CSCI 4961/6961: Homework 5

Assigned Thursday October 15 2020. Due by 11:59pm Thursday October 22 2020.

Create a Jupyter notebook for this assignment, and use Python 3. Write documented, readable and clear code (e.g. use reasonable variable names). Submit this notebook along with a pdf in which the answers to each question are legible, and clearly labeled. You will be graded primarily based on the solutions and answers in the pdf, but the notebook must be runnable. Name the files `RPIid_hw5.ipynb` and `RPIid_hw5.pdf`, where `RPIid` is your six letter RPI id.

In this homework, you will compare, empirically, gradient descent and steepest descent, on a multiclass classification problem using the Chinese MNIST dataset. Specifically you will fit a multiclass logistic regression model to solve the problem of recognizing which of 15 digits a given image contains.

1. Download the file

   `http://www.cs.rpi.edu/~gittea/teaching/fall2020/files/hw5start.ipynb`,

   rename it to `RPIid_hw5.ipynb`, and use it as the starting point of your solution. Download the dataset from

   `http://www.cs.rpi.edu/~gittea/teaching/fall2020/files/chinesemnist.tgz`

   and untar it into your working directory.

   Answer the following problems *in your pdf* in full sentences and provide the plots asked for in the pdf. The TA will not look in your Python code for answers that are not present in the pdf.

   – Our objective function for this ERM problem is:

   $$f(\mathbf{W}, \mathbf{b}) = -\frac{1}{n_{\text{train}}} \sum_{i=1}^{n} \left[\mathbf{y}_i^T (\mathbf{W}\mathbf{x}_i + \mathbf{b}) - \log\left(\mathbf{1}^T \exp(\mathbf{W}\mathbf{x}_i + \mathbf{b})\right)\right]$$

   where $\mathbf{W} \in \mathbb{R}^{k \times d}$ and $\mathbf{b} \in \mathbb{R}^k$. For this problem, $k = 15$ and $d = 4096$.
   Write a function `loss(X, Y, W, b)` that computes $f(\mathbf{W}, \mathbf{b})$ on the training set given in the matrices `X` and `Y`, and functions `gradWloss(X, Y, W, b)` and `gradbloss(X, Y, W, b)` that compute the gradients $\nabla_{\mathbf{W}} f(\mathbf{W}, \mathbf{b})$ and $\nabla_{\mathbf{b}} f(\mathbf{W}, \mathbf{b})$, respectively, on the training set.

   – Implement a gradient descent solver function

   ```
   graddescent(X, Y, gradWloss, gradbloss, W1, b1, eta, a, tol)
   ```

   that takes as input:

   * `X`, an array of size $n_{\text{train}} \times d$, each row of which is a feature vector,
   * `Y`, an array of size $n_{\text{train}} \times k$, each row of which is the one-hot encoding of the target class for that instance,
   * `gradWloss(X, Y, W, b)`, described above,
   * `gradbloss(X, Y, W, b)`, described above,
   * `W1`, the initial guess for the parameter matrix $\mathbf{W}$,
   * `b1`, the initial guess for the parameter vector $\mathbf{b}$,
   * `eta` and `a`, parameters governing the step size as $\alpha_t = \frac{\eta}{1+at}$,

    * `tol`, the stopping tolerance: terminate the algorithm at the first iteration T when [1]

$$\max\{\|\nabla_{\mathbf{b}} f(\mathbf{W}_T, \mathbf{b}_T)\|_\infty, \|\nabla_{\mathbf{W}} f(\mathbf{W}_T, \mathbf{b}_T)\|_\infty\} \leq \texttt{tol}.$$

  This function should return the sequence $(\mathbf{W}_1, \mathbf{b}_1), \ldots, (\mathbf{W}_T, \mathbf{b}_T)$ of model parameters.

- Implement a steepest descent solver function that uses backtracking linesearch to choose the stepsize,

  `steepestdescent(X, Y, loss, gradWloss, gradbloss, W1, b1, c, beta, tol).`

  Most of these arguments are described above; the new parameters `c` and `beta` are the parameters for backtracking line search. This function should return the sequence $(\mathbf{W}_1, \mathbf{b}_1), \ldots, (\mathbf{W}_T, \mathbf{b}_T)$ of model parameters and another sequence $(e_1, \ldots, e_T)$, where e_t counts the number of times `loss` and `gradWloss` were called to compute the models up to and including the model at iteration t.

- As in the previous homework, use a subset of the training data to select parameters `eta` and `a` that allow you to achieve a tolerance of 10^{-3} (meaning that when you use this as `tol`, your gradient descent solver terminates eventually). Run gradient descent on the full dataset with this same tolerance.

- Select values for c and β in accordance with the guidance given in the notes for Lecture 13, and run the steepest descent method until a convergence tolerance of 10^{-3} is achieved.

- Plot, on the same graph, for both the test and training datasets, the accuracy (percentage of correctly predicted labels) of

    * The sequence of models returned by the gradient descent method.
    * The sequence of models returned by the steepest descent method.

  Let the x-axis be the number of iterations used to compute each model.

- Plot, on the same graph, the accuracy of the two sequences of models again, but this time let the x-axis be the number of function and gradient evaluations used to compute each model (note that gradient descent uses one gradient evaluation at each iteration).

- Compare and contrast the behaviors of gradient descent and the method of steepest descent on this problem.

2. [For CSCI6961 students.]

   Prove that if the objective is a convex quadratic of the form $f(\mathbf{x}) = c + \mathbf{a}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{M}\mathbf{x}$, where $\mathbf{M} \succeq 0$, then Newton's method with a stepsize of 1 reaches $\mathbf{x}_\star$ in one iteration, from any arbitrary starting point.

[1]Recall that for a vector $\mathbf{v}$, the max-norm $\|\mathbf{v}\|_\infty = \max_i |v_i|$. Similarly, by the max-norm of a matrix, we mean $\|\mathbf{W}\|_\infty = \max_{i,j} |w_{ij}|$. So we stop when the gradients with respect to each model parameter is very close to zero, indicating that we're close to optimality.