Modern Computational Statistics lec03 Notes
Part II

This is based on PKU course: Modern Computational Statistics. Thanks to Prof. Cheng Zhang, this is a very interesting course.

This lecture will introduce some advanced gradient descent methods(which is some kind of hard). In Part I, I will introduce the upper bound convergence rate of gradient descent and help you understand the Nestrov's accelaration method.

1 Reformulation of NAG and Proof

Last time we have talked about the Nesterov's accelerated gradient descent method, and we found that Nesterov's accelerated gradient descent achieved the oracle convergence rate of first-order methods. Now let's give the reformulation of NAG and prove the previous lemma.

1.1 Reformulation

Initialize $x_0 = u_0$, and for $k = 1, 2, ...$

$$\begin{aligned} y &= (1 - \theta_k)x_{k-1} + \theta_k u_{k-1} \\ x_k &= y - t_k \nabla f(y) \\ u_k &= x_{k-1} + \frac{1}{\theta_k}(x_k - x_{k-1}) \end{aligned} \tag{1}$$

with $\theta_k = 2/(k+1)$.

1.2 Proof

The proof is as below(forgive my poor handwriting):

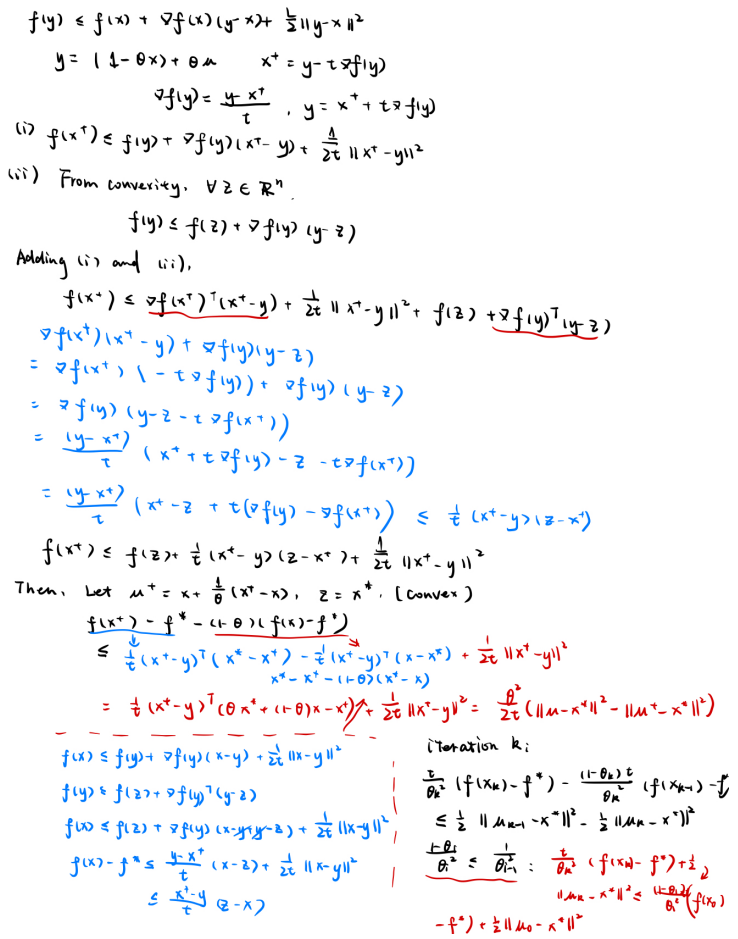

Figure 1: Proof the previous theorem

(Continue)

$$= \frac{1}{2}||x^{(0)} - x^*||^2$$

$$So \quad f(x_k) - f^* \leq \frac{\theta_k^2}{2t}||x_0 - x||^2 = \frac{2}{t(k+1)^2}||x^{(0)} - x^*||^2$$

2 Proximal Gradient Descent

2.1 Motivation

The objective in many unconstrained optimization problems can be split in two components:

$$minmize \quad f(x) = g(x) + h(x)$$

where g is convex and differentiable, h is convex and simple, may be not differentiable.

