CS 189 Introduction to Machine Learning
Spring 2020 Jonathan Shewchuk HW4

Due: Wednesday, March 11 at 11:59 PM

This homework consists of coding assignments and math problems. **Begin early; you can submit models to Kaggle only twice a day!**

1. Kaggle: Submit your predictions to `https://www.kaggle.com/t/253477f577514ce8babcd1e86089d3c8`

2. Write-up: Submit your solution in **PDF** format to "Homework 4 Write-Up" on Gradescope.
   - State your name, and if you have discussed this homework with anyone (other than GSIs), list the names *of them all*.
   - Begin the solution for each question in a new page. Do not put content for different questions in the same page. You may use multiple pages for a question if required.
   - If you include figures, graphs or tables for a question, any explanations should accompany them in *the same page*. Do NOT put these in an appendix!
   - **Only PDF uploads to Gradescope will be accepted**. You may use LaTeX or Word to typeset your solution or scan a neatly handwritten solution to produce the PDF.
   - **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.

3. Code: Additionally, submit all your code as a ZIP to "Homework 4 Code" on Gradescope.
   - **Set a seed for all pseudo-random numbers generated in your code.** This ensures your results are replicated when readers run your code.
   - Include a README with your name, student ID, the values of the random seed (above) you used, and any instructions for compilation.
   - Do NOT provide any data files, but supply instructions on how to add data to your code.
   - Code requiring exorbitant memory or execution time won't be considered.
   - Code submitted here must match that in the PDF Write-up, and produce the *exact* output submitted to Kaggle. Inconsistent or incomplete code won't be accepted.

Notation. In this assignment we use the following conventions.

- Symbol "defined equal to" ($\triangleq$) *defines* the quantity to its left to be the expression to its right.
- Scalars are lowercase non-bold: x, u_1, α_i. Matrices are uppercase alphabets: A, B_1, C_i. Vectors (column vectors) are in bold: $\mathbf{x}, \boldsymbol{\alpha}_\mathbf{1}, \mathbf{X}, \mathbf{Y}_\mathbf{j}$.
- $||\mathbf{v}||$ denotes the Euclidean norm (length) of vector $\mathbf{v}$: $||\mathbf{v}|| \triangleq \sqrt{\mathbf{v}\cdot\mathbf{v}}$. $||A||$ denotes the (operator) norm of matrix A, the magnitude of its largest singular value: $||A|| = \max_{||v||=1} ||Av||$.
- $[n] \triangleq \{1, 2, 3, \ldots, n\}$. $\mathbf{1}$ and $\mathbf{0}$ denote the vectors with all-ones and all-zeros, respectively.

1 Honor Code

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."

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 Logistic Regression with Newton's Method

Given examples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and associated labels $y_1, y_2, \ldots, y_n \in \{0, 1\}$, the cost function for *unregularized* logistic regression is

$$J(\mathbf{w}) \triangleq -\sum_{i=1}^{n} \left(y_i \ln s_i + (1 - y_i) \ln(1 - s_i) \right)$$

where $s_i \triangleq s(\mathbf{x}_i \cdot \mathbf{w})$, $\mathbf{w} \in \mathbb{R}^d$ is a weight vector, and $s(\gamma) \triangleq 1/(1 + e^{-\gamma})$ is the logistic function.

Define the $n \times d$ design matrix X (whose i^{th} row is $\mathbf{x}_i^T$), the label n-vector $\mathbf{y} \triangleq [y_1 \ \ldots \ y_n]^T$, and $\mathbf{s} \triangleq [s_1 \ \ldots \ s_n]^{\mathsf{T}}$. For an n-vector $\mathbf{a}$, let $\ln \mathbf{a} \triangleq [\ln a_1 \ \ldots \ \ln a_n]^T$. The cost function can be rewritten in vector form as $J(\mathbf{w}) = -\mathbf{y} \cdot \ln \mathbf{s} - (\mathbf{1} - \mathbf{y}) \cdot \ln (\mathbf{1} - \mathbf{s})$.

