CS589: Machine Learning - Fall 2025

Homework 2: Learning, Capacity Control and Multi Layer Perceptrons

Assigned: Thursday, September 18. Due: Wednesday, September 24 at 8:00pm

Getting Started: This assignment consists of written problems and coding problems. Download the assignment archive from Canvas and unzip the file. Starter code for coding problems is provided in the code directory. For general clarification questions, please submit public posts to Campuswire. For questions related to your specific solutions, please submit private posts on Campuswire. In-person help is available through regularly scheduled office hours.

Due Date and Late Work: The assignment is due at 8:00pm ET on September 24th, 2025. Students can submit up to 11:59pm on the due date with no penalty. Work submitted up to 11:59pm on one day after the assignment is due is subject to a penalty of 10%. Work submitted up to 11:59pm two days after the assignment is due is subject to a penalty of 20%. Gradescope will close at 11:59pm two days after the assignment is due and work can not be submitted for credit after this point.

How to Submit: Your written report must be submitted to Gradescope as a PDF file. The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You must select the page on which each answer appears when uploading your report to Gradescope. You are encouraged to typeset your PDF solutions using LaTeX. The source of this assignment is provided to help you get started. You may also submit a PDF containing scans of *clear* hand-written solutions. Work that is illegible will not count for credit. For this assignment, code should be submitted to Gradescope as Python 3.10+ Jupyter Notebooks. Autograding will not be used for this assignment. You may submit both the code and the report as many times as you like before Gradescope closes. Only your final submission will be graded. Any late penalties will be based on the timestamp of your final submission as determined by Gradescope.

Academic Honesty Reminder: Homework assignments are individual work. Being in possession of another student's solutions, code, code output, or plots/graphs for any reason is considered cheating. Sharing your solutions, code, code output, or plots/graphs with other students for any reason is considered cheating. Copying solutions from external sources (books, web pages, etc.) is considered cheating. Collaboration indistinguishable from copying is considered cheating. Posting your code to public repositories like GitHub (during or after the course) is not allowed. Manual and algorithmic cheating detection are used in this class. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Generative AI Use Reminder: The use of generative AI tools to help with coding is permitted for programming problems in this course. The use of generative AI for all other work is considered cheating.

1. (*10 points*) **Generalization and Model Capacity.** The following questions concern generalization and model capacity.

a. (*5 pts*) Explain what the OOD problem is and how it affects the ability to accurately estimate the performance of machine learning models on future data.

b. (*5 pts*) Suppose we train a binary logistic regression model and find that the training error is much lower than the test error. Explain what problem has occurred and suggest one approach that could fix the issue.

2. (*30 points*) **Learning Probability Mass Functions**. Consider a binary random variable X taking values in $\mathcal{X} = \{0, 1\}$. Let the probability mass function (PMF) for X be given by $P(X = 0) = \pi_0$ and $P(X = 1) = \pi_1$. This PMF is subject to a non-negativity constraint $\pi_x \geq 0$ for all $x \in \mathcal{X}$ and a normalization constraint $\sum_{x \in \mathcal{X}} \pi_x = 1$. In this question, we will explore learning the parameters π_x for this simple model.

a. (*2 pts*) When representing the probability of different data cases x_n using the model, it is more convenient to express the PMF as $P(X = x_n) = \pi_0^{\mathbb{I}(x_n=0)} \cdot \pi_1^{\mathbb{I}(x_n=1)}$. To begin, show that this alternative expression reduces to $P(X = x_n) = \pi_{x_n}$ for any $x_n \in \mathcal{X}$.

b. (*5 pts*) Suppose we have a data set $\mathcal{D}_{tr} = \{x_1, ..., x_{N_{tr}}\}$ with $x_n \in \mathcal{X}$. We can learn the model by minimizing the negative average log likelihood function shown below. Using the alternative formulation of $P(X = x_n)$ from part (a), expand the $nall(\pi, \mathcal{D})$ function in terms of the π_x parameters. Show your work. Make sure to simplify the log and exponent terms.

$$nall(\pi, \mathcal{D}) = -\frac{1}{N_{tr}} \sum_{n=1}^{N_{tr}} \log P(X = x_n)$$

c. (*3 pts*) Since the π_x parameters are subject to non-negativity and normalization constraints, to learn the parameters, we need to deal with the constraints. We can eliminate the sum-to-one constraint by changing to a new parameterization where there is a single parameter ϕ. We set $\pi_1 = \phi$ and $\pi_0 = 1 - \phi$. What constraints is the new parameter ϕ subject to? Explain your answer.

d. (*5 pts*) Continuing from part (c), re-write the negative average log likelihood function in terms of the new parameter ϕ to obtain $nall(\phi, \mathcal{D})$. Show your work.

e. (*5 pts*) We will now solve the learning problem from part (d) by ignoring the constraints, finding the analytic minimizer of $nall(\phi, \mathcal{D})$, and checking if the resulting optimal value satisfies the constraints. As a first step, find $\frac{d}{d\phi} nall(\phi, \mathcal{D})$, the derivative of $nall(\phi, \mathcal{D})$ with respect to ϕ. Show your work.

f. (*5 pts*) Next, solve the stationary equation $\frac{d}{d\phi} nall(\phi, \mathcal{D}) = 0$ to obtain the estimator for ϕ. The result will be an equation relating the training data to the optimal value of ϕ. Show your work.

g. (*5 pts*) Lastly, show that the formula that you found for the optimal value of ϕ in the previous part results in values of π_0 and π_1 that satisfy non-negativity and normalization for any data set $\mathcal{D}_{tr}$.

