CS 229, Fall 2018
Problem Set #1: Supervised Learning

Due Wednesday, Oct 17 at 11:59 pm on Gradescope.

Notes: (1) These questions require thought, but do not require long answers. Please be as concise as possible. (2) If you have a question about this homework, we encourage you to post your question on our Piazza forum, at `http://piazza.com/stanford/fall2018/cs229`. (3) If you missed the first lecture or are unfamiliar with the collaboration or honor code policy, please read the policy on Handout #1 (available from the course website) before starting work. (4) For the coding problems, you may not use any libraries except those defined in the provided `environment.yml` file. In particular, ML-specific libraries such as scikit-learn are not permitted. (5) To account for late days, the due date listed on Gradescope is Oct 20 at 11:59 pm. If you submit after Oct 17, you will begin consuming your late days. If you wish to submit on time, submit before Oct 17 at 11:59 pm.

All students must submit an electronic PDF version of the written questions. We highly recommend typesetting your solutions via LaTeX. If you are scanning your document by cell phone, please check the Piazza forum for recommended scanning apps and best practices. All students must also submit a zip file of their source code to Gradescope, which should be created using the `make_zip.py` script. In order to pass the auto-grader tests, you should make sure to (1) restrict yourself to only using libraries included in the `environment.yml` file, and (2) make sure your code runs without errors using the `run.py` script. Your submission will be evaluated by the auto-grader using a private test set.

Honor code: We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down the solutions independently, and without referring to written notes from the joint session. In other words, each student must understand the solution well enough in order to reconstruct it by him/herself. In addition, each student should write on the problem set the set of people with whom s/he collaborated. Further, because we occasionally reuse problem set questions from previous years, we expect students not to copy, refer to, or look at the solutions in preparing their answers. It is an honor code violation to intentionally refer to a previous year's solutions.

1. **[40 points] Linear Classifiers (logistic regression and GDA)**

In this problem, we cover two probabilistic linear classifiers we have covered in class so far. First, a discriminative linear classifier: logistic regression. Second, a generative linear classifier: Gaussian discriminant analysis (GDA). Both the algorithms find a linear decision boundary that separates the data into two classes, but make different assumptions. Our goal in this problem is to get a deeper understanding of the similarities and differences (and, strengths and weaknesses) of these two algorithms.

For this problem, we will consider two datasets, provided in the following files:

i. `data/ds1_{train,valid}.csv`

ii. `data/ds2_{train,valid}.csv`

Each file contains m examples, one example $(x^{(i)}, y^{(i)})$ per row. In particular, the i-th row contains columns $x_0^{(i)} \in \mathbb{R}$, $x_1^{(i)} \in \mathbb{R}$, and $y^{(i)} \in \{0, 1\}$. In the subproblems that follow, we will investigate using logistic regression and Gaussian discriminant analysis (GDA) to perform binary classification on these two datasets.

(a) [10 points] In lecture we saw the average empirical loss for logistic regression:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})),$$

where $y^{(i)} \in \{0, 1\}$, $h_\theta(x) = g(\theta^T x)$ and $g(z) = 1/(1 + e^{-z})$.
Find the Hessian H of this function, and show that for any vector z, it holds true that

$$z^T H z \geq 0.$$

Hint: You may want to start by showing that $\sum_i \sum_j z_i x_i x_j z_j = (x^T z)^2 \geq 0$. Recall also that $g'(z) = g(z)(1 - g(z))$.
Remark: This is one of the standard ways of showing that the matrix H is positive semi-definite, written "$H \succeq 0$." This implies that J is convex, and has no local minima other than the global one. If you have some other way of showing $H \succeq 0$, you're also welcome to use your method instead of the one above.

(b) [5 points] **Coding problem.** Follow the instructions in `src/p01b_logreg.py` to train a logistic regression classifier using Newton's Method. Starting with $\theta = \vec{0}$, run Newton's Method until the updates to θ are small: Specifically, train until the first iteration k such that $\|\theta_k - \theta_{k-1}\|_1 < \epsilon$, where $\epsilon = 1 \times 10^{-5}$. Make sure to write your model's predictions to the file specified in the code.

(c) [5 points] Recall that in GDA we model the joint distribution of (x, y) by the following equations:

$$\begin{aligned}
p(y) &= \begin{cases} \phi & \text{if } y = 1 \\ 1 - \phi & \text{if } y = 0 \end{cases} \\
p(x|y = 0) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_0)^T \Sigma^{-1}(x - \mu_0)\right) \\
p(x|y = 1) &= \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^T \Sigma^{-1}(x - \mu_1)\right),
\end{aligned}$$

where ϕ, μ_0, μ_1, and Σ are the parameters of our model.