Hint: Recall matrix calculus identities $\nabla_{\mathbf{x}} \alpha \mathbf{y} = \left(\nabla_{\mathbf{x}} \alpha\right) \mathbf{y}^T + \alpha \nabla_{\mathbf{x}} \mathbf{y}$; $\nabla_{\mathbf{x}} \left(\mathbf{y} \cdot \mathbf{z}\right) = \left(\nabla_{\mathbf{x}} \mathbf{y}\right) \mathbf{z} + \left(\nabla_{\mathbf{x}} \mathbf{z}\right) \mathbf{y}$; $\nabla_{\mathbf{x}} \mathbf{f}(\mathbf{y}) = \left(\nabla_{\mathbf{x}} \mathbf{y}\right)\left(\nabla_{\mathbf{y}} \mathbf{f}(\mathbf{y})\right)$; $\nabla_{\mathbf{x}} g(\mathbf{y}) = \left(\nabla_{\mathbf{x}} \mathbf{y}\right)\left(\nabla_{\mathbf{y}} g(\mathbf{y})\right)$; and $\nabla_{\mathbf{x}} C \mathbf{y}(\mathbf{x}) = \left(\nabla_{\mathbf{x}} \mathbf{y}(\mathbf{x})\right) C^T$ (where C is a constant matrix).

1 Derive the gradient $\nabla_{\mathbf{w}} J(\mathbf{w})$ of cost $J(\mathbf{w})$ as a matrix-vector expression. Also derive *all intermediate derivatives* in matrix-vector form. Do NOT specify them in terms of their individual components.

2 Derive the Hessian $\nabla^2_{\mathbf{w}} J(\mathbf{w})$ for the cost function $J(\mathbf{w})$ as a matrix-vector expression.

3 Write the matrix-vector update law for one iteration of Newton's method, substituting the gradient and Hessian of $J(\mathbf{w})$.

4 You are given four examples $\mathbf{x_1} = [0.2 \quad 3.1]^T, \mathbf{x_2} = [1.0 \quad 3.0]^T, \mathbf{x_3} = [-0.2 \quad 1.2]^T, \mathbf{x_4} = [1.0 \quad 1.1]^T$ with labels $y_1 = 1, y_2 = 1, y_3 = 0, y_4 = 0$. These points cannot be separated by a line passing through origin. Hence, as described in lecture, append a 1 to each $\mathbf{x}_{i \in [4]}$ and use a weight vector $\mathbf{w} \in \mathbb{R}^3$ whose last component is the bias term (called α in lecture). Begin with initial weight $w^{(0)} = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^{\mathsf{T}}$. For the following, state only the final answer with four digits after the decimal point. You may use a calculator or write a program to solve for these, but do NOT submit any code for this part.

(a) State the value of $\mathbf{s}^{(0)}$ (the initial value of $\mathbf{s}$).

(b) State the value of $\mathbf{w}^{(1)}$ (the value of $\mathbf{w}$ after 1 iteration).

(c) State the value of $\mathbf{s}^{(1)}$ (the value of $\mathbf{s}$ after 1 iteration).

(d) State the value of $\mathbf{w}^{(2)}$ (the value of $\mathbf{w}$ after 2 iterations).

3 Wine Classification with Logistic Regression

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

Begin by normalizing each feature and adding a fictitious dimension. Whenever required, it is recommended that you tune hyperparameter values with cross-validation.

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

1 *Batch Gradient Descent Update*. State the batch gradient descent update law for logistic regression with ℓ_2 regularization. As this is a "batch" algorithm, each iteration should use *every training example*. You don't have to show your derivation. You may reuse results from your solution to question 4.1.

2 *Batch Gradient Descent Code*. Choose reasonable values for the regularization parameter and step size (learning rate), specify your chosen values, and train your model from question 3.1. Plot the value of the cost function versus the number of iterations spent in training.

3 *Stochastic Gradient Descent (SGD) Update*. State the SGD update law for logistic regression with ℓ_2 regularization. Since this is not a "batch" algorithm anymore, each iteration uses *just one* training example. You don't have to show your derivation.

4 *Stochastic Gradient Descent Code*. Choose a suitable value for the step size (learning rate), specify your chosen value, and run your SGD algorithm from question 3.3. Plot the value of the cost function versus the number of iterations spent in training.

Compare your plot here with that of question 3.2. Which method converges more quickly? Briefly describe what you observe.

5 Instead of using a constant step size (learning rate) in SGD, you could use a step size that slowly shrinks from iteration to iteration. Run your SGD algorithm from question 3.3 with a step size $\epsilon_t = \delta/t$ where t is the iteration number and δ is a hyperparameter you select empirically. Mention the value of δ chosen. Plot the value of cost function versus the number of iterations spent in training.

