CS 189 Introduction to Machine Learning
Spring 2022 Marvin Zhang

HW2

Due 2/28/22 at 11:59pm

- Homework 2 consists of both written and coding questions.
- We prefer that you typeset your answers using LaTeX or other word processing software. If you haven't yet learned LaTeX, one of the crown jewels of computer science, now is a good time! Neatly handwritten and scanned solutions will also be accepted for the written questions.
- In all of the questions, **show your work**, not just the final answer.

Deliverables:

1. Submit a PDF of your homework ,**with an appendix listing all your code**, to the Gradescope assignment entitled "HW2 Write-Up". **Please start each question on a new page.** If there are graphs, include those graphs in the correct sections. **Do not** put them in an appendix. We need each solution to be self-contained on pages of its own.
    - In your write-up, please state with whom you worked on the homework. This should be on its own page and should be the first page that you submit.
    - In your write-up, please copy the following statement and sign your signature next to it. (Mac Preview and FoxIt PDF Reader, among others, have tools to let you sign a PDF file.) We want to make it *extra* clear so that no one inadvertently cheats. *"I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."*
    - **Replicate all your code in an appendix**. Begin code for each coding question in a fresh page. Do not put code from multiple questions in the same page. When you upload this PDF on Gradescope, *make sure* that you assign the relevant pages of your code from appendix to correct questions.

1 Administrivia (2 points)

1. Please fill out the Check-In Survey if you haven't already. Please write down the 10 digit alphanumeric code you get after completing the survey.
2. **Declare and sign the following statement:**

   "I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

   Signature: ______________

   While discussions are encouraged, *everything* in your solution must be your (and only your) creation. Furthermore, all external material (i.e., anything outside lectures and assigned readings, including figures and pictures) should be cited properly. We wish to remind you that consequences of academic misconduct are particularly severe

2 Convergence for Logistic Regression (10 points)

In this problem, we will take steps toward proving that gradient descent converges to a unique minimizer of the logistic regression cost function, binary cross-entropy, when combined with L_2 regularization. In particular, we will consider the simplified case where we are minimizing this cost function for a single data point. For weights $w \in \mathbb{R}^d$, data $x \in \mathbb{R}^d$, and a label $y \in \{0, 1\}$, the L_2-regularized logistic regression cost function is given by

$$J(w) = -y \log s(x \cdot w) - (1 - y) \log(1 - s(x \cdot w)) + \frac{1}{2}\lambda \|w\|_2^2$$

Where $s(\gamma) = 1/(1 + \exp(-\gamma))$ is the logistic function (also called the sigmoid) and $\lambda > 0$. You may assume that $x \neq 0$.

(a) (2 points) To start, write the gradient descent update function $G(w)$, which maps w to the result of a single gradient descent update with learning rate $\epsilon > 0$.

(b) (2 points) Show that the cost function J has a unique minimizer w^* by proving that J is strictly convex.
Hint: A sufficient condition for a function to by strictly convex is that its Hessian is positive definite.

(c) (4 points) Next, show that for a sufficiently chosen ϵ, G is a *contraction*, which means that there is a constant $0 < \rho < 1$ such that, for any $w, w' \in \mathbb{R}^d$, $\left\|G(w) - G(w')\right\|_2 < \rho \|w - w'\|_2$.
Hint: this is equivalent to showing that the Jacobian of G has bounded spectral norm: $\left\|\nabla_w G(w)\right\|_2 < \rho$.
Hint 2: for a PSD matrix, the eigenvalues and singular values are equivalent.
Hint 3: consider the eigenvalues of $A + \alpha I$ for a square matrix, A and $\alpha \in \mathbb{R}$.

(d) (2 points) Finally, calling $w^{(t)}$ the t-th iterate of gradient descent, show that $\left\|w^* - w^{(t)}\right\|_2 < \rho^t \left\|w^* - w^{(0)}\right\|_2$, so that $\lim_{t\to\infty} \left\|w^* - w^{(t)}\right\|_2 = 0$.

3 Theory of Hard-Margin Support Vector Machines (7 points)

A *decision rule* (or *classifier*) is a function $r : \mathbb{R}^d \to \pm 1$ that maps a feature vector (test point) to $+1$ ("in class") or -1 ("not in class"). The decision rule for linear SVMs is

$$r(x) = \begin{cases} +1 & \text{if } w \cdot x + \alpha \geq 0, \\ -1 & \text{otherwise,} \end{cases} \tag{1}$$

where $w \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ are the weights (parameters) of the SVM. The hard-margin SVM optimization problem (which chooses the weights) is

$$\min_{w,\alpha} \; |w|^2 \;\text{ subject to }\; y_i(X_i \cdot w + \alpha) \geq 1, \;\; \forall i \in \{1, \ldots, m\}, \tag{2}$$

where $|w| = \|w\|_2 = \sqrt{w \cdot w}$.

