Homework 3 - Oct. 20

Deadline: Monday, Nov. 3, at 11:59pm.

Submission: You need to submit through Crowdmark (you can find the link on Quercus) with your answers to Questions 1 and 2. You will upload your answers to each subquestion separately. You can produce the submissions however you like (e.g. LaTeX, Microsoft Word, scanner), as long as they are readable.

Marking: Your mark will be out of 7.5. This mark will be your mark on *either* question 1 *or* question 2. We will decide which question to mark after the deadline, and we will mark the same question for everyone in the class. To aid your learning, we will be releasing the solutions to both questions, and covering the solutions in office hours.

Neatness: We reserve the right to deduct a point for neatness, if we have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the total possible marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Other Policies: See the syllabus[1] for detailed collaboration, generative AI, and academic integrity policies.

1. **[7.5 pts] Logistic Regression.** In this problem, you will implement logistic regression by completing the provided code in `logistic_regression.py` and experiment with the completed code.

   Throughout this homework, you will be working with a subset of hand-written digits, 2's and 3's, represented as 16×16 pixel arrays. We show the example digits in Figure 1. The pixel intensities are between 0 and 1, and were read into the vectors in a raster-scan manner. You are given two training sets: `digits_train` which contains 300 examples of each class and `digits_train_small` which contains 2 examples of each class. You can access these training sets by using functions `load_train` and `load_train_small` in `utils.py`. You are also given a validation set that you should use for model selection and a test set that you should use for reporting the final performance. Optionally, the code for visualizing the dataset is located at `utils.py`.

   Carefully read the provided code in `logistic_regression.py`. You should understand the code instead of using it as a black box. You need to implement the penalized logistic regression model, where the average train loss is defined as:

$$\begin{aligned}\hat{\mathcal{R}}[\mathbf{w}, b] &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_{\text{CE}}(y^{(i)}, t^{(i)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \\ &= \frac{1}{N}\sum_{i=1}^{N} \Big(-t^{(i)} \log y^{(i)} - (1 - t^{(i)}) \log(1 - y^{(i)})\Big) + \frac{\lambda}{2}\|\mathbf{w}\|^2,\end{aligned}$$

[1] https://www.cs.toronto.edu/~cmaddis/courses/sta314_f25/sta314_f25_syllabus.pdf

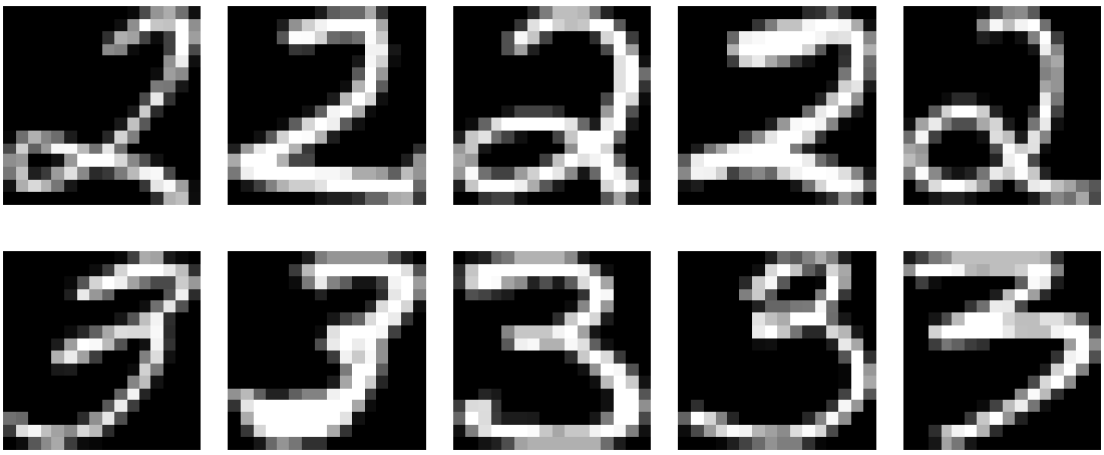

Figure 1: Example digits. Top and bottom show digits of 2s and 3s, respectively.

where

$$y^{(i)} = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x}^{(i)} - b)}.$$

Here N is the total number of data points, $\mathbf{w}$ is the weight of logistic regression model, and b is the bias parameter.

Note: in this question we will only regularize the weights w and not the bias parameter b. This makes sense for logistic regression because we don't have any reason to expect the intercept to be small, but we may want to control the margin, i.e., the norm of the weights w.

