COMP 790-125, HW1

Calvin Deutschbein

October 8, 2015

Problem 1(0.01pt) Open `hw1.tex`, replace "Wile E. Coyote" with your name. Run `pdflatex hw1.tex`, look at hw1.pdf, and confirm that your name is in the right place.

Problem 2(0.5pt)

1. Plot the sigmoid function in MATLAB using script

   ```
   z = [-5:0.1:5];
   fz = 1./(1 + exp(-z));
   plot(z,fz,'LineWidth',3);
   xlabel('z');ylabel('f(z)'); % we always label axes, yes we do!
   hwplotprep
   print -dpdf sigmoid.pdf
   ```

   Find the resulting figure in file `sigmoid.pdf`.

2. In hw1.tex, find the segment of the file that sets up the first figure – it starts with `\begin{figure}` and ends with `\end{figure}`. Inside this segment replace `emptiness.pdf` with `sigmoid.pdf`.

3. Change the text under `\caption` – right now it says "This is emptiness, it earns no points." – to say what the figure is about.

4. Remake hw1.pdf by running in shell/command prompt

   `pdflatex hw1.tex`

   and check that your plot and caption are now in.

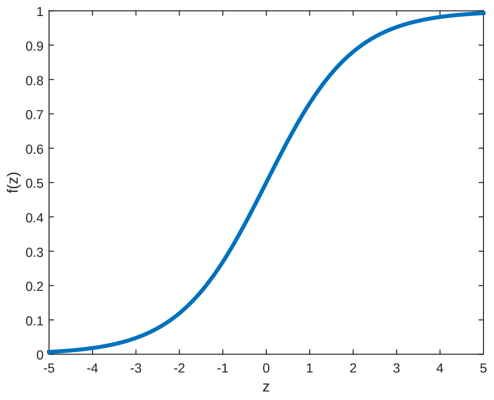

Figure 1: This is a sigmoid function.

Problem 3(0.5pt) Fill in the first derivative and second derivative of sigmoid function in the hw1.tex.

The first derivative and second derivative

$$\frac{df(z)}{dz} = \frac{e^x}{(e^x+1)^2}, \frac{d^2f(z)}{dz} = \frac{-1}{(e^x+1)} + \frac{3}{(e^x+1)^2} - \frac{2}{(e^x+1)^3}.$$

Problem 4(0.5pt) Write a MATLAB function that implements computation of the first derivative of f at a particular point. You just did the math for this. Here is a function that is *wrong*

```
function d = dsigmoid(z)
% This function computes first derivative of sigmoid function at z
d = exp(x)/(exp(x) + 1)^2
end
```

Created a file `dsigmoid.m` that *correctly* computes the first derivative.

Problem 5(0.5pt) We will use your function `dsigmoid.m` to plot the first derivative.

```
zs = [-5:0.01:5];
for i = 1:length(zs)
    ds(i) = dsigmoid(zs(i));
end
plot(zs,ds,'LineWidth',3);
xlabel('z');ylabel('df(z)');
hwplotprep
print -dpdf dsigmoid.pdf
```

Find the resulting plot in file `dsigmoid.pdf`. In hw1.tex replace `emptiness.pdf` with `dsigmoid.pdf` . Change the caption in the figure to say what the figure is about. Remake hw1.pdf and check that your plot has made it in.

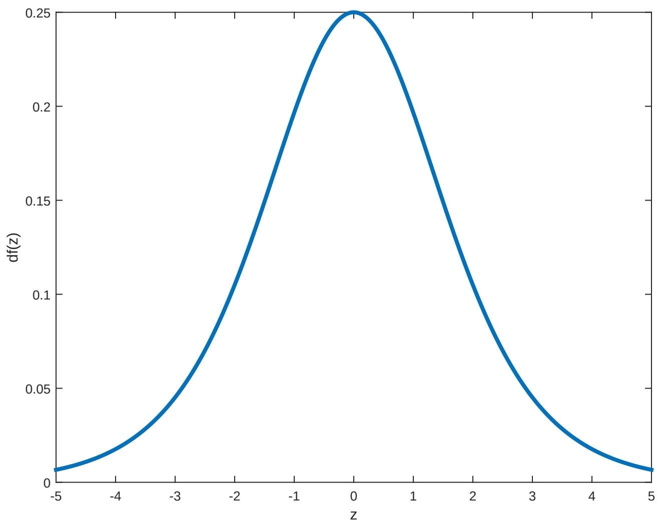

Figure 2: This is dsigmoid.