We can rewrite this optimization problem by using Lagrange multipliers to eliminate the constraints. We thereby obtain the equivalent optimization problem

$$\max_{\lambda_i \geq 0} \min_{w,\alpha} \; |w|^2 - \sum_{i=1}^{m} \lambda_i (y_i (X_i \cdot w + \alpha) - 1). \tag{3}$$

(a) (5 points) Show that Equation (3) can be rewritten as the *dual optimization problem*

$$\max_{\lambda_i \geq 0} \; \sum_{i=1}^{m} \lambda_i - \frac{1}{4} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j X_i \cdot X_j \;\text{ subject to }\; \sum_{i=1}^{m} \lambda_i y_i = 0. \tag{4}$$

Hint: Use calculus to determine what values of w and α optimize Equation (3). Explain where the new constraint comes from.

Learning how to take the dual is useful because sometimes it is easier to solve the dual quadratic problem rather than the primal quadratic problem.

(b) (1 point) Suppose we know the values λ_i^* and α^* that optimize Equation (3). Show that the decision rule specified by Equation (1) can be written

$$r(x) = \begin{cases} +1 & \text{if } \alpha^* + \frac{1}{2}\sum_{i=1}^{m} \lambda_i^* y_i X_i \cdot x \geq 0, \\ -1 & \text{otherwise.} \end{cases} \tag{5}$$

(c) (1 point) The training points X_i for which $\lambda_i^* > 0$ are called the support vectors. In practice, we frequently encounter training data sets for which the support vectors are a small minority of the training points, especially when the number of training points is much larger than the number of features (i.e., the dimension of the feature space). Explain why the support vectors are the only training points needed to evaluate the decision rule. What influence do the non-support vectors have on the decision rule?

4 Wally's Winery (15 points)

Wally Walter and the Walter family run the most popular winery in Napa. This summer, UC Berkeley students are flooding in to try the Walter Family's fine wines! Wally's children, Wilma and Willy, are slowly helping Wally take ownership of the family business. However, they do not possess the quintessential Walter talent: identifying red and white wines based on attributes! To help them, Wally has decided to train a machine learning classifier to distinguish Walter's Red Wine from Walter's White Wine.

In this exercise, we will build a logistic regression classifier to classify whether a specific wine is a white (class label 0) or red (class label 1) wine, based on its features

Use of automatic logistic regression libraries/packages is prohibited for this question. If you are coding in python, it is better to use np.true_divide for evaluating logistic functions as its code is numerically stable, and doesn't produce NaN or MathOverflow exceptions.

The wine dataset given to you as part of the homework in `data.mat` consists of 6,497 data points, each having 12 features. The description of these features is provided in `data.mat`. The dataset includes a training set of 6,000 data points and a test set of 497 data points. Your classifier needs to predict whether a wine is white (class label 0) or red (class label 1).

For this homework, you need to do the following.

1. (2 points) Preprocess the data by first adding a fake feature to the data matrix. Then, split the data into training and validation sets. Finally, normalize each set so that each feature has mean 0 and variance 1.

2. (2 point) Derive and write down the batch gradient descent update equation for logistic regression with ℓ_2 regularization. As this is a "batch" algorithm, each iteration should use every training example.

3. (5 points) Implement the sigmoid function and training using batch gradient descent. Then, plot the loss function as a function of the number of iterations. Use a learning rate of $1e-4$ and a regularization parameter of $\lambda = 0.1$

4. (2 point) Derive and write down the stochastic gradient descent update equation for logistic regression with ℓ_2 regularization. Since this is not a "batch" algorithm anymore, each iteration uses *just one* training example.

5. (2 points) Implement training using stochastic gradient descent. Then, plot the loss function as a function of the number of iterations. Use a learning rate of $1e-6$ and a regularization parameter of $\lambda = 0.1$

   Comment on the differences between the convergence of batch and stochastic gradient descent.

6. (2 points) Instead of a constant learning rate lr, repeat part 2 where the learning rate in every iteration decays as $lr = \frac{lr_0}{i+1}$, where lr_0 was the original learning rate. Begin with a learning rate of 1e-4, and report your results.Plot the loss function vs. the number of iterations. Is this

strategy better than having a constant *lr*? What could you do to improve performance with a decaying learning rate?

