COMP 790-124, HW1

September 15, 2014

Deadline: 9/23/2014 11:59PM EST
Submit `hw1.pdf` by e-mail, `mailto:vjojic+comp790+hw1@cs.unc.edu`.

Instructions

Software In most cases the department and university machines come with software that we need pre-installed. If not you will need to install

1. a LaTeX implementation, on Windows MikTex `http://miktex.org/`, on Mac MacTex `http://www.tug.org/mactex/2011/`
2. MATLAB following instructions from `http://software.unc.edu`
3. RECOMMENDED: git to keep version of your homework files as you work on them `http://git-scm.com/downloads/`

Files for HW1 Download `http://www.cs.unc.edu/~vjojic/comp790/notes/hw1.zip` and unpack it into a directory. In the directory you unpacked the zip file, you will find several .m,.tex,.sty files and hw1.pdf. You will be changing hw1.tex and adding some pdf generated by MATLAB scripts into this directory. You should learn MATLAB and LaTeX, but for this assignment instructions will be pretty straightforward. If you find yourself stuck on how to write out a particular piece of math in LaTeX look at `http://tobi.oetiker.ch/lshort/lshort.pdf`.

Problems and points Maximum number of points that contribute to your grade from this homework is 10. If you earn more than 10 points, this will be recorded, but it will not affect your grade.

Read all the problems CAREFULLY, some will have an obvious `answer` box which means that you will find `\answer` in hw1.tex and replace it with your answer. Others will have you change something in a piece of code or may require you to run a piece of code and include a .pdf in the hw1.tex. Hence, not all the problems will have explicit and obvious `answer` box. As a rule of thumb EVERY problem will have you change the hw1.tex. If you completed a problem, but have not changed hw1.tex, go back and read the problem you

are forgetting something. Once you have changed hw1.tex remake hw1.pdf and look at it. Make sure that your answer has been updated and it looks clean and readable. Keep copies of hw1.tex so you can easily revert to an older version (`git` is fine for this).

Your code may be in Python or R, instead of MATLAB. Just paste it in place of the MATLAB code where appropriate.

Turning the homework in You will send me hw1.pdf file by e-mail to `mailto:vjojic+comp790+hw1@cs.unc.edu`. If you send me multiple versions of the document I will read the latest version that arrived prior to the deadline.

Sigmoids'R'us

Problems in this assignment are meant to ensure that all of the things that you need to complete this and future homework assignments are working. Including your calculus! Importantly, we will get into a habit of checking our math, numerics and intuitions against each other.

We will start with a univariate function that is one of the workhorses in classification.

The function is

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

and it is called sigmoid.

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(1pt)

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.

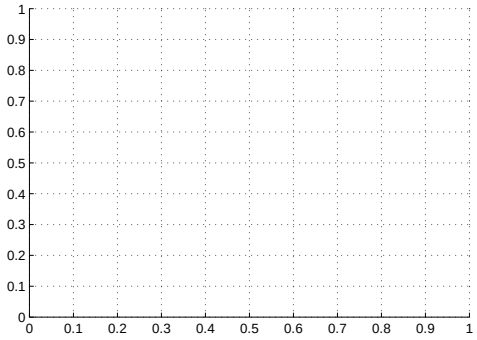

Figure 1: This is emptiness, it earns no points.

Problem 3(1pt) Fill in the first derivative and second derivative of sigmoid function in the hw1.tex.

The first derivative

$$\frac{df(z)}{dz} = \boxed{\texttt{answer}}.$$

You might have to consult an intro to LaTeXin order to figure out how to format your math.

