_i + b, \quad \text{or equivalently} \quad p_i = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}_i - b)}. \tag{6}$$

Lastly, we found the weight vector $\mathbf{w}$ and $b$ by maximizing the log-likelihood function.

In the following we generalize the above idea to the case where $Y_i \in \mathbb{N}$, i.e., $Y_i$ can take any natural number (for instance, when we are interested in predicting the number of customers or network packages).

1. (1 pt) Naturally, we assume $Y_i \in \mathbb{N}$ follows the Poisson distribution (with mean $\mu_i \geq 0$):

$$\Pr(Y_i = k|X_i) = \frac{\mu_i^k}{k!} \exp(-\mu_i), \quad k = 0, 1, 2, \ldots. \tag{7}$$

   Given a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$, what is the log-likelihood function (of $\mu_i$'s) given $\mathcal{D}$?

   Ans:

2. (1 pt) Can you give some justification of the parameterization below?

$$\log \mu_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{8}$$

   Ans:

3. (1 pt) Based on the above, write down the objective function for Poisson regression. Please specify the optimization variables and whether you are maximizing or minimizing. [Constants can be dropped.]

   Ans:

4. (1 pt) Compute the gradient of your objective function above and formulate a gradient algorithm for finding the weight vector $\mathbf{w}$ and $b$.

   Ans: