Homework 1 (Part A; a Part B will be added soon)

Due Date: Wed, January 29th by 5 pm

---

1 Linear Regression

In this problem, you will implement algorithms for linear regression to reproduce curves similar to those you have seen in lecture, exploring how various choices of parameters affect training and test error.

1.1 Task Description

The training and test data are generated by the following model:

$$y = \sin(2\pi x) + \epsilon, \quad \epsilon \sim N(0, 0.04)$$

You are given training and test data, and your job is to

1. Fit the data by applying the psuedo-inverse approach of linear regression using $\{x^i\}, i = 0, 1, \ldots, M$ as features (try $M = 0, 1, \ldots, 9$). Plot training error and test error as Root Mean Square Error (RMSE) against M, the order of polynomial. Your graph should qualitatively resemble the figure in the Lecture 2 slides, page 38.

2. Apply Stochastic Gradient Descend (SGD) to find the coefficients for $M = 3$. To help you get started, we have included pseudo-code for SGD below:

```
w0 := 0; // 0 vector
w := w0;
diff := infinity;
while (diff > tolerance)
    for n=1,2,...,N
        w = w - step_size * put_gradient_here
    end
    diff := norm(w - w0)
end
```

   Here we require that you initialize the weights with the 0 vector (in practice, random initialization or initialize with prior knowledge are common). You will have to tweak the tolerance and step size

parameter in order to get the algorithm to converge properly. As w is updated each time (in the inner loop), record its distance from the analytical solution you get in the first task, and plot this distance against number of updates (the figure is usually referred to as learning curve).

* Note: If you find that the algorithm stops when w is still far away from the analytical solution, you can try the following things: tweak step size factor, tweak tolerance, shuffle the data every time before you enter the inner loop, do resampling instead of a sequential sweep of data, etc.

3. Apply L2 Regularized Linear Regression to find the coefficients for $M = 9$ (use the analytical approach). Plot training and test errors as a function of regularization coefficient in the following range: $\lambda \in \{\exp(-40), \exp(-39), \ldots, \exp(0)\}$ (note: use the logarithm scale for the x-axis). Your graph should qualitatively resemble the figure in Lecture 2 slides, page 51.

1.2 Data Files

Download data.mat and load it into Matlab. 4 variables are included in the file:

- x 10×1 vector, training data input.
- y 10×1 vector, training data output.
- x_{test} 1001×1 vector, test data input.
- y_{test} 1001×1 vector, test data output.

2 Handwritten Digit Classification with Logistic Regression

Download the file

`mnist_49_3000.mat`

from CTools. This is a subset of the MNIST handwritten digit database, which is a well-known benchmark database for classification algorithms. This subset contains examples of the digits 4 and 9.

The data file contains variables x and y, with the former containing patterns and the latter labels. The images are stored as column vectors. To visualize an image, type

```
>> load mnist_49_3000
>> [d,n] = size(x);
>> i = 1; % index of image to be visualized
>> imagesc(reshape(x(:,i),[sqrt(d),sqrt(d)])') % notice the transpose
```

Implement Newton's method (a.k.a. Newton-Raphson) to find a maximizer of the logistic regression conditional log likelihood, initializing with the zero vector. Use the first 2000 examples as training data, and the last 1000 as test data. Please report the following:

1. The test error
2. Your termination criterion (multiple options here)
3. The value of the conditional log-likelihood at the optimum

4. In addition, generate a figure displaying 20 images in a 4×5 array. These images should be the 20 misclassified images for which the logistic regression classifier was most confident about its prediction (you will have to define a notion of confidence in a reasonable way – explain what this is). In the title of each subplot, indicate the true label of the image. What you should expect to see is a bunch of 4s that look like 9s and 9s that look like 4s.

Helpful commands:

`log, exp, .*, ./, sum, min, find, sign, zeros, ones, repmat, figure, subplot, title, num2str`

* Note: you will find that the Hessian is singular or near singular, which means that its inverse does not exist or cannot be reliably computed. To remedy this situation, at every iteration add $-\lambda I$ to the Hessian, where I is the identity matrix and $\lambda = 10$. This makes the Hessian negative denite and well-conditioned.