Problem 6(0.5pt) We can approximate derivatives numerically

$$\frac{df(z)}{dz} \approx \frac{f(z+h) - f(z)}{h}$$

where the right-side of this approximate equality is called *finite difference* approximation. Unlike derivative definition we do not need h to be infinitesimal, just a small value. The numerical approximation of a derivative is tremendously useful trick to check you derivative, gradients, Jacobians, Hessians etc. Make sure that you understand what it does.

We will use this approximation to check your derivatives. Here is a function that computes approximately the derivatives of sigmoid

```
function d = fdsigmoid(z)
f0 = 1/(1 + exp(-z));
f1 = 1/(1 + exp(-(z + 1e-5)));
d = (f1 - f0)/1e-5;
end
```

Save this function into a file names `fdsigmoid.m`.

Try following code in MATLAB

```
zs = randn(100,1);
for i=1:length(zs)
    err(i) = dsigmoid(zs(i)) - fdsigmoid(zs(i));
end
hist(err,30)
hwplotprep
print -dpdf hist.pdf
```

The code above samples 100 normally distributed values and computes the finite differences approximation and the derivative you derived and implemented and then plots histogram of errors.

Find the resulting plot in file `hist.pdf`. In hw1.tex replace `emptiness.pdf` with `hist.pdf` . Change the caption in the figure to say what the figure is about. Remake hw1.pdf and check that your plot has made it in.

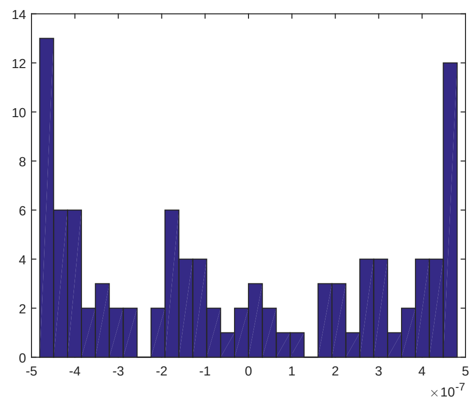

Figure 3: This is the histogram.

Remark 1. *The error ranges between 0 and 13.*

Problem 7(0.5pt) Let

$$f(z) = \frac{1}{1 + \exp\{-z\}} = p \tag{1}$$

express z in terms of p

$$z = log(\frac{-p}{p-1}).$$

Now suppose

$$\frac{\exp\{-z\}}{1+\exp\{-z\}} = q \tag{2}$$

and express z in terms of q

$$z = log(\frac{1-q}{q}).$$

Given Eqs.(1),(2) express q in terms of p

$$q = pe^{\frac{p}{p-1}}.$$

Express $f(-z)$ in terms of $f(z)$

$$f(-z) = \frac{1}{1+e^z}.$$

Hint: the manipulations that are useful here are either subtraction from 1 (as in $1-x$), computing inverse (as in $\frac{1}{x}$), and taking logarithm (as in $\log(x)$).

Log of sigmoid

Problem 8(0.5pt) Let $g(z)$ be log of sigmoid function

$$g(z) = \log\left\{\frac{1}{1+\exp\{-z\}}\right\}.$$

Compute its derivative and fill it in here

$$\frac{dg(z)}{dz} = \frac{1}{1+e^z}.$$

Check your derivative by comparing its value to the finite difference approximation.

Problem 9(0.5pt) Compute second derivative of $g(z)$

$$\frac{d^2g(z)}{d^2z} = -\frac{e^x}{(1+e^x)^2}.$$

Check the second derivative by comparing its value to the finite difference of the *first* derivatives you computed above.

