CS589: Machine Learning - Fall 2025

Homework 6: Active Learning and Clustering

Assigned: Friday, November 14th, 2025

Getting Started: This assignment consists of coding problems. Download the assignment archive from Canvas and unzip the file. Starter code for select 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: This assignment is due at 8:00pm ET on November 20th, 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. You must select the page on which each answer appears. 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. (*50 points*) **Active Learning.** In this question, you will implement and experiment with active learning. In active learning, we start with a small labeled training data set $\mathcal{D}_{tr}$ and a large unlabeled pool of instances $\mathcal{D}_{pool}$. Our goal is to incrementally grow the training data set by selecting instances from $\mathcal{D}_{pool}$ to label. In this assignment, we will simulate the process of labeling data cases by revealing held out labels when data cases are selected to add to the pool. We will use the USPS handwritten digits data set for this question. As a model, we will use scikit-learn's multi-class logistic regression implementation. Complete the code in `active_learning.ipynb` and add answers to your report as noted in each question part.

a. (*10 pts*) To begin, complete the function `random_acquisition`. This function should return a list of length `batch_size` containing randomly selected indices of data cases in the unlabeled pool. We will acquire the labels for the corresponding pool instances later in the implementation. Random acquisition serves as an important baseline for active learning as any method that purports to optimize the selection of instances should be able to beat the performance of random acquisition on average. **Include your code in your report as the answer to this question**.

b. (*10 pts*) Next, complete the function `entropy_acquisition`. The entropy acquisition function computes the entropy of the predicted distribution over classes for each data case in the unlabeled pool using the current trained classifier. It returns a list of length `batch_size` of the indices of the unlabeled instances with the highest entropy. To obtain the predictive distribution of the unlabeled instances in the pool, use `predict_proba` method of the logistic regression model. Let the predictive probability for pool data case n and class c be p_{nc}. The entropy H_n of data case n is computed as shown below. Intuitively, the entropy quantifies how much uncertainty the model currently has regarding the class of each pool instance. By labeling the most uncertain instances, we hope to acquire maximum information per data case. **Include your code in your report as the answer to this question**.

$$H_n = -\sum_{c=1}^{C} p_{nc} \log p_{nc}$$

c. (*10 pts*)Next complete the function `labeling_oracle`. Once the data cases to be acquired have been selected using an acquisition function, the `labeling_oracle` function inserts them into training data set and removes them from the pool. To simulate the labeling process, each of the selected data cases from `Xpool` needs to be added to the array of training feature vectors `Xtr`. To simulate the labeling process, the held out labels of the pool points in `Ypool` also need to be inserted into `Ytr`. Finally, the selected feature vectors and labels need to be removed from `Xpool` and `Ypool`. The functions `np.vstack` and `np.delete` are useful for performing these operations. **Include your code in your report as the answer to this question**.

d. (*10 pts*)To finish the implementation, complete the function `active_learning_loop`. On each active learning iteration, this function needs to (1) initialize and fit the classifier using the current training set `Xtr` and `Ytr`, (2) call the acquisition function to select points to label using the current pool `Xpool`, classifier, and desired batch size as inputs, (3) call the `labeling_oracle` to add the selected points to the training set and remove them from the pool and (4) compute and store the test error rate of the current model. At the end of the active learning iterations, the function returns the list of test error rates to support plotting. The logistic regression model should be initialized using `LogisticRegression(max_iter=1000)`. **Include your code in your report as the answer to this question**.

e. (*10 pts*) Lastly, run the active learning experiment and make one plot showing the test error rate as a function of number of data cases in the training set for the random and entropy acquisition functions. This should be a single plot with two trend lines. Make sure to include a legend and label the axes. Include your figure in your report as your answer to this question. Does entropy-based active learning outperform the random baseline for this data set? (Note: this experiment should take less than 5 minutes to run. If it appears to be taking longer, you may need to vectorize your computations for better computational efficiency.)

2. (*50 points*) **Clustering.** In this question, you will experiment with clustering using the USPS handwritten digits data set. You will compare K-Means clustering to hierarchical agglomerative clustering. Add your code to `clustering.ipynb`.

a. (*10 pts*)A key notion in clustering is the cluster centroid or prototype. This is a representation of the center of the cluster in the data space. To begin, complete the function `cluster_center`. This function takes as input a data matrix `X` and a list of data case indices `ind` belonging to a cluster. It computes and returns the average of the data cases assigned to the cluster. Mathematically, this operation is $\mu = \frac{1}{|\mathcal{I}|}\sum_{n\in\mathcal{I}} \mathbf{X}_n$ with $\mathbf{X}$ the data matrix and $\mathcal{I}$ the list of indices. As your answer to this question, compute the centroid assuming that all of the data cases in the training set are in the same cluster. Convert the centroid to an image (an example is shown in the starter code) and include the image in your report.

b. (*10 pts*)To compare clusterings, we will use the total within-cluster sum of squared distances. Letting $\mathbf{z}$ be a vector containing the index of the cluster that each data case belongs to and μ_k be the cluster prototype, we can express this function as shown below. Lower values indicate better clusterings. Implement this function in `cluster_objective`. Include in your report the value of this objective function when applied to the case where all of the training data are in the same cluster (use the result from (a) as the centroid).

$$\sum_{k=1}^{K}\sum_{n=1}^{N}[\mathbf{z}_n = k]\cdot\|\mathbf{x}_n - \mu_k\|_2^2$$

c. (*10 pts*) Add code to `clustering.ipynb` to fit the K-Means algorithm to the training data set using 10 clusters (matching the number of true classes in the data set) and 20 restarts. To analyze the clusters, use the K-Means `predict` function to obtain the cluster assignment for each data case in the training set and provide the following output:

- Using the `cluster_center` method to obtain the centroids, compute and report the value of the `cluster_objective` as an assessment of the overall clustering quality.
- Make a bar chart showing the number of data cases assigned to each of the 10 clusters and include it in your report.
- Compute the cluster centroid for each cluster, convert to an image and display the centroid in your report. This gives an indication of what digit type(s) are represented in the cluster.
- For each cluster, use the training set labels `Ytrain` to make a bar chart showing the number of data cases belonging to the cluster that have each label. This gives and indication of how precise the correspondence is between clusters and true classes.

d. (*10 pts*)Add code to `clustering.ipynb` to fit the hierarchical agglomerative clustering algorithm to the training data set using 10 clusters (matching the number of true classes in the data set). Use the average linkage. To analyze the clusters, use the `fit_predict` function to obtain the cluster assignment for each data case in the training set and provide the following output:

- Using the `cluster_center` method to obtain the centroids, compute and report the value of the `cluster_objective` as an assessment of the overall clustering quality.
- Make a bar chart showing the number of data cases assigned to each of the 10 clusters and include it in your report.
- Compute the cluster centroid for each cluster, convert to an image and display the centroid in your report. This gives an indication of what digit type(s) are represented in the cluster.
- For each cluster, use the training set labels `Ytrain` to make a bar chart showing the number of data cases belonging to the cluster that have each label. This gives and indication of how precise the correspondence is between clusters and true classes.

e. (*10 pts*) Lastly, compare and contrast the output produced in parts (c) and (d). Which method appears to be doing a better job overall on this data set? To what extent do the clusters correspond to specific digit types? What strengths and weaknesses do you see in the output of each method?

3. (*0 points*) If you used generative AI tools to help complete any programming questions on this assignment, please briefly describe which tools you used, how you used them, and for which problems you used them. If you did not use generative AI tools, please indicate that as your response to this question. (Note: Not answering this question will result in a deduction of 2 points.)