(a) **[2 pts]** Implement the functions `logistic_predict`, `evaluate`, and `logistic` located at `logistic_regression.py`. While implementing the functions, remember to vectorize the operations; you should not have have any `for-loops` in these functions. Include your code in your answer.

(b) **[2.5 pts]** Complete the missing parts in a function `run_logistic_regression`. The function should train the logistic regression model using gradient descent on `digits_train` training set. You may use the implemented functions from part (a). Experiment with the hyperparameters for the learning rate and the number of iterations (if you have a smaller learning rate, your model will take longer to converge). You will fix your ℓ_2 weight regularization to 0 for this part. If you get `NaN/Inf` errors, you may need to reduce your learning rate. Include your code in your answer.

Moreover, in the write-up, report which hyperparameter settings you found worked the best and the final cross-entropy and classification accuracy on the training, validation, and test sets. Note that you should only compute the test error once you have selected your best hyperparameter settings using the validation set.

(c) **[1 pt]** Examine how the cross-entropy changes as the training progresses. Generate and report a plot that shows the training curve (iteration counter on x-axis and cross-entropy on y-axis). The plot should have two curves: one for the training set and one for the validation set. Run your code several times and observe if the results change. If they do, how would you choose the best hyperparameter settings?

(d) **[1 pt]** Using the same hyperparameter settings (for learning rate and number of iteration) that worked well in part (b), train the logistic regression model with different values of weight regularization $\lambda \in \{0., 0.001, 0.01, 0.1, 1.0\}$. Generate and report a plot that shows how the validation cross-entropy changes as you train with different weight regularization λ.

Repeat the same experiment and report an additional plot with `digits_train_small` training set. You may need to additionally tune the hyperparameter settings (e.g. learning rate). You should have two plots in total that show the relationship between weight regularization λ and validation cross-entropy: one for each training set `digits_train` and `digits_train_small`.

(e) **[1 pt]** For each dataset (`digits_train` and `digits_train_small`), how does the train and validation cross-entropy change when you increase λ? Do they go up, down first up and down, or down and then up? Explain why you think they behave this way. Which is the best value of λ based on your experiment? Report the test cross-entropy and classification accuracy for the best value of λ.

2. **[7.5 pts]** ***K*-Means Clustering.** In this question you will be reasoning about the set of global optima of the K-Means algorithm.

A stationary configuration of the K-means algorithm is a configuration of vectors $\{\mathbf{m}_k\}_{k=1}^K \subseteq \mathbb{R}^D$ and assignment vectors $\{\mathbf{r}^{(i)}\}_{i=1}^N \subseteq \{0,1\}^K$ which does not change during refitting (setting $\mathbf{m}_k$ to mean of cluster k) and reassignment (assigning $\mathbf{r}^{(i)}$ to the closest center). A globally optimal configuration is one that minimizes the K-Means objective,

$$\min_{\mathbf{m}_k, \mathbf{r}^{(i)}} \sum_{i=1}^{N} \sum_{k=1}^{K} r_k^{(i)} d(\mathbf{m}_k, \mathbf{x}^{(i)})^2.$$

It's not hard to see that globally optimal configurations must also be stationary configurations (if they weren't, then one step of K-means would produce a configuration with lower objective value, i.e., a contradiction).

You will be asked to list the globally optimal configurations of the K-means algorithm for a specific choice of K and distance metric on the following dataset with 3 points:

$x_1^{(i)}$	$x_2^{(i)}$
0	1
0	-1
4	0

Your answers should be a list of clusterings of the three data points above (corresponding to cluster assignments) together with means for each cluster. Notice that globally optimal configurations cannot have empty clusters (otherwise, we could reduce the K-means objective by assigning one point to the empty cluster and refitting the means), so you can omit those configurations in your analysis.

(a) **[1.5 pts]** For the dataset above, list all of the globally optimal configurations of the K-Means algorithm with $K = 1$ and the Euclidean norm. Justify your answer.

(b) **[3 pts]** For the dataset above, list all of the globally optimal configurations of the K-Means algorithm with $K = 2$ and the Euclidean norm. Justify your answer.

(c) **[3 pts]** For the dataset above, list all of the globally optimal configurations of the K-Means algorithm with $K = 2$ and the following distance metric:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\left(\frac{x_1 - y_1}{4}\right)^2 + (x_2 - y_2)^2}$$

Justify your answer.