Problem 10(0.5pt) Let the dataset be specified by $\mathcal{D} = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$. We specify conditional probability of y

$$p(y_i|\mathbf{x}_i, \beta_0, \beta) = \frac{1}{1+\exp\{-y_i(\beta_0 + \langle\beta, \mathbf{x}_i\rangle)\}} \tag{3}$$

Write a matlab function that computes log probability of label y given a vector of features $\mathbf{x}$ and β_0, β.

```
function logP = logProbLogReg(y,x,beta0,beta)
logP =log(1/(1+exp(-y*(beta0+dot(beta,x)))))
end
```

Now write a matlab function that uses the above function to compute log probability of label +1 for a vector of features $\mathbf{x}$ and β_0, β

```
function predY = predictY(x,beta0,beta)
logProbY = logP(1,x,beta0,beta);
if logProbY > 0
    predY = 1
else
    predY = -1
end
```

Hint: Since $p(y = 1|x) + p(y = -1|x) = 1$, what is the threshold $p(y = 1|x)$ has to exceed for you to predict that y is 1? Consequently what is the threshold that $\log p(y = 1|x)$ has to exceed for you to predict that y is 1?

Generalize this code so that it works for a matrix `X` with each row being a sample, and returns predicted label for each sample.

```
function predY = predictY(X,beta0,beta)
C = num2cell(X,2)
cellfun(@(x) helppredY(x,beta0,beta), C)
end

function helppredY = helppredictY(x,beta0,beta)
logProbY = logP(1,x,beta0,beta);
if logProbY > 0
    helppredY = 1
else
    helppredY = -1
end
```

Problem 11(0.5pt) Given Eq.(3) we can write out log-likelihood

$$\mathrm{ALL}(\beta_0, \beta; \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{1}{1 + \exp\{-y_i(\beta_0 + \langle \beta, \mathbf{x}_i \rangle)\}}. \tag{4}$$

Now using function `logProbLogReg` that you obtained for the previous problem, write a matlab function that computes loglikelihood

```
function val = AverageLogLikLogReg(y,X,beta0,beta)
val = 0;
for i=1:length(y)
   val = val + (logP(y(i),X(i,:),beta0,beta) / length(y))
end
```

Problem 12(0.5pt) Write a function that computes gradient of log-likelihood of logistic regression Eq.(4)

```
function [dbeta0,dbeta] = dAverageLogLikLogReg(y,X,beta0,beta)
h = 1e-5;
dbeta0 = (ALLLR(y,X,beta0+h,beta) - ALLLR(y,X,beta0,beta))/h;
for i=1:length(beta)
    f0 = ALLLR(y,X,beta0,beta);
    beta(i) = beta(i) + h;
    f1 = ALLLR(y,X,beta0,beta);
    beta(i) = beta(i) - h;
dbeta(i) = (f1-f0)/h
end
end
```

You can make sure that your implementation is correct using the finite differences trick.

Problem 13(0.5pt) We will add ridge penalty term to log-likelihood

$$\mathrm{ALL}(\beta_0, \beta; \mathcal{D}) = \left(\frac{1}{N} \sum_{i=1}^{N} \log \frac{1}{1 + \exp\left\{-y_i(\beta_0 + \langle \beta, \mathbf{x}_i \rangle)\right\}} \right) + \lambda \sum_{j=1}^{p} \beta_j^2. \qquad (5)$$

Note that ridge penalty does *not* apply to β_0.

Change functions `AverageLogLikLogReg` and `dAverageLogLikLogReg` so that they compute penalized average-log-likelihood, and its gradient, respectively.

```
function val = AverageLogLikLogReg(y,X,beta0,beta,lambda)
val = 0;
for i=1:length(y)
   val = val + (logP(y(i),X(i,:),beta0,beta) / length(y))
end
for i=1:length(beta)
    val = val + lambda * beta(i) * beta(i)
end
end

function [dbeta0,dbeta] = dAverageLogLikLogReg(y,X,beta0,beta,lambda)
h = 1e-5;
dbeta0 = (ALLLR(y,X,beta0+h,beta,lambda) - ALLLR(y,X,beta0,beta,lambda))/h;
for i=1:length(beta)
    f0 = ALLLR(y,X,beta0,beta,lambda);
    beta(i) = beta(i) + h;
    f1 = ALLLR(y,X,beta0,beta,lambda);
    beta(i) = beta(i) - h;
dbeta(i) = (f1-f0)/h
```

```
end
end
```

You can use finite differences to check the gradient.

