6.790 Homework 3

Revision: 10/7/24 8:12AM **Solutions**

Questions 1 is optional. The remaining problems are required. All the questions are written (don't include running code) and go over modeling choices, decision boundaries, uncertainty estimation, and optimization. There are some hints in the blue boxes that are supposed to help you in your solution, but you can choose to disregard them.

Please hand in your work via Gradescope via the link at https://gradml.mit.edu/info/homeworks/. If you were not added to the course automatically, please use Entry Code R7RGGX to add yourself to Gradescope. Make sure to assign the problems to the corresponding pages in your solution when submitting via Gradescope.

1. Latex is not required, but if you are hand-writing your solutions, please write clearly and carefully. You should include enough work to show how you derived your answers, but you don't have to give careful proofs.

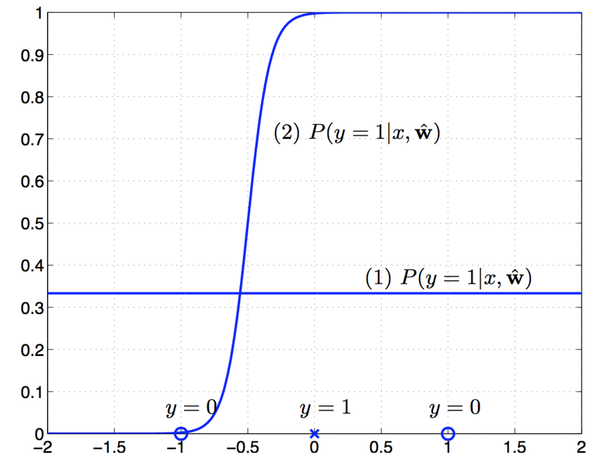
2. Homework is due on Tuesday October 8 at 11PM.

3. Lateness and extension policies are described at https://gradml.mit.edu/info/class_policy/.

Contents

Solution: Don't look at the solutions until you have tried your absolute hardest to solve the problems. This is especially true for optional problems that you didn't work on—it's a good idea to come back to them when studying for exams.

1 Optional warm-up: Logistic regression: basic intuition with one-dimensional data



1. Consider a simple one dimensional logistic regression model

$$P(y=1 \mid x, w) = \sigma(w_0 + w_1 x)$$

where $\sigma(z) = (1+\exp(-z))^{-1}$ is the logistic function.

The figure above shows two possible conditional distributions $P(y=1 \mid x, w)$, viewed as a function of x, that we can get by changing the parameters w.

Assume we have a data set $\mathcal{D} = \{(-1,0),(0,1),(1,0)\}$.

(a) How many classification errors does hypothesis 1 make?

(b) How many classification errors does hypothesis 2 make?

(c) Which of these two hypotheses assigns a higher likelihood to the data?

(d) If your loss function for predictions was

$$L(g,a) = \begin{cases} 0 & \text{if } g=a \\ 1 & \text{if } g=1 \text{ and } a=0 \\ 10 & \text{if } g=0 \text{ and } a=1 \end{cases}$$

- What output would you predict for $x=-1$ when conditional (1) is the result of your learning?

- What output would you predict for $x = -1$ when conditional (2) is the result of your learning?

Solution: 1. Conditional (1) makes (1) classification errors, Conditional (2) makes (1) classification errors.

2. (1). Because:

$$P_1 = 0.67 \times 0.33 \times 0.67 = 0.07, P_2 = 1 \times 1 \times 0.01 = 0.01, P_1 > P_2$$

3. When $P(y = 1|x, w) > 1/11$, predict 1, otherwise, predict 0. See week 1 Exercise, question 6 for the detail.

2 Noisy sensor [10 points]

2. You are trying to solve a perceptual recognition problem for a robot in simulation, and you find that you can find a function $f(x) = \theta^T x$ that has the property that if $f(x) > 0$ then the robot can fit through the door it is looking at and if it's less than 0 then the robot can't fit.