3. (*10 points*) **Learning for Multi Class Logistic Regression:** Consider the multi-class logistic regression model defined below where $c \in \{1, ..., C\}$ and $\mathbf{x} \in \mathbb{R}^D$. This model has a weight vector $\mathbf{w}_c \in \mathbb{R}^D$ and a bias $b_c \in \mathbb{R}$ for each class c as its parameters. We let θ denote the complete set of parameters.

$$P(Y = c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x} + b_c)}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{x} + b_{c'})}$$

a. (*5 pts*) This model is learned using the negative average log likelihood function shown below given a data set $\mathcal{D}_{tr}$. Using the definition of the model given above, expand and simplify the objective function $nall(\theta, \mathcal{D}_{tr})$ in terms of the model parameters $\mathbf{w}_c$, b_c and the data. Show your work.

$$nall(\theta, \mathcal{D}_{tr}) = -\frac{1}{N_{tr}} \sum_{n=1}^{N_{tr}} \log P(Y = y_n | \mathbf{X} = \mathbf{x}_n)$$

b. (*5 pts*) The key ingredient in learning for multi-class logistic regression is the computation of the gradient with respect to the parameters for use in numerical optimization. Derive the gradient of $nall(\theta, \mathcal{D}_{tr})$ with respect to the weight vector $\mathbf{w}_1$. Show your work and simplify the result as much as possible.

4. (*50 points*) **Computational Agriculture:** In this problem, you are going to experiment with KNN and MLP classifiers on a problem from the domain of computational agriculture. Specifically, classifying rice grains into their species based on features derived from images of the grains. For this problem, you will use the data set described here: `https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik`. The code template for this problem is a Jupyter Notebook (`experiments.ipynb`) located in the code directory of the assignment package. You can run the notebook locally by installing the requirement.txt file, or work on the notebook on Google Colab. The template includes methods for downloading and pre-processing the data set that should not be changed. You will upload your report including written answers to Gradescope under HW02-Report. You will upload your completed notebook to Gradescope under HW02-Code.

a. (*5 pts*) Add code to train a KNN classifier on the training data set `X_train` with $K = 5$. Report the train, validation and test error rates.

b. (*5 pts*) Add code to train and evaluate the KNN model using values of K between 1 and 150. For each value of K, compute the training error rate and validation error rate. Include in your report a single graph showing the training error rate vs K as a line graph and the validation error rate vs K as a line graph. Be sure to include a legend and label the axes. Include the graph in your report.

c. (*5 pts*) Add code to determine the optimal value of K from the results computed in part (b). Report what you find for the optimal value of K. Also report the train error rate, validation error rate and test error rate found using the optimal value of K.

d. (*10 pts*)KNN can be highly sensitive to the scaling of the data. In this question, you will try normalizing the data using `sklearn.preprocessing.StandardScaler`.[1]. This class computes the mean and standard deviation for each feature. It then normalizes the data for each feature by subtracting the mean

[1]`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

and dividing by the standard deviation. This is often called the *z-transform*. You will need to apply the `StandardScaler` object's `fit` method to the training set to estimate the means and standard deviations. You then need to apply the `StandardScaler` object's `transform` method to the train, validation and test sets. Now, repeat parts (b) and (c) above, but using the normalized version of the data. Include a new plot of the train and validation error rates vs K, and report the train, validation and test error rates for the new optimal value of K. Did normalization help test performance for this data set?

e. (*5 pts*) Next you will experiment with training a one hidden layer neural network for this task using the scikit-learn `MLPClassifer` class.[2] We have provided a wrapper to instantiate `MLPClassifer` objects in the `BasicMLP` module. The function `BasicMLP.BasicMLP()` allows creating `MLPClassifer` objects while exposing three hyper-parameters: `hidden_layer_size`, `learning_rate_init`, and `max_iter` that allow setting the number of hidden units, the initial learning rate and the number of learning iterations. Using the normalized data set computed in the previous question, add code to instantiate the model with the default parameters (hidden_layer_size=8, learning_rate_init=0.01, max_iter=5000) and train the model on the training data set using the `MLPClassifer`'s `fit` method. Then, add code to evaluate the model by making predictions using the `MLPClassifer`'s `predict` method for the train, validation and test sets. Finally, compute the train, validation and test error rates. Include the error rates in your report.

f. (*10 pts*)Next, you will experiment with the number of hidden units in the model, the primary capacity control hyper-parameter. Using the normalized data set, learn the model using hidden layer sizes $1, 2, 4, 8, 16$. For each number of hidden units, compute the training error rate and validation error rate. Include in your report a single figure showing the training error rate vs the number of hidden units as a line graph and the validation error rate vs the number of hidden units as a line graph. Be sure to include a legend and label the axes. Include the graph in your report.

g. (*5 pts*) Add code to determine the optimal number of hidden units from the results computed in the previous part. Report what you find for the optimal number of hidden units. Also report the train error rate, validation error rate and test error rate found using the optimal number of hidden units. How does the test error rate compare to the test error rate of the optimal KNN model?

h. (*5 pts*) As a last experiment, you will explore training the model while controlling the maximum number of learning iterations. Using the normalized data, train the model with 16 hidden units and maximum number of learning iterations $50, 100, 500, 1000, 5000$. Compute the train and validation error rates for each setting of the maximum number of learning iterations. Include in your report a single figure showing the training error rate vs the maximum number of iterations as a line graph and the validation error rate vs maximum number of iterations a line graph. Be sure to include a legend and label the axes. Include the graph in your report. Looking at this graph, what effect does decreasing the maximum number of iterations have on the performance of the model and how does it compare to controlling the number of hidden units?

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`