How does this compare to the convergence of your previous SGD code?

6 *Kaggle*. Train your *best* classifier on the entire training set and submit your prediction on the test sample points to Kaggle. As always for Kaggle competitions, you are welcome to add

or remove features, tweak the algorithm, and do pretty much anything you want to improve your Kaggle leaderboard performance **except** that you may not replace logistic regression with a wholly different learning algorithm. Your code should output the predicted labels in a CSV file.

Report your Kaggle username and your best score, and briefly describe what your best classifier does to achieve that score.

4 Convergence of Batch Gradient Descent in Logistic Regression

In this problem, you will prove that batch gradient descent converges to a unique optimizer of the ℓ_2-regularized logistic regression cost function.

Given sample points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and associated labels $y_1, y_2, \ldots, y_n \in \{0, 1\}$, define the design matrix X (whose i^{th} row is $\mathbf{x}_i^T$), the label n-vector $\mathbf{y} \triangleq [y_1 \ \ldots \ y_n]^T$, and $\mathbf{s}(X\mathbf{w}) \triangleq [s_1 \ \ldots \ s_n]^T$ containing values $s_{i\in[n]} \triangleq 1/(1 + e^{-\mathbf{x}_i \cdot \mathbf{w}})$. For any vector $\mathbf{a}$, let $\ln \mathbf{a} \triangleq [\ln a_1 \ \ldots \ \ln a_n]^T$.

The cost function for ℓ_2-regularized logistic regression is

$$J(\mathbf{w}) \triangleq \frac{\lambda \|\mathbf{w}\|^2}{2} - \mathbf{y} \cdot \ln(\mathbf{s}(X\mathbf{w})) - (\mathbf{1} - \mathbf{y}) \cdot \ln(\mathbf{1} - \mathbf{s}(X\mathbf{w}))$$

where $\lambda > 0$ is your choice of the regularization parameter.

1 Let $\mathbf{w}^{(t)}$ denote the value of $\mathbf{w}$ at iteration t. The initial, arbitrary weight vector is $\mathbf{w}^{(0)}$. State the gradient descent update rule for calculating the value of $\mathbf{w}^{(t+1)}$ as a function $\mathbf{g}(\mathbf{w}^{(t)})$ of the previous weight vector $\mathbf{w}^{(t)}$, with a constant step size (learning rate) $\epsilon > 0$.

2 Show that $J(\cdot)$ is strictly convex and $J(\mathbf{w})$ has a unique minimizer $\mathbf{w}^*$.
Hint: $f(\mathbf{x})$ is strictly convex if its Hessian $\nabla^2_{\mathbf{x}} f$ is positive definite everywhere.

3 Next, show that if the step size (learning rate) ϵ is a sufficiently small constant, then the update function $\mathbf{g}(\cdot)$ is a *contraction*; i.e., there exists a constant $\rho \in (0, 1)$ such that for every two $\mathbf{w}_1, \mathbf{w}_2$, $\|\mathbf{g}(\mathbf{w}_1) - \mathbf{g}(\mathbf{w}_2)\| \leq \rho \|\mathbf{w}_1 - \mathbf{w}_2\|$.
Hint: The Mean Value Theorem and the Cauchy–Schwarz inequality might both help.

4 Finally, complete your proof by showing that if the step size ϵ is chosen as required in question 4.3, the weight converges to the unique minimizer; that is, $\lim\limits_{t\to\infty} \mathbf{w}^{(t)} = \mathbf{w}^*$.

5 You can refine your proof and guarantee quicker convergence by tightening the contraction in question 4.3. Show that for a clever choice of ϵ, which may depend on X and λ, but *crucially* **not** on the weights $\mathbf{w}^{(0)}$ and $\mathbf{w}^*$, you can guarantee that $\|\mathbf{w}^{(t)} - \mathbf{w}^*\| \propto \exp\left(-t \frac{8\lambda}{8\lambda + \sum_i \|\mathbf{x}_i\|^2}\right)$.

Argue that this is also the *best exponential rate of convergence* one can guarantee when using constant learning rates.

6 If we set $\lambda = 0$, we have the unregularized logistic regression problem. Now that your proof is complete, do you see why the condition $\lambda > 0$ is necessary? What are the reasons that your proof won't be valid anymore if you choose $\lambda = 0$?