   Now you are going to deploy this system in the real world, but find that there are some perturbations in the world that make your prediction rule less accurate than it was. We can model the real-world process as

$$y = \begin{cases} 1 & \text{if } \theta^T x + \epsilon \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$$

   You gather a lot of data and find that the noise ϵ seems to have a distribution with pdf $\frac{e^{-\epsilon}}{(1+e^{-\epsilon})^2}$

   Obviously you can't really apply this predictor in the real world, because you don't know the ϵ value! But, given a new input x, you can compute $f(x)$ and you think the higher that value is, the more sure you ought to be that the actual y value associated with it will be 1.

   Under the assumptions above, what is $p(y = 1 \mid f(x))$? Your final answer can be written in terms of $f(x), \theta, x$ and should not include integrals.

   Hint: it might be helpful (but not strictly necessary) to look at the logistic distribution in order to avoid complicated integral calculations.

Solution:

We are given that:

$$y = \begin{cases} 1 & \text{if } \theta^T x + \epsilon \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$$

This implies that $y = 1$ when $\epsilon \geqslant -\theta^T x$ and $y = 0$ otherwise.

To find $p(y = 1 \mid f(x))$, we want to determine the probability that $\epsilon \geqslant -\theta^T x = -f(x)$. Mathematically, this is:

$$p(y = 1 \mid f(x)) = p(\epsilon \geqslant -f(x))$$

We are given that the noise ϵ has a probability density function (pdf) of:

$$p(\epsilon) = \frac{e^{-\epsilon}}{(1 + e^{-\epsilon})^2}$$

This is the pdf of the *logistic distribution* with a mean of 0 and a scale parameter of 1. The cumulative distribution function (CDF) of a logistic distribution is:

$$P(\epsilon \leqslant f(x)) = \frac{1}{1 + e^{-f(x)}}$$

Thus, the probability that $\epsilon \geqslant -f(x)$ is:

$$p(\epsilon \geqslant -f(x)) = 1 - P(\epsilon < -f(x)) = 1 - \frac{1}{1 + e^{f(x)}} = \frac{1}{1 + e^{-f(x)}} = \sigma(f(x))$$

This is the *logistic function*, which is commonly used in logistic regression models to map the output of a linear function to a probability between 0 and 1.

The solution is:

$$p(y = 1 \mid f(x)) = \frac{1}{1 + e^{-f(x)}} = \sigma(f(x))$$

where σ is the sigmoid function.

Alternative solution:

If one doesn't recognize that ϵ follows a logistic distribution, we can instead explicitly solve an integral to get the same solution.

The probability that $y = 1$, i.e., $\epsilon \geqslant -f(x)$, is expressed as:

$$p(y = 1 \mid f(x)) = p(\epsilon \geqslant -f(x)) = \int_{-f(x)}^{\infty} \frac{e^{-\epsilon}}{(1 + e^{-\epsilon})^2}\, d\epsilon$$

We can solve it using a change of variables and get the same final solution as above.

3 How wrong? [15 points]

3. You are the chief data scientist at Yoyodyne. Your job is to predict the lifetime, in days, of your new Dynamo fan product. You have run 200 dynamo fans to exhaustion in different situations, drawn from the same environmental distribution. For each you have collected information about the humidity, temperature, particulate density, and energy quality of the location it was used in, as well as the working lifetime of the fan in days. You divide the data into a training set of size 180 and a validation set of size 20, and use 6.790 methods on the training set (without peeking at the validation set...even once!) to come up an awesome hypothesis h.

   (You can either do parts a, b, and c **or** do just part d).

   (a) (5 points) You evaluate h on your validation set, using squared loss, and end up with a mean loss of $M_n = 50$ with a standard deviation of $\hat{\sigma} = 10$.

   You are completely convinced that you know the standard deviation and that the true $\sigma = 10$*, as well.*

   Provide a bound on the probability that the true risk of h is greater than 60.

   Feel free to use the following application of Chebychev's inequality:

   Proposition: Let $X_1, \ldots, X_n$ be iid with finite mean μ and variance σ^2, and let $M_n = (X_1 + \ldots + X_n)/n$. Then

$$P(|M_n - \mu| \geqslant \epsilon) \leqslant \frac{\sigma^2}{n\epsilon^2}$$

Solution:
We are given:

- Sample mean loss on the validation set: $M_n = 50$
- Standard deviation of the loss: $\sigma = 10$
- Sample size: $n = 20$
- We need to bound the probability that the true risk μ is greater than 60.

