Homework 2

Convex Optimization 10-725/36-725

Due Friday October 2 at 4:00pm
submitted to Mallory Deptola in GHC 8001
(Remember to submit each problem on a separate sheet of paper, with your name on at the top)

1 Subgradient method convergence [Shashank]

In class, we proved the convergence rate for simple gradient descent for a differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ with Lipschitz gradient is $O(1/\epsilon)$. Said differently, the rate is $O(1/k)$, and we proved that

$$f(x^{(k)}) - f^\star \leq \frac{\|x^{(0)} - x^\star\|_2^2}{2tk}$$

where $0 < t < 1/L$, where t is a fixed step size, L is the Lipschitz constant of the gradient of f, $x^{(0)}$ is the starting point, $x^\star \in \arg\min_{x \in \mathbb{R}^n} f(x)$, $f^\star = f(x^\star)$, and k is the number of iterations.

In this question, you will be proving the convergence rate for the sub-gradient method for a convex function. Assume that:

- f is convex with $dom(f) = \mathbb{R}^n$
- f is Lipschitz continuous with continuity parameter G
- At the kth step, the update is given by $x^{(k)} = x^{(k-1)} - t_k \cdot g^{(k-1)}$, where $g^{(k-1)}$ is an element from the subdifferential of f at $x^{(k-1)}$

(a) Using properties of the subgradient show that

$$\|x^{(k)} - x^\star\|_2^2 - \|x^{(k-1)} - x^\star\|_2^2 \leq -2t_k\big(f(x^{(k-1)}) - f(x^\star)\big) + t_k^2\|g^{(k-1)}\|_2^2$$

and then use a telescoping sum to show

$$\|x^{(k)} - x^\star\|_2^2 \leq \|x^{(0)} - x^\star\|_2^2 - 2\sum_{i=1}^{k} t_i\big(f(x^{(i-1)}) - f(x^\star)\big) + \sum_{i=1}^{k} t_i^2\|g^{(i-1)}\|_2^2$$

(b) We know that the subgradient method is not a strict descent procedure. All through the procedure, we need to keep track of the best iterate $x_{\text{best}}^{(k)}$ such that $f(x_{\text{best}}^{(k)}) = \min_{i=0,\ldots k} f(x^{(i)})$. Now, use the result in the previous part to prove that

$$f(x_{\text{best}}^{(k)}) - f(x^\star) \leq \frac{R^2 + G^2\sum_{i=1}^{k} t_i^2}{2\sum_{i=1}^{k} t_i}$$

where $R = \|x^{(0)} - x^\star\|_2$

(c) From the basic inequality you just derived, verify the following results stated in class:

- For a fixed step size t, the subgradient method satisfies $\lim_{k\to\infty} f(x_{\text{best}}^{(k)}) \leq f^\star + G^2 t/2$
- For diminishing step sizes, the subgradient method satisfies $\lim_{k\to\infty} f(x_{\text{best}}^{(k)}) = f^\star$

(d) Using the result in part (b), prove that for an appropriate fixed step size t, the subgradient method takes $O(1/\epsilon^2)$ iterations to get within ϵ-distance of the optimal objective value.

2 Practice with subgradients and prox operators [Dallas]

1. As we learned in class, subdifferentials collect all subgradients at each point in the domain of a function, i.e. $\partial f(x) = \{g \in \mathbb{R}^n : g \text{ is a subgradient of } f \text{ at } x\}$. In particular, it was given in class that if $f(x) = \max_{i=1,..,m} g_i(x)$, where $g_1, .., g_m$ are convex, then

$$\partial f(x) = \text{conv}\big(\bigcup \{\partial g_i(x) : g_i(x) = f(x)\}\big)$$

i.e. the subdifferential is equal to the convex hull of the union of the subdifferentials of all the given functions that are maximal for the given value of x.

   (a) Prove that $\text{conv}\big(\bigcup \{\partial g_i(x) : g_i(x) = f(x)\}\big) \subseteq \partial f(x)$
   (b) The L_1 penalty can be written as $f(x) = \|x\|_1 = \max\{s^T x : s_i \in \{-1, +1\}\}$. Verify that applying the above formula produces the correct subdifferential for $f(x)$.
   (c) Show that $\partial\lambda_{\max}(Q) \supseteq \text{conv}\Big(\{uu^T : u^T Q u = \lambda_{\max}(Q),\ \|u\|_2 = 1\}\Big)$, for $Q \in \mathbb{S}^n$

   Note: the above is actually an equality, but you are only required to show one direction.

2. The proximal operator (proximal map) is defined as

$$\text{prox}_{h,t}(x) = \underset{z}{\text{argmin}}\ \frac{1}{2}\|z - x\|_2^2 + t h(z)$$

which we can think of as minimizing $h(z)$, while respecting a quadratic penalty on the distance from $z = x$.