2.2 Proximal Mapping

The proximal mapping function h is defined as:

$$prox_h(x) = \arg\min_u (h(u) + \frac{1}{2}||u - x||_2^2)$$

2.3 Proximal Gradient Descent

The proximal gradient algorithm is:

$$x^{(k+1)} = prox_{t_k h}(x^{(k)} - t_k \nabla g(x^{(k)})), k = 0, 1, ...$$

This means that, let $x^+ = prox_{th}(x - t\nabla g(x))$, then:

$$\begin{aligned} x^+ &= \arg\min_u (h(u) + \frac{1}{2t}||u - x + t\nabla g(x)||_2^2) \\ &= \arg\min_u (h(u) + g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t}||u - x||_2^2) \end{aligned} \tag{2}$$

Here, x^+ minimizes $h(u)$ plus a simple quadratic local approximation of $g(u)$ around x.

Next, let's see some examples.

2.4 Examples

- Gradient Descent:

  Get the derivative, and let it equal to 0,(suppose $h(x) = 0$), then we have:

  $$x^+ = x - t\nabla g(x)$$

- Projected Gradient Descent:

  This is similar to gradient descent, with a projected function:

  $$x^+ = P_C(x - t\nabla g(x))$$

- ISTA(Iterative Shrinkage- Thresholding Algorithm):

  In ISTA, $h(x) = ||x||_1$

  $$x^+ = S_t(x - t\nabla g(x))$$

  where $S_t(u) = (|u| - t)_+ sign(u)$

2.5 Accelarated Proximal Gradient Descent

We can apply Nesterov's accelaration for proximal gradient descent. Choose any initial $x^{(0)} = x^{(-1)}, \forall k = 1, 2, ...$

$$\begin{aligned} y &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \\ x^{(k)} &= prox_{t_k h}(y - t_k \nabla g(y)) \end{aligned} \tag{3}$$

3 Stochastic Optimization

Stochastic Optimization can be considered as the following problem:

$$\min_x f(x) = E_w(F(x, w)) = \int F(x, w)p(w)dw$$

where w is a random variable.

3.1 Stochastic Gradient Descent

Gradient descent method can be applied with stochastic optimization as:

$$x^{(k+1)} = x^{(k)} - t_k g(x^{(k)})$$

where $E(g(x)) = \nabla f(x), \forall x$. This means that, stochastic gradient descent separates the dataset into several parts, and perform gradient descent in the separated parts:

Consider supervised learning with observations:$D = x_i, y_i{}_{i=1}^N$

$$\min_w f(w) = \frac{1}{N} \sum_{i=1}^{N} l(h_2(x^(i), y^{(i)}))$$

SGD:

$$w^{k+1} = w^{(k)} - t_k \nabla \ell$$

3.2 Convergence rate of SGD

Assume that $E(||g(x)||^2) \leq M^2$, and $f(x)$ is convex,

$$Ef(x^{[0:k]}) - f^* \leq \frac{||x^{(0)} - x^*||_2^2 + M^2 \sum_{j=0}^k t_j^2}{2 \sum_{j=0}^k t_k}$$

where $x^{[0:k]} = \sum_{j=1}^k t_j x^{(j)} / \sum_{j=1}^k t_j$.

To represent it in a convergence rate way, we fix the number of iterations K, and constant step sizes $j_j = \frac{||x^{(0)} - x^*||}{M\sqrt{K}}, j = 0, 1, ..., K$, we have

$$E(f(x^K)) - f^* \leq \frac{||x^{(0)} - x^*||M}{\sqrt{K}}$$

where $x^K = \frac{1}{K+1}\sum_{j=0}^{K} x^{(j)}$.

The convergence rate for SGD is rather slow, but SGD can be applied to big data problems and require less memory usage.

4 Other Methods

4.1 AdaGrad

Adagrad is an improved version of SGD, which adapts the learning rate to the parameters, thus, learning rate can be iterated with the parameters.

Suppose the vector of parameters: θ, gradient at iteration t: g_t. Let η be the usual learning rate for SGD. Adagrad is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

where G_t is a diagonal matrix where each diagonal element is the sum of the squares of the corresponding gradients up to time step t.

4.2 RMSprop

One problem with AdaGrad is that, its accumulation of the squred gradients always cause the learning rate to shrink and eventually become very small. To improve it, Geoff Hinton resolve AdaGrad's diminishing learning rate via exponentially decaying average:

$$E(g^2)_t = 0.9E(g^2)_{t-1} + 0.1g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E(g^2)_t + \epsilon}} g_t$$

4.3 AdamGrad

The most popular gradient method is proposed by D.P.Kingma(2014), which is AdamGrad. In addition to the squared gradients, Adam also keeps an exponentially decaying average of the past gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$$

Adam uses the same update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon} m_t}$$