AMSC660/CMSC660 Maria Cameron

Homework 11. Due Monday, Nov. 20

(20 pts)
The goals of this project are

- to acquire practice in working with real data;
- to explore various optimization methods for solving classification problems and understand how their performance is affected by their settings.

What to submit. Please submit a report with figures and comments. Link your codes to the report pdf. These can be e.g. Dropbox links or GitHub links, etc. All optimizers should be coded from scratch.

Programming language You can use any suitable programming language. Matlab or Python are preferable. I provide an auxiliary package in Matlab, but it is easy to rewrite whatever you need from it in Python. The content of this package is described throughout the rest of this problem description.

- If you are going to use Matlab, you can use `mnist.mat` as input. Note that 4-pixel paddings are removed in `minst.mat`.
- If you are using Python, you can read the binary data files (the link is below) directly in Python. In this case, you can but do not have to remove the 4-pixel padding. Or read `mnist.mat` in Python. Or, even simpler, save the data postprocessed with SVD in Matab and then read them in Python.

1 MNIST dataset

You will experiment with the MNIST dataset of handwritten digits 0, 1, ..., 9 available at http://yann.lecun.com/exdb/mnist/. The training set has 60000 28 x 28 grayscale images of handwritten digits (10 classes) and a testing set has 10000 images.

The data files are in binary format. The code `readMNIST.m` written by Siddharth Hegde and slightly modified by me reads these binary files and strips 4-pixel paddings from the images. The code `saveMNIST2mat.m` saves the resulting data to a mat file `mnist.mat`. The file `mnist.mat` and all codes I am mentioning are packaged to `MNISTaux.zip`.

2 Classification problem

The task is to select all images with digits 1 and all images with digits 7 from the training set, find a dividing surface that separates them, and test this dividing surface on the 1's and 7's from the test set. A sample of 1's and 7's from the training set is shown in Fig. 1.

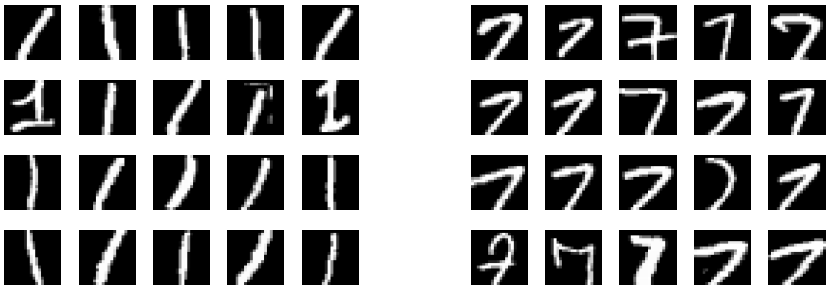

Figure 1: Samples of 20-by-20 images of 1's (left) and 7's (right) from MNIST.

Each image is a point in $\mathbb{R}^{400}$ (the images with stripped paddings are 20-by-20). It is convenient to reduce dimensionality of data by using SVD and mapping the set to $\mathbb{R}^d$ where $d \ll 400$, e.g. $d = 3$, $d = 10$, $d = 20$ – see `mnist_2categories_hyperplane.m`. We label all images with "1" by 1 and all images with "7" by -1. The training data set `Xtrain` (or `X` for brevity) is `Ntrain`-by-`d` matrix. The vector of labels `y` is `Ntrain`-by-`1`.

We pose three kinds of unconstrained optimization problems.

2.1 A smooth loss function for the optimal hyperplane with Tikhonov regularization

In the simplest setting, we aim at finding a dividing hyperplane $w^\top x + b = 0$ with that $w^\top x_j + b > 0$ for all (almost all) x_j corresponding to 1 (labelled with $y_j = 1$) and $w^\top x_j + b < 0$ for all (almost all) x_j corresponding to 7 (labelled with $y_j = -1$). Hence, x_j is classified correctly if

$$\mathsf{sign}(y_j(w^\top x_j + b)) = 1.$$

Instead of the discontinuous sign function, we use a smooth sigmoid-type function (we call it *residual*)

$$r_j \equiv r(x_j; \{w, b\}) := \log\left(1 + e^{-y_j(w^\top x_j + b)}\right) \tag{1}$$

that is close to zero if $y_j(w^\top x_j + b) > 0$ and grows linearly in the negative range of the aggregate $y_j(w^\top x_j + b)$. For brevity, we will denote the d+1-dimensional vector of parameters $\{w, b\}$ by $\mathbf{w}$. We form the loss function by averaging up the residuals and adding a Tikhonov regularization term:

$$f(\mathbf{w}) = \frac{1}{n}\sum_{j=1}^{n} \log\left(1 + e^{-y_j(w^\top x_j + b)}\right) + \frac{\lambda}{2}\|\mathbf{w}\|^2. \tag{2}$$