Problem 14(2pt) Implement a gradient ascent algorithm for fitting logistic regression and paste it below. Remember, gradient ascent iterates updates to parameters by taking a step in the direction of the gradient.

```
function [beta0,beta] = fitLogReg(trainY,trainX,lambda,s)
beta0 = trainY(1)
beta = trainX(1,:)
for i=1:20
    [dbeta0,dbeta] = dgradient(trainY,trainX,beta0,beta,lambda);
    beta0 = dbeta0*s+beta0;
    beta = dbeta*s+beta;
end
end
```

Problem 15(1pt) In Matlab load data stored in `hw1.mat`. There are six variables stored in this environment: `trainX,trainy,validX,validy,testX,testy`. First two variables store a training set, second two a validation set, and the last two a test set. You will inspect the dataset.

Visualize sample i's features by running

```
imagesc(reshape(trainX(i,:),[192 168]));colormap(gray)
```

What is the range of values in each sample's feature vector (`trainX(i,:)`)? [0,255]

How many different label values are in the set (look at `trainy`)? 2

What is the nature of the data? It is a multidimensional image.

Problem 16(2pt) You will fit penalized logistic regression using the code you developed earlier. You will run this code for different combinations of step-sizes (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}) and for penalty weight `lambda` (0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1) on the data stored in `hw1.mat`.

Recall that `load hw1.mat` loads the `trainy` and `trainX` variables. You can train on this dataset by running command

```
[beta0,beta] = fitLogReg(trainy,trainX,lambda,s)
```

Write code that runs this fitting procedure for each step-size and λ pair and stores resulting `beta0` and `beta`.

```
stepsizes = 10.^[-1 -2 -3 -4 -5];
lambdas = [0.001 0.01 0.1 0.2 0.4 0.8 1.0];
for i=1:length(stepsizes)
    for j=1:length(lambdas)
```

```
        [beta0s(i*length(lambas)+j),betas(i*length(lambas)+j)] = fitLogReg(trainy,trainX,lam
    end
end
```

This will amount to running the fitting procedure 35 times and getting new beta0 and beta each time.

Using stored beta0 and beta, make predictions on validation set validX and compare the predicted labels to validy. If a predicted label differs from the corresponding value in validy, that sample has been misclassified. Misclassification error is the fraction of misclassified samples in the set. For example, if there are 10 misclassification errors on a set of size 135, then the misclassification error will be $\frac{10}{135} = 0.0741$.

On the validation set, compute misclassification error for each of the 35 stored beta0 and beta.

```
for i=1:length(stepsizes)
    for j=1:length(lambdas)
        val = 0;
        ind = i + length(lambdas)*j
        for k=1:length(validy)
            val = val + abs(validy(k)-predY(validX(k,:),beta0s(ind),betas(ind,:)));
        err(i,j) = val / length(validy);
    end
end
```

Populate the table below with these errors.

Matlab took it's sweet time here so I took the liberty of continuing on to latter sections and doing my best as I have some concern that it will complete in any reasonable amount of time at all. I, unfortunately, am not experienced enough with matlab to be able to determine what aspects of my code are causing these high complexities. I would welcome feedback in this area.

s \ lambda	0.001	0.01	0.1	0.2	0.4	0.8	1
10^{-1}							
10^{-2}							
10^{-3}							
10^{-4}							
10^{-5}							

Problem 17(1pt) Given the above validation errors, select the beta0 and beta that achieve lowest misclassification error on the validation set.

For the selected beta0 and beta, evaluate misclassification error on the test set, testX and testy. Report the beta0.

Unfortunately, whatever trap I fell into in Matlab has also crippled my ability to answer this question with actual values.

```
beta0 = ...
```

Sort the entries in `beta` based on their absolute value and the top 10 entries' indices.

```
indices_top10 = ...
```

Report test set misclassification error.

```
testerr = ...
```

Problem 18(2pt) β vector contains weights of entries in a feature vector. Hence, you can determine which features are more important than others by comparing their weights. You already identified top 10 features based on their weights. Explain which features are useful for correctly predicting the label, and why. Argue strengths and weaknesses of the dominant features.

Features are a single data value known across all samples for which the likelihood of a specific label being applied can be reasonably assumed. The beta value on the features determines the strength or importance of the correlation of a certain feature with a label.

Features are useful when they have higher correlations. However, a member of a set of features may not be useful despite having high meaningful correlations if it lacks independence from other features in a set such that by removing that feature and expressing it in terms of other features little information may be lost. The value of a future is directly proportionate to the absolute value of its correlation with labels in proportion with its independence.