5 A Bayesian Interpretation of Lasso

Suppose you are aware that the labels $y_{i\in[n]}$ corresponding to sample points $\mathbf{x}_{i\in[n]} \in \mathbb{R}^d$ follow the density law

$$f(y|\mathbf{x},\mathbf{w}) \triangleq \frac{1}{\sigma\sqrt{2\pi}} e^{-(y-\mathbf{w}\cdot\mathbf{x})^2/(2\sigma^2)}$$

where $\sigma > 0$ is a known constant and $\mathbf{w} \in \mathbb{R}^d$ is a random parameter. Suppose further that experts have told you that

- each component of $\mathbf{w}$ is independent of the others, and
- each component of $\mathbf{w}$ has the Laplace distribution with location 0 and scale being a known constant b. That is, each component $\mathbf{w}_i$ obeys the density law $f(\mathbf{w}_i) = e^{-|\mathbf{w}_i|/b}/(2b)$.

Assume the outputs $y_{i\in[n]}$ are independent from each other.

Your goal is to find the choice of parameter $\mathbf{w}$ that is *most likely* given the input-output examples $(\mathbf{x}_i, y_i)_{i\in[n]}$. This method of estimating parameters is called *maximum a posteriori* (MAP); Latin for *"maximum [odds] from what follows."*

1. Derive the *posterior* probability density law $f(\mathbf{w}|(\mathbf{x_i}, y_i)_{i\in[n]})$ for $\mathbf{w}$ *up to a proportionality constant* by applying Bayes' Theorem and using the densities $f(y_i|\mathbf{x_i},\mathbf{w})$ and $f(\mathbf{w})$. Don't try to derive an exact expression for $f(\mathbf{w}|(\mathbf{x_i}, y_i)_{i\in[n]})$, as it is very involved.

2. Define the log-likelihood for MAP as $\ell(\mathbf{w}) \triangleq \ln f(\mathbf{w}|\mathbf{x}_{i\in[n]}, y_{i\in[n]})$. Show that maximizing the MAP log-likelihood over all choices of $\mathbf{w}$ is the same as minimizing $\sum_{i=1}^n (y_i - \mathbf{w}\cdot\mathbf{x_i})^2 + \lambda\|\mathbf{w}\|_1$ where $\|\mathbf{w}\|_1 = \sum_{j=1}^d |w_j|$ and λ is a constant.

6 ℓ_1-regularization, ℓ_2-regularization, and Sparsity

You are given a design matrix X (whose i^{th} row is sample point $\mathbf{x}_i^T$) and an n-vector of labels $\mathbf{y} \triangleq [y_1 \ \dots \ y_n]^T$. For simplicity, assume X is whitened, so $X^\top X = nI$. Do not add a fictitious dimension/bias term; for input $\mathbf{0}$, the output is always 0. Let $\mathbf{x}_{*i}$ denote the i^{th} column of X.

1. Show that the cost function for ℓ_1-regularized least squares, $J_1(\mathbf{w}) \triangleq \|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|_1$ (where $\lambda > 0$), can be rewritten as $J_1(\mathbf{w}) = \|\mathbf{y}\|^2 + \sum_{i=1}^d f(\mathbf{x}_{*i}, \mathbf{w}_i)$ where $f(\cdot,\cdot)$ is a suitable function whose first argument is a vector and second argument is a scalar.

2. Using your solution to question 6.1, derive necessary and sufficient conditions for the i^{th} component of the optimizer $\mathbf{w}^*$ of $J_1(\cdot)$ to satisfy each of these three properties: $w_i^* > 0, w_i^* = 0$, and $w_i^* < 0$.

3. For the optimizer $\mathbf{w}^\#$ of the ℓ_2-regularized least squares cost function $J_2(\mathbf{w}) \triangleq \|X\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2$ (where $\lambda > 0$), derive a necessary and sufficient condition for $\mathbf{w}_i^\# = 0$, where $\mathbf{w}_i^\#$ is the ith component of $\mathbf{w}^\#$.

4. A vector is called *sparse* if most of its components are 0. From your solutions to questions 6.2 and 6.3, which of $\mathbf{w}^*$ and $\mathbf{w}^\#$ is more likely to be sparse? Why?