Here n is the number of data points and λ is a parameter for the Tikhonov regularization. This loss function and its derivatives are encoded in functions `fun0` and `gfun0` at the bottom of the code `mnist_2categories_hyperplane.m`. The optimization problem in `mnist_2categories_hyperplane.m` is solved using the stochastic inexact Newton method. The approximations for Newton's directions are found by the conjugate gradient method. A line search algorithm is used along each proposed direction. If you set `nPCA = 3` in line 5, i.e., $d = 3$, then the dividing hyperplane is visualized.

2.2 A smooth loss function for the optimal quadratic hypersurface with Tikhonov regularization

As you will see, a quadratic dividing hypersurface may lead to much fewer misclassified digits. We are seeking a quadratic hypersurface of the form:

$$x^\top W x + v^\top x + b.$$

Hence, the quadratic test function is

$$q(x_j; \mathbf{w}) := y_j \left(x^\top W x + v^\top x + b \right). \tag{3}$$

The loss function is defined in a similar manner:

$$f(\mathbf{w}) = \frac{1}{n} \sum_{j=1}^{n} \log\left(1 + e^{-q(x_j;\mathbf{w})}\right) + \frac{\lambda}{2} \|\mathbf{w}\|^2. \tag{4}$$

Here $\mathbf{w}$ denotes the $d^2 + d + 1$-dimensional vector of coefficients of $\{W, v, b\}$. This loss function and its gradient are available in the file `qloss.m`.

2.3 A nonlinear least squares problem for the optimal quadratic hypersurface

Finally, we can design the loss function to fit the framework of the nonlinear least squares problem:

$$f(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^{n} [r_j(\mathbf{w})]^2, \quad r_j(\mathbf{w}) = \log\left(1 + e^{-q(x_j;\mathbf{w})}\right). \tag{5}$$

The vector of the residuals and the Jacobian matrix are available in the file `Res_and_Jac.m`.

3 The research tasks

3.1 Levenberg-Marquardt

Set the number of PCAs $d = 20$. Find the optimal quadratic dividing surface. With this number of PCAs and the quadratic surface, you should be able to achieve good accuracy

(the ratio of correctly classified test data to the total number of test data is about 99%). Implement the Levenberg-Marquardt algorithm. A driver for it is
`mnist_2categories_quadratic_NLLS.m`.
It calls (line 94)

```
[w,f,gnorm] = LevenbergMarquardt(r_and_J,w,kmax,tol);
```

You need to code the function `LevenbergMarquardt` yourself. To avoid problems with inverting the matrix $J^\top J$, regularize it by changing it to

$$J^\top J + I \cdot 10^{-6}.$$

3.2 Stochastic optimizers

Implement the following stochastic optimizers.

1. Stochastic gradient descent (experiment with various batch sizes and stepsize decreasing strategies.

2. Stochastic Nesterov (experiment with various batch sizes). Its deterministic version is given by Eqs. (61)–(62) in `Optimization.pdf`.

3. Stochastic Adam (experiment with various batch sizes). Its deterministic version is proposed in a paper by D. P. Kingma and J. L. Ba "Adam: A Method for Stochastic Optimization" where ADAM is introduced: https://arxiv.org/pdf/1412.6980.pdf.

Use these optimizers to minimize the loss function (4) and find the optimal dividing quadratic hypersurface. For each solver, experiment with the appropriate settings. Compare the performance of these optimizers to each other.

Run the stochastic optimizers for the same number of epochs (if you have n data points and your batch size is m then $\mathsf{round}(n/m)$ timesteps is one epoch). Which stochastic optimizer do you find the most efficient?

Include a detailed discussion on the performance of these solvers in various settings in your report. Supplement your report with plots of the estimates for the loss function and the norm of its gradient. Include tables, if appropriate.