Suppose we have already fit ϕ, μ_0, μ_1, and Σ, and now want to predict y given a new point x. To show that GDA results in a classifier that has a linear decision boundary, show the posterior distribution can be written as

$$p(y = 1 \mid x; \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-(\theta^T x + \theta_0))},$$

where $\theta \in \mathbb{R}^n$ and $\theta_0 \in \mathbb{R}$ are appropriate functions of ϕ, Σ, μ_0, and μ_1.

(d) [7 points] For this part of the problem only, you may assume n (the dimension of x) is 1, so that $\Sigma = [\sigma^2]$ is just a real number, and likewise the determinant of Σ is given by $|\Sigma| = \sigma^2$. Given the dataset, we claim that the maximum likelihood estimates of the parameters are given by

$$\begin{aligned}
\phi &= \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \\
\mu_0 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}} \\
\mu_1 &= \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}} \\
\Sigma &= \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^T
\end{aligned}$$

The log-likelihood of the data is

$$\begin{aligned}
\ell(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma) \\
&= \log \prod_{i=1}^{m} p(x^{(i)} | y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi).
\end{aligned}$$

By maximizing ℓ with respect to the four parameters, prove that the maximum likelihood estimates of ϕ, μ_0, μ_1, and Σ are indeed as given in the formulas above. (You may assume that there is at least one positive and one negative example, so that the denominators in the definitions of μ_0 and μ_1 above are non-zero.)

(e) [3 points] **Coding problem.** In `src/p01e_gda.py`, fill in the code to calculate ϕ, μ_0, μ_1, and Σ, use these parameters to derive θ, and use the resulting GDA model to make predictions on the validation set.

(f) [5 points] For Dataset 1, create a plot of the training data with x_1 on the horizontal axis, and x_2 on the vertical axis. To visualize the two classes, use a different symbol for examples $x^{(i)}$ with $y^{(i)} = 0$ than for those with $y^{(i)} = 1$. On the same figure, plot the decision boundary found by logistic regression in part (b). Make an identical plot with the decision boundary found by GDA in part (e).

(g) [5 points] Repeat the steps in part (f) for Dataset 2. On which dataset does GDA seem to perform worse than logistic regression? Why might this be the case?

(h) **[3 extra credit points]** For the dataset where GDA performed worse in parts (f) and (g), can you find a transformation of the $x^{(i)}$'s such that GDA performs significantly better? What is this transformation?

2. **[30 points] Incomplete, Positive-Only Labels**

In this problem we will consider training binary classifiers in situations where we do not have full access to the labels. In particular, we consider a scenario, which is not too infrequent in real life, where we have labels only for a subset of the positive examples. All the negative examples and the rest of the positive examples are unlabelled.

That is, we assume a dataset $\{(x^{(i)}, t^{(i)}, y^{(i)})\}_{i=1}^m$, where $t^{(i)} \in \{0, 1\}$ is the "true" label, and where

$$y^{(i)} = \begin{cases} 1 & x^{(i)} \text{ is labeled} \\ 0 & \text{otherwise.} \end{cases}$$

All labeled examples are positive, which is to say $p(t^{(i)} = 1 \mid y^{(i)} = 1) = 1$, but unlabeled examples may be positive or negative. Our goal in the problem is to construct a binary classifier h of the true label t, with only access to the partial labels y. In other words, we want to construct h such that $h(x^{(i)}) \approx p(t^{(i)} = 1 \mid x^{(i)})$ as closely as possible, using only x and y.

Real world example: Suppose we maintain a database of proteins which are involved in transmitting signals across membranes. Every example added to the database is involved in a signaling process, but there are many proteins involved in cross-membrane signaling which are missing from the database. It would be useful to train a classifier to identify proteins that should be added to the database. In our notation, each example $x^{(i)}$ corresponds to a protein, $y^{(i)} = 1$ if the protein is in the database and 0 otherwise, and $t^{(i)} = 1$ if the protein is involved in a cross-membrane signaling process and thus should be added to the database, and 0 otherwise.

(a) [5 points] Suppose that each $y^{(i)}$ and $x^{(i)}$ are conditionally independent given $t^{(i)}$:

$$p(y^{(i)} = 1 \mid t^{(i)} = 1, x^{(i)}) = p(y^{(i)} = 1 \mid t^{(i)} = 1).$$

Note this is equivalent to saying that labeled examples were selected uniformly at random from the set of positive examples. Prove that the probability of an example being labeled differs by a constant factor from the probability of an example being positive. That is, show that $p(t^{(i)} = 1 \mid x^{(i)}) = p(y^{(i)} = 1 \mid x^{(i)})/\alpha$ for some $\alpha \in \mathbb{R}$.