We will use Chebyshev's inequality:

$$P(|M_n - \mu| \geqslant \epsilon) \leqslant \frac{\sigma^2}{n\epsilon^2}$$

where:

- $\epsilon = 60 - 50 = 10$
- $\sigma^2 = 100$ (variance of individual observations)

We are interested in $P(\mu - M_n \geqslant 10)$ which is less than or equal to $P(|M_n - \mu| \geqslant 10)$. Now, applying Chebyshev's inequality:

$$P(|M_n - \mu| \geqslant 10) \leqslant \frac{100}{20 \times 10^2} = \frac{1}{20} = 0.05$$

Thus, the probability that the true risk μ is greater than 60 is bounded by:

$$P(\mu > 60) \leqslant 0.05$$

(b) (5 points) Now, the marketing department wants to make a web site that allows customers to input the parameters of their intended deployment, and then outputs the result of applying h to that situation to predict the Dynamo's lifetime. They call to ask you what kind of claim it can make about the reliability your predictions!

In particular, they want to say that, 99 times out of 100, the prediction is within some δ of the true lifetime. Assuming that *the standard deviation of the distribution of the prediction error* is $\sigma_{\text{err}} = 10$, what is the smallest value of δ you can responsibly claim? (It is fine to assume that the mean prediction error is 0.)

Here's the basic version of Chebychev's inequality (you should be able to figure out the relationship between this and the previous version), which you might find helpful.

Proposition: Let X be a random variable with finite mean μ and variance σ^2. Then

$$P(|X - \mu| \geqslant k\sigma) \leqslant \frac{1}{k^2}$$

Is the *standard deviation of the prediction error* the same as the *standard error* (described in lecture 7)? Explain.

Solution: We need to determine the smallest value of δ such that, with 99

$$P(|h(x) - y| \leqslant \delta) \geqslant 0.99$$

We want to ensure that:

$$P(|h(x) - y| \geqslant \delta) \leqslant 0.01$$

In this version of Chebyshev's inequality, the probability we will achieve our error bound is $1/k^2$ which needs to be less than or equal to 0.01, so we'll let $k = 10$.

And so, the guarantee we can make is that

$$P(|h(x) - y| \geqslant 10 \cdot 10) \leqslant 0.01$$

Thus, you can claim that "99 times out of 100, the predicted Dynamo lifetime is within approximately 100 days of the true lifetime."

The *standard error* is the standard deviation of our *estimate of the mean* of the distribution of interest. It is influenced by the sample standard deviation, but it is not itself an estimate of the standard deviation of the distribution we are sampling from. In parts 1 and 2 of this problem, we assumed a known standard deviation of the distribution we are sampling from. If we hadn't known it, we couldn't have used Chebychev's inequality and would have had to use something looser like Hoeffding's inequality. Or! See the last part of this question.

The original version of the problem was slightly different, below is the solution to that version.

We need to determine the smallest value of δ such that, with 99

$$P(|h(x) - y| \leqslant \delta) \geqslant 0.99$$

Chebyshev's inequality states:

$$P(|M_n - \mu| \geqslant \delta) \leqslant \frac{\sigma^2}{n\delta^2}$$

We want to ensure that:

$$P(|M_n - \mu| \geqslant \delta) \leqslant 0.01$$

Substituting the given values:

- $\sigma^2 = 100$ (since $\sigma = 10$)
- $n = 20$

$$\frac{100}{20\delta^2} \leqslant 0.01$$

Solving for δ:

$$\delta^2 \geqslant \frac{5}{0.01} = 500$$

$$\delta \geqslant \sqrt{500} \approx 22.36$$

Thus, you can claim that "99 times out of 100, the predicted Dynamo lifetime is within approximately 22.36 days of the true lifetime."

(c) (5 points) Thinking back, you wonder what would have happened if you had changed the percentage of your data you used for training vs testing. Qualitatively, how would you expect your answers to the previous two questions to change, if you had used 50% of your data for training and 50% for validation, compared to your original setup, where 90% of the data was used for training and 10% for validation?