Derive the proximal map for each of the following functions:

   (a) L_2 norm: $h(z) = \|z\|_2$
   (b) Elastic net penalty: $h(z) = \alpha\|z\|_1 + (1-\alpha)\|z\|_2^2$, $0 < \alpha < 1$
   (c) $h(z) = \|z\|_q^q = \big(\sum_{i=1}^p |z_i|^q\big)$, $0 < q < 1$

3. The *proximal minimization* algorithm is:

$$x^{(k+1)} = \text{prox}_{h,t}(x^{(k)})$$

   (a) Write out the proximal minimization algorithm applied to $h(x) = \frac{1}{2}x^T A x - b^T x$, where $A \in \mathbb{S}^n$
   (b) Show that this is equivalent to the *iterative refinement* algorithm:

$$x^{(k+1)} = x^{(k)} + (A + \epsilon I)^{-1}(b - Ax^{(k)})$$

   where $\epsilon > 0$ is a constant
   (c) Assuming that proximal minimization converges to the minimizer of $h(x)$, what would the iterations of iterative refinement converge to? Why would this be useful in the case of a rank deficient A?

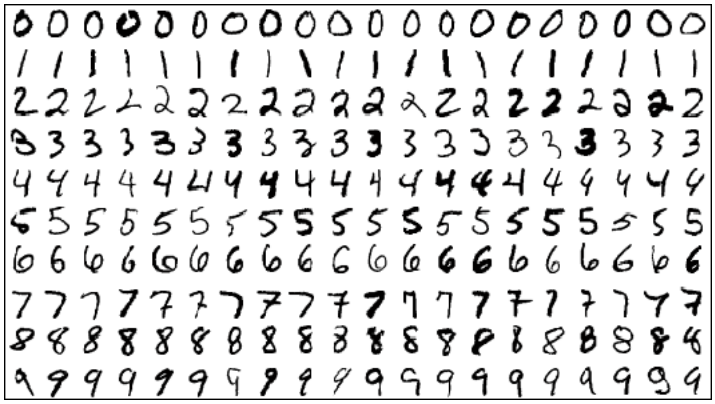

Figure 1: Digit examples from the MNIST dataset

3 Proximal gradient for sparse logistic regression [Matt]

In this problem we employ proximal gradient methods to build a classifier for recognizing handwritten digits from images. Instead of using the pixel values of the digit images directly, we have generated random Fourier features for the classification task which allow us to learn a nonlinear decision function with a linear classifier. We formulate the problem as multiclass classification with output label $y \in \{0, \ldots, 9\}$ and input features $x \in \mathbb{R}^n$. We model $y|x$ with the probabilistic model

$$p(y = j - 1|x) = \frac{\exp(x^T \beta_j)}{\sum_{k=1}^{10} \exp(x^T \beta_k)}$$

corresponding to a multinomial distribution and feature weights $\beta_j \in \mathbb{R}^n$ for digit $j - 1$. Note that we could also model the multinomial distribution over 10 classes with only 9 sets of parameters, but using separate parameters for each class simplifies implementation.

We will learn the weights $\beta_1, \ldots, \beta_{10}$ from training data by minimizing the negative log-likelihood over m training examples

$$\begin{aligned} g(\beta_1, \ldots, \beta_{10}) &= \sum_{i=1}^{m} -\log p(y_i|x_i; \beta_1, \ldots, \beta_{10}) \\ &= \sum_{i=1}^{m} \left(\log \sum_{k=1}^{10} \exp\left(x_i^T \beta_k\right) - x_i^T \beta_{y_i} \right) \end{aligned}$$

with the addition of ℓ_1-regularization. The final optimization problem is

$$\min_{\beta_1,\ldots,\beta_{10}} f(\beta_1, \ldots, \beta_{10}) = \min_{\beta_1,\ldots,\beta_{10}} g(\beta_1, \ldots, \beta_{10}) + \lambda \sum_{k=1}^{10} \|\beta_k\|_1.$$

The data for this problem is from the classic machine learning dataset MNIST. This dataset has 60K training examples but in order to reduce computation time, we will use only 1K in this problem. The training and test datasets with the features we will use are in `mnist.mat` and the original images are in `mnist_original.mat`, both available on the class website. (Note to R users: to read Matlab files into R, you can use the package `R.matlab` on CRAN.)

