Take home final

SDS385

Fall 2019

This exam has two theoretical and three coding questions. You need to answer all questions.
Read each question carefully, **show your work** and clearly present your answers. Note, the exam is printed two-sided - please don't forget the problems on the even pages!
Please make sure these are your own answers. You can talk to the instructor or the TA if you have any questions.
Turn in your solution via canvas by Dec 16th 10am.

Good Luck!

Name: _____

UTeid: _____

1 Experimental questions (30 points)

1. (10 pts) **LSH for Project Gutenberg Data**: Download **Gutenberg.zip** which contains 3036 e-books from project Gutenberg (a small subset of their full corpus) in separate **.txt** files. Convert each book into its set of 2-shingles using scikit-learns **HashVectorizer** with 2^{18} features. The output will be a **scipy.sparse** matrix X. For Jaccard similarity, use a binary (present/absent) representation instead of shingle counts or frequencies. For each document, generate the MinHash signatures for $n = 90$ hash functions of the same form as in your homework, using a prime p larger than 2^{18}. Use the LSH procedure with $n = 90$ signatures and $b = 30$ bands. Rather than matching signatures explicitly, hash the band of signatures with a separate hash table for each band (see Mining of Massive Datasets section 3.4.1). Let any two documents with the same signature key for at least one band be called *neighbors*.

   (a) Approximate the pairwise Jaccard similarity as the fraction of total matching MinHash signatures between all *neighboring* documents (notice how many comparisons this saves vs. $O(3036^2)$). Then find the k most similar document pairs according to this approximation. For $k = 10, 20, 30$, show your results in a table. Specifically, show the document pairs, their true Jaccard score. Hint: You may find it helpful while developing your code to serialize your partial computations (the shingle matrix, signature matrix, etc.) using **cPickle** or **numpy.save**.

   (b) How long will exact search take for $k = 10, 20$ and 30? Compare the search results from LSH and exact search. Also compare the time taken by both for $k = 10, 20$ and 30.

2. **Stochastic Variance Reduced Gradient Descent (SVRG).** (10 pts) We are going to implement SVRG which we saw in one of the class presentations.

   Consider minimizing a function of the following form:

$$F(\beta) = \frac{1}{n} \sum_{i=1}^{n} f(x_i, \beta).$$

   A lot of objective functions defined over i.i.d data points, like least squares or maximum likelihood has this form. f is typically convex, and may or may not be differentiable.

   There are several possible choices of a stochastic subgradient. Here are two different ones, along with the names they are most known by:

- SGD $\tilde{g}(\beta) = \nabla f_i(\beta)$, $i \in [n]$ random
- SVRG $\tilde{g}(\beta) = \nabla f_i(\beta) + \nabla f_i(\beta^{\text{old}}) - \nabla F(\beta^{\text{old}})$, $i \in [n]$ random

You have already seen SGD (aka stochastic gradient descent). We have read about SVRG (stochastic variance reduced gradient descent). β^{old} is some old point which we now define. The SVRG algorithm is as follows:

(a) Initialize at some $\boldsymbol{y_1}$, and set $\boldsymbol{x}_1 = \boldsymbol{y}_1$.

(b) Outer loop: For $s = 1, 2, ...$, compute $\mu_s = \nabla F(\boldsymbol{y_s})$, and set $\boldsymbol{x_0} = \boldsymbol{y_s}$.

(c) Inner loop: For $k = 1, ..., m$, run stochastic gradient descent with a re-centered stochastic gradient:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta(\nabla f_I(\boldsymbol{x}_k) - \nabla f_I(\boldsymbol{y}_s) + \mu_s).$$

(d) Option I: set

$$\boldsymbol{y}_{s+1} = \boldsymbol{x}_{m+1}.$$

(e) Option II: set

$$\boldsymbol{y}_{s+1} = \frac{1}{m}\sum_{k=1}^{m} \boldsymbol{x}_{k+1}.$$

(f) Option III: Choose a random index $J \in [m]$, and set

$$\boldsymbol{y}_{s+1} = \boldsymbol{x}_{J+1}.$$

In practice, Option I seems the most natural, though some of the standard proofs rely on other versions. This algorithm explains what β^{old} means. The rationale is that, the re-centered stochastic gradient has a significantly reduced variance.