Solution:

1. Probability bound on true risk (part a): A larger validation set reduces the variance in the estimate of the true risk, leading to a tighter (smaller) bound on the probability that the true risk exceeds 60.

   But! Using less of the data for constructing the hypothesis might result in making worse predictions with higher risk.

2. The smallest δ you can claim with 99% confidence might increase, as the model built from a smaller training set could be less accurate. Therefore, the predictions might be farther from the true lifetime, requiring a larger δ to maintain the same confidence level.

(d) (0 points) (Alternative to a,b,c (15 points): interesting if you like statstics!) Equation 2 of the paper *Multivariate Chebyshev Inequality with Estimated Mean and Variance* by Stellato, Van Parys, and Goulart, `https://mitsloan.mit.edu/shared/ods/documents?PublicationDocumentID=4759` contains a bound (due to Saw et al) on the error of using the sample mean as a prediction for the $n + 1$st value, *as a function of the sample mean, sample standard deviation, and sample size*. This is pretty cool! Play with it and see if you can use it to estimate an error of your next prediction, given a sample mean of 50 and *sample* standard deviation of 10. You might need to increase n to get a reasonable bound.

4 Why not regression? [10 points]

4. Jan is trying to do classification: $y^{(i)} \in \{0, 1\}$ and $x^{(i)} \in \mathbb{R}$. But he slept through all of the classification lectures, so he decides to solve classification using regression. That is, he ignores the fact that $y^{(i)}$ is binary, and fits a linear regression function via least squares. The resulting regression function is:

$$\hat{y} = f(x; w) = w_0 + x w_1$$

Jan uses the decision rule: label 1 if $f(x; w) > 1/2$; and label 0 otherwise.

Suppose the training data is linearly separable. Is Jan's decision rule (with associated regression function) guaranteed to classify the training data without error?

1. Yes. Provide a short argument:
2. No. Provide a counterexample:

Solution: No.

Key points: Full credit given for counterexample (does not need to be explicitly spelled out with all the constants, see below). Partial credit for correct answer without counterexample.

Consider the dataset:

$(-1/2, 0)$ repeated n times
$(0, 1)$
$(1, 1)$ repeated n times

The data is linearly separable in X (for example, by $X = \frac{-1}{4}$). As $n \to \infty$, the best fit line $\to \frac{2x}{3} + \frac{1}{3}$. Thus, for large n, the prediction at $x = 0$ will be class 0 since $\frac{2*0}{3} + \frac{1}{3} = 1/3 < 1/2$, so the middle point (which is of class 1) will be misclassified by Rick's decision rule.

5 Modeling the distribution versus training a classifier [15 points]

5. (Murphy 10.3)

Suppose we train the following binary classifiers via maximum likelihood.

(a) GaussI: A generative classifier, where the class conditional densities are Gaussian, with both covariance matrices set to I (identity matrix), i.e., $p(x \mid y = c) = \mathcal{N}(x \mid \mu_c, I)$. We assume $p(y)$ is uniform.
(b) GaussX: as for GaussI, but the covariance matrices are unconstrained, i.e., $p(x \mid y = c) = \mathcal{N}(x \mid \mu_c, \Sigma_c)$.
(c) LinLog: A logistic regression model with linear features.
(d) QuadLog: A logistic regression model, using linear and quadratic features (i.e., polynomial basis function expansion of degree 2).

After training we compute the performance of each model M on the training set as follows:

$$L(M) = \frac{1}{n}\sum_{i=1}^{n} \log p(y^{(i)} \mid x^{(i)}, \hat{\theta}, M)$$

Note that this is the conditional log-likelihood $p(y \mid x, \hat{\theta}, M)$ and not joint log-likelihood $p(y, x \mid \hat{\theta}, M)$. We now want to compare the performance of each model. We will write $L(M) \leqslant L(M')$ if model M must have lower or equal log likelihood on the training set than M', for any training set (in other words, M is worse than M', at least as far as training set logprob is concerned). For each of the following model pairs, state whether $L(M) \leqslant L(M'), L(M) \geqslant L(M')$, or whether no such statement can be made (i.e., M might sometimes be better than M' and sometimes worse); also, for each question, briefly (1-2 sentences) explain why.