IMPORTANT: Do NOT use any software package for logistic regression that you didn't write yourself! You are only allowed to use numpy for your machine learning models!

5 Gaussian Classifiers for Digits and Spam (13 points)

In this problem, you will build classifiers based on Gaussian discriminant analysis. You are NOT allowed to use any libraries for out-of-the-box classification (e.g. `sklearn`). You may use anything in `numpy` and `scipy`.

The training and test data can be found with this homework. Submit your predicted class labels for the test data on the Kaggle competition website and be sure to include your Kaggle display name and scores in your writeup. Also be sure to include an appendix of your code at the end of your writeup.

(a) (2 points) Taking pixel values as features (no new features yet, please), fit a Gaussian distribution to each digit class using maximum likelihood estimation. This involves computing a mean and a covariance matrix for each digit class, as discussed in lecture.

Hint: You may, and probably should, contrast-normalize the images before using their pixel values. One way to normalize is to divide the pixel values of an image by the l_2-norm of its pixel values.

(b) (1 point) (Written answer.) Visualize the covariance matrix for a particular class (digit). How do the diagonal terms compare with the off-diagonal terms? What do you conclude from this?

(c) (10 points) Classify the digits in the test set on the basis of posterior probabilities with two different approaches.

(1) Linear discriminant analysis (LDA). Model the class conditional probabilities as Gaussians $\mathcal{N}(\mu_C, \Sigma)$ with different means μ_C (for class C) and the same covariance matrix Σ, which you compute by averaging the 10 covariance matrices from the 10 classes.

Linear discriminant analysis (LDA). Model the class conditional probabili- ties as Gaussians N(C,) with different means C (for class C) and the same pooled

within-class covariance matrix , which you compute from a weighted average of the 10 covariance matrices from the 10 classes, as described in lecture.

You may find **scipy.stats.multivariate_normal.logpdf** and/or **numpy.linalg.slogdet** helpful

Hold out 10,000 randomly chosen training points for a validation set. Classify each image in the validation set into one of the 10 classes (with a 0-1 loss function). Compute the error rate and plot it over the following numbers of randomly chosen training points: 100, 200, 500, 1,000, 2,000, 5,000, 10,000, 30,000, 50,000. (Expect some variation in your error rate when few training points are used.)

(2) Using the `mnist_data.mat`, train your best classifier for the `training_data` and classify the images in the `test_data`. Submit your labels to the online Kaggle competition. Record your optimum prediction rate in your submission. You are welcome to compute extra features for the Kaggle competition. If you do so, please describe your implementation in your assignment. Please use extra features **only** for the Kaggle portion of the assignment.

In your submission, include plots of error rate versus number of training examples for both LDA and QDA. Similarly, include a plot of validation error versus the number of training points for each digit. Plot all the 10 curves on the same graph as shown in Figure 1. Which digit is easiest to classify? Include written answers where indicated.

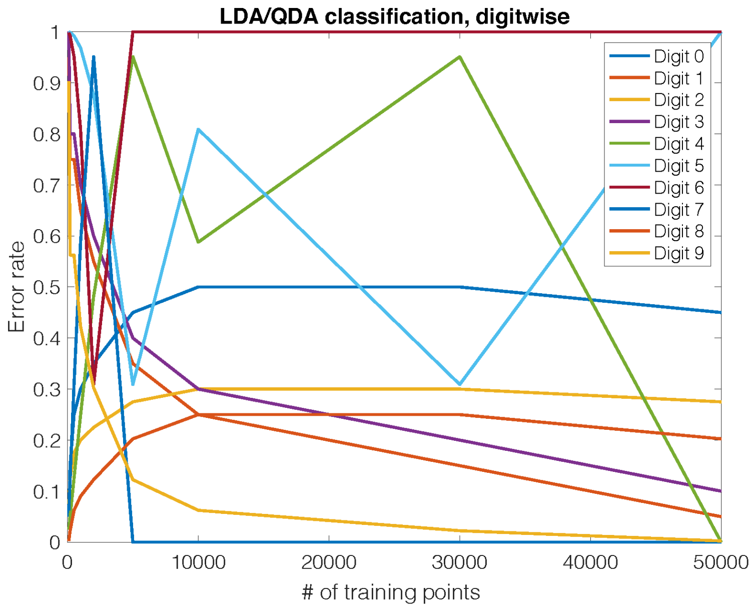

Figure 1: Sample graph with 10 plots