Problem 4(1pt) 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 = ...
end
```

Correct hw1.tex by replacing `...` above with the correct MATLAB code to compute expression you obtained in previous problem.

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

Problem 5(1pt) 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.

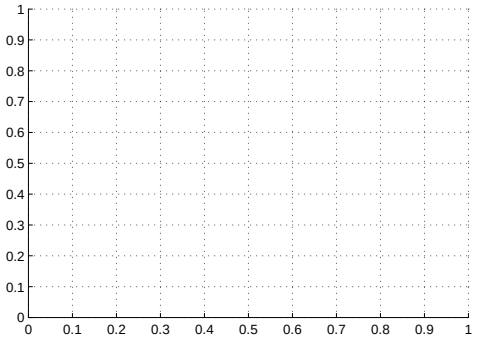

Figure 2: This is the emptiness, it earns no points.

Problem 6(1pt) 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.

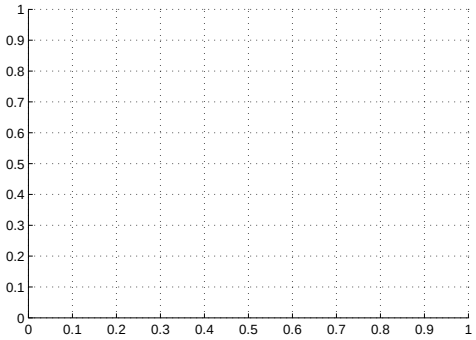

Figure 3: This is the emptiness, it earns no points.

Remark 1. *The error ranges between* `answer` *and* `answer` .

Problem 7(1pt) From Taylor's theorem (first year calculus) we can obtain

$$f(z+h) = f(z) + \frac{df(z)}{dz}h + \frac{1}{2}\frac{d^2f(z)}{d^2z}h^2 + O(h^3).$$

Derive a bound on the error of the finite differences approximation using the above expression. You should use big O notation to express this bound, for

example $O(h^{1.5})$.

$$\mathrm{Err}(z_0, h) = \left| \frac{f(z_0 + h) - f(z_0)}{h} - \frac{df(z_0)}{dz} \right| \leq \boxed{\texttt{answer}}$$

Specifically for sigmoid function plug in appropriate derivative on the right hand side of the inequality. If $h = 10^{-5}$ and $z_0 = 0$ the error of the finite difference should be about $\boxed{\texttt{answer}}$. Does this agree with the histogram of error that is in the figure above?

Problem 8(1pt) Let

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

express z in terms of p

$$z = \boxed{\texttt{answer}}.$$

Now suppose

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

and express z in terms of q

$$z = \boxed{\texttt{answer}}.$$

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

$$q = \boxed{\texttt{answer}}.$$

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

$$f(-z) = \boxed{\texttt{answer}}.$$

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 9(1pt) 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

$$\frac{dg(z)}{dz} = \boxed{\texttt{answer}}.$$

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

Problem 10(1pt) Compute second derivative of $g(z)$

$$\frac{d^2 g(z)}{d^2 z} = \boxed{\texttt{answer}}.$$

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

Problem 11(1pt) 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( .... )
```

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 = logProgLogReg(1,X,beta0,beta);
if logProbY > ...
    predY = ...
else
    predY = ...
end
```

Problem 12(1pt) Given Eq.(3) we can write out log-likelihood

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

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

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

Problem 13(1pt) Write a function that computes gradient of log-likelihood of logistic regression Eq.(4)

```
function [dbeta0,dbeta] = dLogLikLogReg(y,X,beta0,beta)
dbeta0 = ...
for i=1:length(beta)
    dbeta(i) = ...
end
```

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

Problem 14(1pt) Implement a gradient ascent algorithm for fitting logistic regression and paste it below.

```
function [beta0,beta] = fitLogReg(trainY,trainX)
...
```

Run it with fixed step size $s = 10^{-5}$, for 2000 iterations, on data stored in `hw1.mat`. Note that `load hw1.mat` loads the y and X variables, on which you can run by issuing command `[beta0,beta] = fitLogReg(trainY,trainX)`. Report resulting β_0, β

```
beta0 = ...
beta0 = ...
```

Problem 15(1pt) Implement estimation of prediction error using cross validation

```
load hw1.mat
rand('seed',1); K = 5; N = length(y);
indices = crossvalind('Kfold',N,K);
for k=1:K
    testX = x(:,indices == k);
    testY = y(indices == k);

    trainX = x(:,indices ~= k);
    trainY = y(indices ~= k);

    [beta0,beta] = fitLogLikLogReg(trainY,trainX);

    for i=1:length(testY)
        predY = ...
        err(k) = err(k) + ...
    end
end
cvErr = sum(err)/length(y);
```

Once done, run this on data stored in `hw1.mat`. The cross-validation estimate of error on that dataset is `answer` .

For the curious - no points here folks!

Complex-step derivative There is a different way to compute a numerical approximation of a derivative that is more accurate than the finite differences. It is called complex step derivative `http://dx.doi.org/10.1145/838250.838251`.