(a) (3 points) GaussI, LinLog
(b) (3 points) GaussX, QuadLog
(c) (3 points) LinLog, QuadLog
(d) (3 points) GaussI, QuadLog

(e) (3 points) Now suppose we measure performance in terms of the average misclassification rate on the training set:

$$R(M) = \frac{1}{n}\sum_{i=1}^{n} I(y^{(i)} \neq \hat{y}(x^{(i)}))$$

where $\hat{y}(x^{(i)})$ is the predicted y for $x^{(i)}$. Is it true in general that $L(M) > L(M')$ implies that $R(M) < R(M')$? Explain why or why not.

Solution:

(a) GaussI $\leqslant$ LinLog. Both have logistic (sigmoid) posteriors $p(y \mid x, w) = \sigma(y|w^T x)$, but LinLog is the logistic model which is trained to maximize $p(y \mid x, w)$. (GaussI may have high joint $p(y, x)$, but this does not necessarily mean $p(y \mid x)$ is high; LinLog can achieve the maximum of $p(y \mid x)$, so will necessarily do at least as well as GaussI). Note that Logistic regression (LinLog) directly models the conditional probability $p(y|x)$, optimizing it via maximum likelihood.

(b) GaussX $\leqslant$ QuadLog. Both have logistic posteriors with quadratic features, but QuadLog is the model of this class maximizing the average log probabilities.

(c) LinLog $\leqslant$ QuadLog. Logistic regression models with linear features are a subclass of logistic regression models with quadratic functions. The maximum from the superclass is at least as high as the maximum from the subclass.

(d) GaussI $\leqslant$ QuadLog. Follows from above inequalities.

(e) Although one might expect that higher log likelihood results in better classification performance, in general, having higher average $\log p(y \mid x)$ does not necessarily translate to higher or lower classification error. For example, consider linearly separable data. We have L(LinLog) > L(GaussI), since maximum likelihood logistic regression will set the weights to infinity, to maximize the probability of the correct labels (hence $p(y^{(i)} \mid x^{(i)}, \hat{w}) = 1$ for all i). However, we have R(LinLog) = R(GaussI), since the data is linearly separable. (The GaussI model may or may not set σ very small, resulting in possibly very large class conditional pdfs; however, the posterior over y is a discrete pmf, and can never exceed 1.)

As another example, suppose the true label is always 1 (as opposed to 0), but model M always predicts $p(y = 1 \mid x, M) = 0.49$. It will always misclassify, but it is at least close to the decision boundary. By contrast, there might be another model M' that predicts $p(y = 1 \mid x, M') = 1$ on even-numbered inputs, and $p(y = 1 \mid x, M') = 0$ on odd-numbered inputs. Clearly $R(M') = 0.5 < R(M) = 1$, but $L(M') = -\infty < L(M) = \log(0.49)$.

6 Naive Bayes Classification [15 points]

Consider a K-class classification problem where the input vector $x = (x_1, x_2, \dots, x_M)$ consists of M binary components, where each $x_i \in \{0, 1\}$, and the output y is a one-hot vector of length K, encoding the class label.

Consider the following generative model:

- The probability of y is given by $p(y = k)$ for each class k, and $\sum_{k=1}^{K} p(y = k) = 1$.
- The features x_i are conditionally independent given the class y, so the joint probability can be factorized as:

$$p(x, y) = p(y) \prod_{i=1}^{M} p(x_i \mid y)$$

This represents an example of the naive Bayes model. Assuming you have access to labeled training data, we can use Maximum Likelihood Estimation (MLE) to estimate the parameters:

- The prior probability estimate is $\hat{p}(y = k) = \frac{N_k}{N}$, where N_k is the number of training examples that belong to class k and N is the total number of training examples.
- The MLE estimate for the conditional probability is $\hat{p}(x_i = 1 \mid y = k) = \frac{N_{ki}}{N_k}$, where N_{ki} is the number of examples in class k where $x_i = 1$, and N_k is the number of examples in class k.

