CSCI 4961/6961: Homework 4

Assigned Monday October 5 2020. Due by 11:59pm Monday October 12 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_hw2.ipynb` and `RPIid_hw2.pdf`, where `RPIid` is your six letter RPI id.

1. You will compare, empirically, subgradient descent to stochastic subgradient descent, usingthe ℓ_2-regularized support vector machine optimization problem, which has objective function

$$f(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\big(1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)\big)_+ + \frac{\lambda}{2}\|\mathbf{w}\|_2^2.$$

Here $\mathbf{x}_i \in \mathbb{R}^d$ are feature vectors and $y_i \in \{-1, 1\}$.

Specifically, you will fit a SVM on the UCI Adult data set, where the task is to predict whether a person's income is above $50K$/yr based on census data, using both subgradient descent and stochastic subgradient descent, and compare the accuracy vs computational effort for the two methods. The computational effort will be measured by the number of data points touched in generating a particular set of model parameters.

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.

- What is the subdifferential of f, considered as a function of the vector $(\mathbf{w}, b)$? Give the complete subdifferential, not just a subgradient.
- Implement a stochastic subgradient descent solver function `sgdmethod(X, y, subgradloss, subgradreg, regparam, w1, T, a, m)` that takes as input:
  - `X`, an array of size $n \times d$, each row of which is a feature vector,
  - `y`, an array with n rows, each row of which is the corresponding target,
  - `subgradloss(x, y, w)`, a function that computes a subgradient of the loss on a single training example at the current parameter vector,
  - `subgradreg(regparam, w)`, a function that computes a subgradient of the regularizer at the current parameter vector,
  - `regparam`, the regularization parameter λ,
  - `w1`, the initial guess for the parameter vector $(\mathbf{w}, b)$,
  - `T`, the number of iterations,
  - `a`, a parameter governing the step size as $\alpha_t = (1 + at)^{-1}$,
  - `m`, the minibatch size,

  and returns the sequence $\boldsymbol{\omega}_1 = (\mathbf{w}_1, b_1), \ldots, \boldsymbol{\omega}_T = (\mathbf{w}_T, b_T)$ of model parameters. Sample m training pairs without replacement to form each minibatch. Note that when $m = n_{\text{train}}$, `sgdmethod` actually implements the subgradient descent method.
- Load the `a9a` training and testing data sets from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html` using `sklearn.datasets.load_svmlight_file`. Remove the last column from the training data set, so both data sets have 122 features (the 123rd feature doesn't exist in the test data). Preprocess the features by scaling, as in previous homeworks, in order to increase the accuracy of the SVM model.

- Take $\lambda = 1/n_{\text{train}}$ and, by looking at the convergence curves for different values of a on a subset of the training data, choose a parameter a that you think will perform well for the subgradient method on the entire training data set.
- Run the subgradient descent method using this value of a to 'convergence', as determined visually by the asymptoting out of $f(\boldsymbol{\omega}_t)$ to a flat line.
- Similarly choose an a appropriate for sgd with minibatch size of one and run the method to convergence.
- Plot, on the same graph, the accuracy (percentage of correctly predicted labels) of:
  * The sequence of models returned by the subgradient descent method, on the training set.
  * The sequence of models obtained by averaging the last half of the models returned by the subgradient descent method up to iterate t, on the training set. That is, for each $t \geq 2$, plot the accuracy of the model with parameter $\frac{1}{t-\lfloor t/2 \rfloor+1} \sum_{i=\lfloor t/2 \rfloor}^{t} \boldsymbol{\omega}_i$. This is a more robust alternative to the model averaging we saw in class.
  * The sequence of models returned by the subgradient descent method, on the test set.
  * The sequence of models obtained by averaging the last half of the models returned by the subgradient method up to iterate t, on the test set.

  To facilitate comparison between the rates of convergence, let the x-axis be the number of datapoints used (counting multiple passes) to generate each model — e.g. the first model returned from the subgradient method touched n_{train} datapoints, the second touched $2n_{\text{train}}$ and so on. Label your data series and axes! You may find it useful to omit the first few points to more clearly compare the behaviors.
- Make a similar plot, but this time plot the performance of the sgd method on the training and test sets.
- Report the highest test accuracy for each of four methods (gradient and subgradient descent with the last model returned, and with the averaged models), along with the number of datapoints used to obtain those accuracies.
- What observations do you have about the performance of sgd and the subgradient method, and the impact of averaging?[1]

2. [For CSCI6961 students.] Consider the following iterative algorithm for minimizing a function f on a convex set C: given a current point $\mathbf{x}_t$, linearize the function f around $\mathbf{x_t}$, then minimize this linear approximation over C, using a quadratic regularization that forces us not to stray too far from $\mathbf{x}_t$. That is, given $\mathbf{g} \in \partial f(\mathbf{x}_t)$

$$\mathbf{x}_{t+1} = \operatorname{argmin}_{\mathbf{u} \in C} f(\mathbf{x}_t) + \langle \mathbf{g}, \mathbf{u} - \mathbf{x}_t \rangle + \frac{1}{2\alpha_t} \|\mathbf{u} - \mathbf{x}_t\|_2^2.$$

Argue that this algorithm is exactly the projected subgradient method with stepsize given by α_t.

[1]You may want to play around with ℓ_1 regularization and minibatch sizes to see how they affect performance as well.