(b) [5 points] Suppose we want to estimate α using a trained classifier h and a held-out validation set V. Let V_+ be the set of labeled (and hence positive) examples in V, given by $V_+ = \{x^{(i)} \in V \mid y^{(i)} = 1\}$. Assuming that $h(x^{(i)}) \approx p(y^{(i)} = 1 \mid x^{(i)})$ for all examples $x^{(i)}$, show that

$$h(x^{(i)}) \approx \alpha \quad \text{for all } x^{(i)} \in V_+.$$

You may assume that $p(t^{(i)} = 1 \mid x^{(i)}) \approx 1$ when $x^{(i)} \in V_+$.

(c) [5 points] **Coding problem.** The following three problems will deal with a dataset which we have provided in the following files:

`data/ds3_{train,valid,test}.csv`

Each file contains the following columns: x_1, x_2, y, and t. As in Problem 1, there is one example per row.

First we will consider the ideal case, where we have access to the true t-labels for training. In `src/p02cde_posonly`, write a logistic regression classifier that uses x_1 and x_2 as input features, and train it using the t-labels (you can ignore the y-labels for this part). Output the trained model's predictions on the test set to the file specified in the code.

(d) [5 points] **Coding problem.** We now consider the case where the t-labels are unavailable, so you only have access to the y-labels at training time. Add to your code in `p02cde_posonly.py` to re-train the classifier (still using x_1 and x_2 as input features), but using the y-labels only.

(e) [10 points] **Coding problem.** Using the validation set, estimate the constant α by averaging your classifier's predictions over all labeled examples in the validation set:

$$\alpha \approx \frac{1}{|V_+|} \sum_{x^{(i)} \in V_+} h(x^{(i)}).$$

Add code in `src/p02cde_posonly.py` to rescale your classifier's predictions from part (d) using the estimated value for α.

Finally, using a threshold of $p(t^{(i)} = 1 \mid x^{(i)}) = 0.5$, make three separate plots with the decision boundaries from parts (c) - (e) plotted on top of the test set. Plot x_1 on the horizontal axis and x_2 on the vertical axis, and use two different symbols for the positive ($t^{(i)} = 1$) and negative ($t^{(i)} = 0$) examples. In each plot, indicate the separating hyperplane with a red line.

Remark: We saw that the true probability $p(t \mid x)$ was only a constant factor away from $p(y \mid x)$. This means, if our task is to only rank examples (*i.e.* sort them) in a particular order (e.g, sort the proteins in order of being most likely to be involved in transmitting signals across membranes), then in fact we do not even need to estimate α. The rank based on $p(y \mid x)$ will agree with the rank based on $p(t \mid x)$.

3. **[25 points] Poisson Regression**

(a) [5 points] Consider the Poisson distribution parameterized by λ:

$$p(y; \lambda) = \frac{e^{-\lambda}\lambda^y}{y!}.$$

Show that the Poisson distribution is in the exponential family, and clearly state the values for $b(y)$, η, $T(y)$, and $a(\eta)$.

(b) [3 points] Consider performing regression using a GLM model with a Poisson response variable. What is the canonical response function for the family? (You may use the fact that a Poisson random variable with parameter λ has mean λ.)

(c) [7 points] For a training set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$, let the log-likelihood of an example be $\log p(y^{(i)}|x^{(i)}; \theta)$. By taking the derivative of the log-likelihood with respect to θ_j, derive the stochastic gradient ascent update rule for learning using a GLM model with Poisson responses y and the canonical response function.

(d) [7 points] **Coding problem.** Consider a website that wants to predict its daily traffic. The website owners have collected a dataset of past traffic to their website, along with some features which they think are useful in predicting the number of visitors per day. The dataset is split into train/valid/test sets and follows the same format as Datasets 1-3:

`data/ds4_{train,valid}.csv`

We will apply Poisson regression to model the number of visitors per day. Note that applying Poisson regression in particular assumes that the data follows a Poisson distribution whose natural parameter is a linear combination of the input features (*i.e.,* $\eta = \theta^T x$). In `src/p03d_poisson.py`, implement Poisson regression for this dataset and use gradient ascent to maximize the log-likelihood of θ.

4. **[15 points] Convexity of Generalized Linear Models**

In this question we will explore and show some nice properties of Generalized Linear Models, specifically those related to its use of Exponential Family distributions to model the output.

Most commonly, GLMs are trained by using the negative log-likelihood (NLL) as the loss function. This is mathematically equivalent to Maximum Likelihood Estimation (*i.e.,* maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood). In this problem, our goal is to show that the NLL loss of a GLM is a convex function w.r.t the model parameters. As a reminder, this is convenient because a convex function is one for which any local minimum is also a global minimum.