(a) (5 points) Suppose we are given a new binary feature vector $x = (x_1, x_2, \dots, x_M)$, and we want to predict the class label. The prediction is made using the posterior probability $p(y = k \mid x)$. Show that the posterior probability can be written in the form:

$$p(y = k \mid x) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}$$

where

$$a_k = \log p(x, y = k)$$

Solution: The posterior probability is given by Bayes' theorem:

$$p(y = k \mid x) = \frac{p(x \mid y = k)p(y = k)}{p(x)}$$

Since we are interested in comparing the class probabilities, we can express this as:

$$p(y = k \mid x) = \frac{p(x \mid y = k)p(y = k)}{\sum_{j=1}^{K} p(x \mid y = j)p(y = j)}$$

We can rewrite the posterior as:

$$p(y = k \mid x) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}$$

where

$$a_k = \log\left(p(x \mid y = k)p(y = k)\right)$$

(b) (10 points) Show that a_k is a linear function of the components of x.

Solution: We have:

$$a_k = \log\left(p(x \mid y=k)p(y=k)\right)$$

Using the conditional independence assumption, we can factor $p(x \mid y=k)$ as:

$$p(x \mid y=k) = \prod_{i=1}^{M} p(x_i \mid y=k)$$

Thus, we can express a_k as:

$$a_k = \log p(y=k) + \sum_{i=1}^{M} \log p(x_i \mid y=k)$$

For each binary feature x_i, we can write $p(x_i \mid y=k)$ as:

$$p(x_i \mid y=k) = p(x_i = 1 \mid y=k)^{x_i} \cdot p(x_i = 0 \mid y=k)^{1-x_i}$$

Substituting this into the expression for a_k:

$$\begin{aligned} a_k &= \log p(y=k) + \sum_{i=1}^{M} \left[x_i \log p(x_i=1 \mid y=k) + (1-x_i)\log p(x_i=0 \mid y=k)\right] \\ &= \log \frac{N_k}{N} + \sum_{i=1}^{M} \left[x_i \log \frac{N_{ki}}{N_k} + (1-x_i)\log\left(1 - \frac{N_{ki}}{N_k}\right)\right] \end{aligned}$$

This shows that a_k is a linear function of the binary components x_i, as it is a sum of terms proportional to x_i.

7 Optimizing logistic regression [15 points]

6. In this problem, suppose you were hypothetically provided with a classification dataset `dataset.csv` for binary classification. Suppose the file `dataset.csv` contains a matrix of size 1672×65, where the last column of the matrix is the class label. Thus, the data matrix X is of dimension $n \times d$, where $n = 1672$ and $d = 64$ (*Remark*: The digits are 8×8 pixel images that have been turned into vectors of length 64), and the label vector $y \in \{0,1\}^n$ is the last (hence, 65th) column.

   We will use this data with regularized logistic regression that has the objective function:

$$R(w) := \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell_i(w)}_{\ell(w)} + \underbrace{\frac{\lambda}{2}\|w\|_2^2}_{r(w)}, \tag{1}$$

   where $\ell_i(w) := \log(1 + e^{-y_i w^\top x^i})$ and $r(w)$ is the regularizer. **Note:** (1) expects $y_i \in \{-1, 1\}$. Moreover, we have designed the dataset to be linearly separable, so actually a loss value close to 0 should be achievable empirically (which corresponds to 100% training accuracy) when $\lambda = 0$ is used.

In this question, we will recall gradient descent (GD) and stochastic gradient descent (SGD) for optimizing the regularized loss function (1) with the choice $\lambda > 0$.

(a) (5 points) Provide an expression for the gradient of the objective function, i.e., an expression for $\nabla R(w)$.

Solution: $\frac{1}{n}\sum_{i=1}^{n} z_i x^i + \lambda w$, where $z_i = \frac{-y_i}{1+e^{y_i w^\top x^i}}$.

Key points: Take the gradient of each component $\ell_i(w)$ of $\ell(w)$, and of $r(w) = \frac{\lambda}{2}\|w\|_2^2$.