(a) When implementing numerical methods, it is often convenient (and more efficient) to arrange

parameters in vectors and matrices rather than dealing with sums explicitly. Let

$$B = \begin{bmatrix} | & & | \\ \beta_1 & \cdots & \beta_{10} \\ | & & | \end{bmatrix}, \quad X = \begin{bmatrix} — & x_1^T & — \\ & \vdots & \\ — & x_m^T & — \end{bmatrix}$$

and let $Y \in \mathbb{R}^{m \times 10}$ be the "one-hot" encoding of the training labels: $y_{ik} = 1$ if example i has label $k-1$ and $y_{ik} = 0$ otherwise[1]. Show that the gradient of the smooth component of the objective is

$$\nabla_B g(B) = X^T(Z \exp(XB) - Y)$$

where $\exp(\cdot)$ is applied elementwise and $Z \in \mathbb{R}^{m \times m}$ is the diagonal matrix with

$$Z_{ii} = \frac{1}{\sum_{k=1}^{10} \exp(x_i^T \beta_k)}.$$

(b) Implement proximal gradient method with and without acceleration. Fit the model parameters on the training data (`X` and `y`) using regularization parameter $\lambda = 1$. Run both algorithms for 5000 iterations with fixed step size $t = 10^{-4}$. Plot the convergence of both methods in terms of $f - f^\star$ (provided as `f_star` in `mnist.mat`) scaled logarithmically on the y-axis and number of iterations on the x-axis.

(c) Next, we will evaluate how error rate decreases as a function of the regularization parameter λ. For this part, we will run the accelerated proximal gradient method from part (b) for 1000 iterations. Solve the problem with a sequence logarithmically-spaced values of λ from 10^1 to 10^{-2} and plot the error rate measured on the test data (`Xtest` and `ytest`) as a function of λ. What value of λ achieves the lowest error rate on the test data? How many nonzeros does the solution $B^\star$ have for this value of λ?

(d) Finally, using the best value of λ found in part (c) build a model and use it to make predictions on the test set. Using these predictions, order the examples in the test set by the maximum probability for a single class, i.e. by $\max_{k=1,\ldots,10} p(y = k-1|x)$. Identify the 5 correctly classified examples and the 5 incorrectly classified classified examples for which the maximum single class probability is largest. Print these 10 images (the pixel data is in `mnist_original.mat`, use `imshow()` or similar) and include them in your report.

4 Stochastic proximal gradient [Hanzhang]

In this problem we extend the previous proximal gradient method with stochastic schemes.

When the loss function we minimize can be written in the form of $\sum_{i=1}^m g_i(\beta)$, where each $g_i(\beta)$ is the loss at sample i, and the training time is long, then stochastic schemes should be considered. In practice, this typically happens when data is too large to be loaded in memory, or is stored in a distributed fashion. For this problem, you are going to implment stochastic proximal gradient descent method for MNIST in the last problem.

In the last problem, the loss function has the form of $\sum_{i=1}^m g_i(\beta) + h(\beta)$, where g_i is the negative log-likelihood for sample i, and h is the convex ℓ_1-regularization. Then stochastic proximal gradient method does the following in every epoch (cycle of the data).

- Randomly shuffle the data, and partition into a total of n_{batch} batches.

[1] In Matlab, `Y = sparse(1:m, y+1, 1)`.

- For $k = 1, 2, 3, \ldots, n_{batch}$, do

$$\beta \leftarrow \underset{t_k, h}{\text{prox}} \left(\beta - t_k \frac{m}{|\text{Batch}_k|} \sum_{i \in \text{Batch}_k} \nabla g_i(\beta) \right)$$

The key of the stochastic methods is to update the solution β for every sample. In practice, however, it is more computationally efficient to cycle through the samples in mini-batches than through them individually due to issues like disk paging. In addition, one can reuse their implementation of the batch version from the last problem. The stochastic gradient need to be scaled by a constant, however, so that in expectation, it's the same as the true gradient. In this problem, this constant is $\frac{m}{|\text{Batch}_k|}$ for each batch.

For this problem, implement the mini-batch version of stochastic proximal gradient descent with batch size of 100 (number of samples per batch), epoch number of 5000, $t_k = 10^{-5}$, and $\lambda = 1$. Compute the training set objective after every epoch and plot the graph of number of epochs vesus $f - f^*$, where y-axis is in logarithmic scale. Compare this plot with those of the two proximal gradient descents from the last problem part (b).

Bonus question

The accelerated proximal gradient descent algorithm (with constant step size), as presented in Beck and Teboulle (2008), is:

$$x_k = \text{prox}_{t,h}(v_k - t\nabla g(v_k)) \tag{1}$$

$$a_1 = 1 \tag{2}$$

$$a_{k+1} = \frac{1 + \sqrt{1 + 4a_k^2}}{2} \tag{3}$$

$$v_{k+1} = x_k + \left(\frac{a_k - 1}{a_{k+1}} \right) (x_k - x_{k-1}) \tag{4}$$

Show that this algorithm is approximately equivalent to the form present in the lecture slides.