To recap, an exponential family distribution is one whose probability density can be represented

$$p(y;\eta) = b(y)\exp(\eta^T T(y) - a(\eta)),$$

where η is the *natural parameter* of the distribution. Moreover, in a Generalized Linear Model, η is modeled as $\theta^T x$, where $x \in \mathbb{R}^n$ are the input features of the example, and $\theta \in \mathbb{R}^n$ are learnable parameters. In order to show that the NLL loss is convex for GLMs, we break down the process into sub-parts, and approach them one at a time. Our approach is to show that the second derivative (*i.e.,* Hessian) of the loss w.r.t the model parameters is Positive Semi-Definite (PSD) at all values of the model parameters. We will also show some nice properties of Exponential Family distributions as intermediate steps.

For the sake of convenience we restrict ourselves to the case where η is a scalar. Assume $p(Y|X;\theta) \sim \text{ExponentialFamily}(\eta)$, where $\eta \in \mathbb{R}$ is a scalar, and $T(y) = y$. This makes the exponential family representation take the form

$$p(y;\eta) = b(y)\exp(\eta y - a(\eta)).$$

(a) [5 points] Derive an expression for the mean of the distribution. Show that $\mathbb{E}[Y \mid X;\theta]$ can be represented as the gradient of the log-partition function a with respect to the natural parameter η.

Hint: Start with observing that $\frac{\partial}{\partial\eta}\int p(y;\eta)dy = \int \frac{\partial}{\partial\eta}p(y;\eta)dy$.

(b) [5 points] Next, derive an expression for the variance of the distribution. In particular, show that $\text{Var}(Y \mid X;\theta)$ can be expressed as the derivative of the mean w.r.t η (*i.e.,* the second derivative of the log-partition function $a(\eta)$ w.r.t the natural parameter η.)

(c) [5 points] Finally, write out the loss function $\ell(\theta)$, the NLL of the distribution, as a function of θ. Then, calculate the Hessian of the loss w.r.t θ, and show that it is always PSD. This concludes the proof that NLL loss of GLM is convex.

Hint: Use the chain rule of calculus along with the results of the previous parts to simplify your derivations.

Remark: The main takeaways from this problem are:

- Any GLM model is convex in its model parameters.
- The exponential family of probability distributions are mathematically nice. Whereas calculating mean and variance of distributions in general involves integrals (hard), surprisingly we can calculate them using derivatives (easy) for exponential family.

5. **[25 points] Locally weighted linear regression**

(a) [10 points] Consider a linear regression problem in which we want to "weight" different training examples differently. Specifically, suppose we want to minimize

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m} w^{(i)} \left(\theta^T x^{(i)} - y^{(i)}\right)^2.$$

In class, we worked out what happens for the case where all the weights (the $w^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting.

i. [2 points] Show that $J(\theta)$ can also be written

$$J(\theta) = (X\theta - y)^T W (X\theta - y)$$

for an appropriate matrix W, and where X and y are as defined in class. Clearly specify the value of each element of the matrix W.

ii. [4 points] If all the $w^{(i)}$'s equal 1, then we saw in class that the normal equation is

$$X^T X\theta = X^T y,$$

and that the value of θ that minimizes $J(\theta)$ is given by $(X^T X)^{-1} X^T y$. By finding the derivative $\nabla_\theta J(\theta)$ and setting that to zero, generalize the normal equation to this weighted setting, and give the new value of θ that minimizes $J(\theta)$ in closed form as a function of X, W and y.

iii. [4 points] Suppose we have a dataset $\{(x^{(i)}, y^{(i)}); i = 1\ldots, m\}$ of m independent examples, but we model the $y^{(i)}$'s as drawn from conditional distributions with different levels of variance $(\sigma^{(i)})^2$. Specifically, assume the model

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2(\sigma^{(i)})^2}\right)$$

That is, each $y^{(i)}$ is drawn from a Gaussian distribution with mean $\theta^T x^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known, constants). Show that finding the maximum likelihood estimate of θ reduces to solving a weighted linear regression problem. State clearly what the $w^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

(b) [10 points] **Coding problem.** We will now consider the following dataset (the formatting matches that of Datasets 1-4, except $x^{(i)}$ is 1-dimensional):

`data/ds5_{train,valid,test}.csv`

In `src/p05b_lwr.py`, implement locally weighted linear regression using the normal equations you derived in Part (a) and using

$$w^{(i)} = \exp\left(-\frac{\|x^{(i)} - x\|_2^2}{2\tau^2}\right).$$

Train your model on the `train` split using $\tau = 0.5$, then run your model on the `valid` split and report the mean squared error (MSE). Finally plot your model's predictions on the validation set (plot the training set with blue 'x' markers and the validation set with a red 'o' markers). Does the model seem to be under- or overfitting?

(c) [5 points] **Coding problem.** We will now tune the hyperparameter τ. In `src/p05c_tau.py`, find the MSE value of your model on the validation set for each of the values of τ specified in the code. For each τ, plot your model's predictions on the validation set in the format described in part (b). Report the value of τ which achieves the lowest MSE on the `valid` split, and finally report the MSE on the `test` split using this τ-value.