First, $\nabla r(w) = \nabla \frac{\lambda}{2} w^\top w = \lambda w$. Then, for any i and j,

$$\frac{\partial}{\partial w_j}\ell_i(w) = \frac{\partial}{\partial w_j}\log(1 + e^{-y_i w^\top x^i}) \tag{2}$$

$$= \frac{-y_i x_j^i}{1 + e^{-y_i w^\top x^i}} e^{-y_i w^\top x^i} \tag{3}$$

$$= \frac{-y_i}{1 + e^{y_i w^\top x^i}} x_j^i \tag{4}$$

(note the cancellation of $e^{-y_i w^\top x^i}$ and $1/e^{-y_i w^\top x^i} = e^{y_i w^\top x^i}$) and hence, letting $z_i = \frac{-y_i}{1+e^{y_i w^\top x^i}}$, we have

$$\nabla \ell_i(w) = z_i x^i \tag{5}$$

Then, we put it together:

$$\nabla R(w) = \nabla\left(\frac{1}{n}\sum_{i=1}^{n} \ell_i(w) + r(w)\right) = \frac{1}{n}\sum_{i=1}^{n} z_i x^i + \lambda w \tag{6}$$

(b) (5 points) In SGD, we use a stochastic gradient $g(w)$ instead of $\nabla R(w)$. We compute $g(w)$ by selecting a batch of b data points $\{x^{i_1}, \ldots, x^{i_b}\}$, where each i_j is sampled uniformly with replacement from $\{1, 2, \ldots, n\}$, and each x^{i_j} is the corresponding row from the data matrix X. Provide an expression for the resulting (mini-batch) stochastic gradient .

Solution:
The stochastic gradient assumes the form

$$g(w) := \frac{1}{b}\sum_{j=1}^{b} z_{i_j} x^{i_j} + \lambda w, \tag{7}$$

where $z_i = \frac{-y_i}{1+e^{y_i w^\top x^i}}$.

Key points: The stochastic gradient is just the gradient from the previous part (4.1(a)) on a randomly-chosen subset of data points; reuse the expression from the previous part but with $x^{i_1}, \ldots, x^{i_b}$ rather than $x^1, \ldots, x^n$.

(c) (5 points) Show that the gradient you derived above is unbiased, i.e., $\mathbb{E}[g(w)] = \nabla R(w)$, where the expectation is computed over the randomness of the batch.

Solution: *Key points:* Take an expectation of the expression from 4.1(b) over uniformly-random points and derive the expression from 4.1(a).

Since each i_j is uniformly randomly chosen from $[n]$, for any j,

$$\mathbb{E}[z_{i_j} x_{i_j}] = \frac{1}{n}\sum_{i=1}^{n} z_i x^i \tag{8}$$

Then by linearity of expectation, we have

$$\begin{aligned}
\mathbb{E}[g(w)] &= \frac{1}{b}\sum_{j=1}^{b} \mathbb{E}[z_{i_j} x_{i_j}] + \lambda w && (9)\\
&= \frac{b}{b}\frac{1}{n}\sum_{i=1}^{n} z_i x^i + \lambda w && (10)\\
&= \frac{1}{n}\sum_{i=1}^{n} z_i x^i + \lambda w && (11)\\
&= \nabla R(w) && (12)
\end{aligned}$$

from question 4.1(a).

8 Softmax [20 points]

An approach to multi-class classification is to use a generalized version of the logistic model. Let $x = [x_1, x_2, \ldots, x_d]$ be an input vector, and suppose we would like to classify into k classes; that is, the output y can take a value in $1, \ldots, k$. The softmax generalization of the logistic model uses $k(d+1)$ parameters $\theta = (\theta_{ij}), i = 1, \ldots, k, j = 0, \ldots, d$, which define the following k intermediate values:

$$\begin{aligned}
z_1 &= \theta_{10} + \sum_j \theta_{1j} x_j \\
&\cdots \\
z_i &= \theta_{i0} + \sum_j \theta_{ij} x_j \\
&\cdots \\
z_k &= \theta_{k0} + \sum_j \theta_{kj} x_j
\end{aligned}$$

The classification probabilities under the softmax model are:

$$\Pr(y = i \mid x; \theta) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \ .$$

(a) (5 points) Show that when $k = 2$ the softmax model reduces to the logistic regression model. That is, show how both give rise to the same classification probabilities $\Pr(y \mid x)$. Do this by constructing an explicit transformation between the parameters: for any given set of $2(d+1)$ softmax parameters, show an equivalent set of $(d+1)$ logistic parameters.