Recall the multiclass logistic regression problem from HW1. You will again work on the digits data and the news data. For each of these, compare (plot) the performance of GD, SGD, and SVRG for logistic regression **with ℓ^2 regularization**. Plot your results against the number of gradient evaluations for a single training example (GD performs n such evaluations per iteration and SGD performs 1).

Set the regularizer to $1/n$, where n is the number of data points. How does the choice of m affect the performance of SVRG? What seems to be a good choice of m? Now choose a bigger regularizer. What do you observe?

Do option I, II and III perform differently? Show plots to compare their performance on the two datasets.

3. **Spectral Clustering on large sparse graphs.** In class we have learned about Spectral Clustering for datasets. In this question you will do Spectral Clustering for sparse networks. In network.txt find a sparse representation of a network in (src, dest, weight) format. Note that the weight is always 1. Build an undirected and unweighted adjacency matrix from this. An adjacency matrix of a graph A is an n by n matrix (n is the number of nodes). $A_{ij} = 1$ if there is an edge between nodes i and j. The true labels of the nodes can be found in comm.txt.

   (a) Do Spectral Clustering on this adjacency matrix using your own eigenvector routine. Report the accuracy.

   (b) Open the paper on network clustering by Amini et al from **here**. Read section 2.3.2 to better understand why Spectral Clustering sometimes does not perform well on very sparse networks. Now implement the regularized spectral clustering algorithm proposed in this paper using your own eigenvector routine. You should code up your eigenvector computation such that you do not need to create a $n \times n$ dense matrix. Report the time and accuracy.

2 Theory questions (30 points)

1. Consider the sub-gradient descent algorithm you learned in class (see lecture4). In this question we will work on obtaining rates of convergence for this algorithm. Assume that you are minimizing a convex function f with minimizer x^*. Also assume that f is Lipschitz continuous, i.e. all sub-gradientsare bounded above:

$$\|g\|_2 \leq G.$$

Also assume that the distance of the initial point from the truth x^* is bounded as below:

$$\|x^{(1)} - x^*\|_2 \leq R.$$

Define $d_k := \|x^{(k)} - x^*\|_2$.

(a) Prove that

$$d_{k+1}^2 \leq d_k^2 - 2\alpha_k(f(x^{(k)}) - f(x^*)) + \alpha_k^2\|g^{(k)}\|^2.$$

(b) Using the above show that

$$2\sum_{j=1}^{k} \alpha_j(f(x^{(j)}) - f(x^*)) \leq R^2 + \sum_{j=1}^{k} \alpha_j^2\|g^{(j)}\|^2$$

(c) Now show that,

$$f_{\text{best}}^{(k)} - f(x^*) \leq \frac{R^2 + G^2\sum_{j=1}^{k}\alpha_j^2}{2\sum_{j=1}^{k}\alpha_j},$$

where $f_{\text{best}}^{(k)} = \min(f(x^{(1)}), \ldots, f(x^{(k)}))$.

(d) Find a sequence of α_i's such that the above is $O(\log k/\sqrt{k})$. Give a proof.

2. Consider a design matrix $\boldsymbol{X}$ such that $\boldsymbol{X}$ has orthonormal columns, i.e. $\boldsymbol{X^T X} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the $p \times p$ identity matrix. Consider the following regularization:

$$\min_{\beta} \frac{1}{2}(\boldsymbol{X\beta - y})^T(\boldsymbol{X\beta - y}) + \lambda||\boldsymbol{\beta}||_m^q \tag{1}$$

(a) Derive the solution to equation (1) for $m = 2, q = 2$ (ridge regression).

(b) Derive the solution to equation (1) for $m = 1, q = 1$ (lasso).

(c) When $m = 0, q = 1$, the penalty involves $||\boldsymbol{\beta}||_0 = \sum_{i=1}^{p} \mathbb{1}(\beta_i \neq 0)$.

i. Is this a convex optimization problem? Why or why not?

ii. Show that the solution to equation (1) with $m = 0$ is given by $\tilde{\boldsymbol{\beta}}$, where

$$\tilde{\beta}_i = \begin{cases} \boldsymbol{v_i^T y}, & \text{if } |\boldsymbol{v_i^T y}| > \sqrt{2\lambda} \\ 0 & \text{if } |\boldsymbol{v_i^T y}| \leq \sqrt{2\lambda} \end{cases}$$

This is also called the hard thresholding estimator. $\boldsymbol{v_i}$ is the i^{th} column of $\boldsymbol{X}$.