Midterm Test: Questions and Solutions

Name:

Student Number:

Closed book. 1-page, single-sided cheat sheet allowed (8.5x11in, no more than 6000 characters, 12pt font or larger if typed). No other aids allowed.

50 minutes. 7 pages. 4 questions. 46 points. Write all answers on the test booklet, using the backs of pages if necessary. The last page is blank. Clear, concise answers will receive more marks than long, rambling ones. Unless specified otherwise, all answers should be justified. Good luck!

I don’t know policy: If you do not know the answer to a question (or part), and you write “I don’t know”, you will receive 20% of the marks of that question (or part). If you just leave a question blank with no such statement, you will get 0 marks for that question.
1. (16 points total) True or False.

For each of the following statements, state whether it is true or false without giving an explanation (2 points each):

(a) Because of the non-linear sigmoid function, logistic regression classifiers can have a non-linear decision boundary.

   ANSWER: FALSE

(b) If a function underfits the data, then both the training error and the test error are too large.

   ANSWER: TRUE

(c) For nearest-neighbour classifiers, the decision boundary is piecewise linear.

   ANSWER: TRUE

(d) If $\sigma(z)$ is the sigmoid function, then $\partial \sigma(z) / \partial z \approx 0$ when $|z|$ is very large.

   ANSWER: TRUE

(e) Ideally, the precision of a classifier is high and the recall is low.

   ANSWER: FALSE

(f) A sigmoid function can be viewed as a smooth version of a step function.

   ANSWER: TRUE

(g) Multi-class logistic regression applied to a 2-class problem is equivalent to binary logistic regression.

   ANSWER: TRUE

(h) Regularization decreases the training error of a classifier.

   ANSWER: FALSE
2. (10 points) Over and Underfitting

To prevent overfitting during training, we often add a regularization term $\alpha \sum_k w_k^2$ to the loss function, where $\alpha > 0$ and the sum is over all weights, $w_k$, that we are trying to learn.

Draw two curves: training error vs $\alpha$, and test error vs $\alpha$. Draw both curves on a single pair of axes in which the horizontal axis is $\alpha$. Indentify the curves clearly. On the horizontal axis, indicate areas where underfitting occurs, and areas where overfitting occurs. Also indicate the best value of $\alpha$. 
3. (10 points) *Numpy Programming*

Recall from Assignment 1 that for linear least squares, the gradients of the loss function are given by

\[
\frac{\partial l(w_0, w)}{\partial w} = 2Z^T(y - t)
\]

\[
\frac{\partial l(w_0, w)}{\partial w_0} = 2\vec{1}^T(y - t)
\]

Here $Z$ is the feature matrix, $y$ is the vector of predicted values, $t$ is the vector of target values, $w$ is the weight vector, and $w_0$ is the bias weight. In addition, $y = w_0 \vec{1} + Zw$, where $\vec{1}$ is a vector of 1’s. All vectors are column vectors. Finally, recall that one iteration of gradient descent is given by the following two statements:

\[
w = w - \lambda \frac{\partial l(w_0, w)}{\partial w}
\]

\[
w_0 = w_0 - \lambda \frac{\partial l(w_0, w)}{\partial w_0}
\]

where $\lambda$ is the learning rate.

Write a NumPy function `bar(X,T,N,1rate)` that performs $N$ iterations of gradient descent and returns the final weight vector $w$ and bias weight $w_0$. Here, $X$ and $T$ contain the training data, where $X$ is a matrix of input values, and $T = t$ is the vector of target values. $1rate = \lambda$ is the learning rate. You can assume that $w$ and $w_0$ are initialized by the following command:

\[
w, w_0 = \text{init}()
\]

and that the feature matrix is computed by the following command:

\[
Z = \text{foo}(X)
\]

(You do not have to write `foo` and `init`.) Your function should have only one loop (and no nested loops) and should begin with the following line:

```python
import numpy as np
```
This page is for answers and rough work

**ANSWER:**

```python
def bar(X,T,N,lrate):
    import numpy as np
    w0, w = init()
    Z = foo(X)
    for n in range(N):
        Y = w0 + Z*w
        Err = Y - T
        w = w - 2*lrate*np.matmul(Z.T,Err)
        w0 = w0 - 2*lrate*np.sum(Err)
    return w, w0
```
4. (10 points) Gradients

Recall that multi-class logistic regression uses the softmax function, which is given by the following equation:

\[ y_k = \frac{e^{z_k}}{\sum_i e^{z_i}} \]

To perform gradient descent, we need the gradients of this function. Prove the following:

\[ \frac{\partial y_k}{\partial z_j} = \delta_{jk}y_j - y_jy_k \]

where \( \delta_{jk} = 1 \) if \( j = k \) and 0 otherwise.

Hint: Consider two cases: \( j = k \) and \( j \neq k \).

ANSWER: For convenience, let \( u = \sum_i e^{z_i} \). Then \( y_j = e^{z_j}/u \) and \( \partial u/\partial z_j = e^{z_j} \), for all \( j \).

Case (1): Suppose \( j \neq k \). We must prove that \( \partial y_k/\partial z_j = -y_jy_k \).

\[
\frac{\partial y_k}{\partial z_j} = \frac{\partial(e^{z_k}/u)}{\partial z_j} = -\left(\frac{e^{z_k}/u^2}{u}\right) \frac{\partial u}{\partial z_j} \quad \text{by the chain rule}
\]

\[ = -\left(\frac{e^{z_k}/u^2}{u}\right) e^{z_j} \quad \text{as noted above}
\]

\[ = \left(\frac{e^{z_k}/u}{u}\right)(e^{z_j}/u) \]

\[ = -y_k y_j \quad \text{as noted above}
\]

Case (2): Suppose \( j = k \). We must prove that \( \partial y_k/\partial z_k = y_k - y_k^2 \). For convenience, let \( v = e^{z_k} \) and \( w = 1/u \). Then, \( y_k = vw \) and

\[
\frac{\partial y_k}{\partial z_k} = \frac{\partial(vw)}{\partial z_k} = (\partial v/\partial z_k) w + v (\partial w/\partial z_k) \quad \text{by the multiplication rule for differentiation}
\]

\[ = (\partial e^{z_k}/\partial z_k) w + v (\partial(1/u)/\partial z_k) \]

\[ = e^{z_k} w - (v/u^2) \frac{\partial u}{\partial z_k} \quad \text{by the chain rule}
\]

\[ = e^{z_k} u - (v/u^2) e^{z_k} \quad \text{as noted above}
\]

\[ = e^{z_k}/u - (e^{z_k}/u)^2 \quad \text{since } v = e^{z_k}
\]

\[ = y_k - y_k^2 \quad \text{since } y_k = e^{z_k}/u
\]


This page is for answers and rough work