Hint: you can write the posterior of the logistic model given its parameter θ', and find the relationship between θ' and θ

Solution: The posterior of a logistic model with weights θ'

$$P(Y = 1|x; \theta) = \frac{1}{1 + e^{-z'}}$$

where $z' = \theta'_0 + \sum_j \theta'_j x_j$. The posterior of the softmax model when $k = 2$,

$$P(Y = 1|x; \theta) = \frac{e^{z_1}}{e^{z_1} + e^{z_2}}$$

equating the two

$$\begin{aligned}
\frac{1}{1 + e^{-z'}} &= \frac{e^{z_1}}{e^{z_1} + e^{z_2}} \\
e^{z_1} + e^{z_2} &= e^{z_1} + e^{z_1} e^{-z'} = e^{z_1} + e^{z_1 - z'} \\
e^{z_1 - z'} &= e^{z_2} \\
z' &= z_1 - z_2
\end{aligned}$$

If $\theta'_j = \theta_{1j} - \theta_{2j}$ for each j, the softmax model reduces to the logistic model.

(b) (5 points) What type of decision boundary is possible for a softmax model? (e.g. linear, quadratic etc.)

Solution: Only linear decision boundaries are possible.
Explanation: A softmax model, like logistic regression, is a linear classifier. Each class i is associated with a linear decision function $z_i = \theta_{i0} + \sum_j \theta_{ij} x_j$. The decision boundaries between any two classes are determined by the hyperplane where $z_i = z_j$, which corresponds to a linear decision boundary in the input space. Therefore, softmax can only create linear decision boundaries.

(c) (10 points) A stochastic gradient ascent learning rule for softmax is given by:

$$\theta_{ij} \leftarrow \theta_{ij} + \alpha \sum_t \frac{\partial}{\partial \theta_{ij}} \log \Pr(y^t \mid x^t; \theta) \ ,$$

where (x^t, y^t) are the training examples. We would like to rewrite this rule as a delta rule. In a delta rule the update is specified as a function of the difference between the target and the prediction. In our case, our target for each example will actually be a vector $y^t = (y^t_1, \ldots, y^t_k)$ where $y^t_i = 1$ if $y^t = i$ and 0 otherwise.

Our prediction will be a corresponding vector of probabilities:

$$\hat{y}^t = (\Pr(y = 1 \mid x^t; \theta), \ldots, \Pr(y = k \mid x^t; \theta))$$

Calculate the derivative above (i.e. $\frac{\partial}{\partial \theta_{ij}} \log \Pr(y^t \mid x^t; \theta)$) and rewrite the update rule as a function of $y - \hat{y}$

Hint: it might be helpful to calculate $\frac{\partial z_i}{\partial \theta_{ij}}$

Hint: it might be helpful to consider two cases: $y = i$ and $y \neq i$

Solution: Sometimes it is easier to calculate derivatives in log-scale.

$$\log P(y = i|x) = z_i - \log \sum_l e^{z_l}$$
$$\frac{\partial z_i}{\partial \theta_{ij}} = x_j$$

Two cases

$$y = i: \ \frac{\partial \log P(y = i)}{\partial \theta_{ij}} = 1 \cdot x_j - \frac{e^{z_i}}{\sum_l e^{z_l}} x_j = y_i x_j - \hat{y}_i x_j$$
$$y \neq i: \ \frac{\partial \log P(y = k \neq i)}{\partial \theta_{ij}} = 0 \cdot x_j - \frac{e^{z_i}}{\sum_l e^{z_l}} x_j = y_i x_j - \hat{y}_i x_j$$

Combining them in the vector form

$$\frac{\partial \log P(y^t|\mathbf{x}^t)}{\partial \theta_{ij}} = y_i^t x_j - \hat{y}_i^t x_j = (y^t - \hat{y}^t)^T \mathbf{x}^t$$

Therefore, the update rule is

$$\theta \leftarrow \theta + \alpha \sum_t (y^t - \hat{y}^t)^T \mathbf{x}^t$$