Notes are taken from Tibshirani's lecture notes: https://www.stat.cmu.edu/~ryantibs/convexopt-F13/scribes/

8.1.2 Projected Subgradient Method

Suppose we wanted to solve the following convex minimization problem

$$\min_x f(x) \text{ subject to } x \in C$$

This problem can be solved by pushing C into the objective function using the indicator function, and then using subgradient descent.

$$\min_x f(x) + I_C(x)$$

If we try to apply the subgradient method to this objective function, we have to take the subgradient of the indicator function. The subgradient of $\partial I_C(x) = N_C(x)$ the normal cone of C at x, which is generally pretty difficult to compute.

Instead, we can use projected subgradient descent. It has the same update as the subgradient method, except we project onto C instead of calculating the subgradient at each step

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

If this projection can be done, the projected subgradient method achieves the same convergence guarantees as subgradient descent.

8.2 Proximal Gradient Descent

Instead of trying to improve rate of convergence for all types of problems, we instead focus on minimizing composite functions of the form

$$f(x) = g(x) + h(x)$$

where g is convex and differentiable, and h is simple and convex but nonsmooth. This relaxes the class of functions we can minimize compared to gradient descent, but for many problems we can still achieve the $O(\frac{1}{\epsilon})$ rate of convergence from gradient descent.

Since h is not differentiable, we cannot directly take the gradient of f and apply the gradient descent update:

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

Instead, we can try another method motivated by the same principles as gradient descent. Recall that the gradient descent method is motivated by minimizing a quadratic approximation to f around x, replaying the Hessian with $\frac{1}{t}I$. Instead of trying to minimize the quadratic around all of f, which we can't do because

h is not differentiable, we can minimize just the quadratic approximation to g and leave h alone. Make the update:

$$\begin{aligned} x^+ &= \text{argmin}_z \bar{g}_t(z) + h(z) \\ &= \text{argmin}_z g(x) + \nabla g(x)^T(z-x) + \frac{1}{2t}||z-x||_2^2 + h(z) \\ &= \text{argmin}_z \frac{1}{2t}||z - (x - t\nabla g(x))||_2^2 + h(z) \end{aligned}$$

The first term $||z - (x - t\nabla g(x))||_2^2$ forces us to stay close to the gradient update for g and the second terms forces us to make $h(z)$ small. This is the principle behind proximal mapping.

8.2.1 Proximal Mapping

Proximal mapping is given by

$$\text{prox}_{h,t}(x) = \arg\min_z ||x-z||_2^2 + h(z)$$

In *proximal gradient descent* we choose an initial $x^{(0)}$ and iteratively update

$$x^{(k)} = \text{prox}_{h,t_k}(x^{(k-1)} - t_k \nabla g(x^{(k-1)}))$$

We can rewrite in the same form as a gradient step by defining

$$G_t(x) = \frac{x - \text{prox}_{h,t}(x - t\nabla g(x))}{t}$$

then rewriting the update as:

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

The advantage of the proximal mapping is that $\text{prox}_{h,t}(x)$ has a closed-form solution for many important functions h, which may not be differentiable but are simple. Making the proximal mapping only depends on g, and because g is smooth we can compute its gradients even though it may be very complicated. The computational cost of making the mapping, however, depends on the function and can be either expensive or cheap.

8.2.2 Example: ISTA

Given $y \in \mathbb{R}^n$, $X \in \mathbb{R}^{n \times p}$, the lasso criterion is given by

$$\begin{aligned} f(\beta) &= \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1 \\ &= g(\beta) + h(\beta) \end{aligned}$$

Then the proximal mapping for $h(\beta) = \lambda||\beta||_1$ is

$$\begin{aligned} \text{prox}_{h,t}(\beta) &= \arg\min_z \frac{1}{2t}||x - z||_2^2 + \lambda||z||_1 \\ &= S_{\lambda t}(\beta) \end{aligned}$$

where $S_t(\beta)$ is the soft thresholding operator.

$$[S_t(\beta)]_i = \begin{cases} \beta_i - \lambda & \beta_i > \lambda \\ 0 & -\lambda \leq \beta_i \leq \lambda \\ \beta_i + \lambda & \beta_i < -\lambda \end{cases}$$

The proximal update is then given by

$$\beta^+ = \text{prox}_{h,t}(\beta - t\nabla g(\beta))$$

Using the fact that $\nabla g(\beta) = -X^T(y - X\beta)$ and the definition of $S_{\lambda t}$,

$$= S_{\lambda t}(\beta + tX^T(y - X\beta))$$

8.2.3 Backtracking Line Search

Backtracking works the same as it does for gradient descent, but because we require the derivative we use backtracking only on g, not f. Choose a parameter $0 < \beta < 1$. At each iteration start at $t = t_{init}$ and while

$$g(x - tG_t(x)) > g(x) - t\nabla g(x)^T G_t(x) + \frac{t}{2}||G_t(x)||_2^2$$

shrink the step size $t = \beta t$. Otherwise perform the proximal gradient update.

Theorem 8.2 *Proximal gradient descent with step size $t \leq \frac{1}{L}$ satisfies*

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

and the same result holds for backtracking, with t replaced by $\frac{\beta}{L}$

Accelerated proximal gradient method

Our problem, as before:

$$\min \; g(x) + h(x)$$

where g convex, differentiable, and h convex. Accelerated proximal gradient method: choose an initial point $x^{(0)} = x^{(-1)} \in \mathbb{R}^n$, repeat for $k = 1, 2, 3, \ldots$

$$\begin{aligned} v &= x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)}) \\ x^{(k)} &= \text{prox}_{t_k}\big(v - t_k \nabla g(v)\big) \end{aligned}$$

- First step $k = 1$ is just usual proximal gradient update
- After that, $v = x^{(k-1)} + \frac{k-2}{k+1}(x^{(k-1)} - x^{(k-2)})$ carries some "momentum" from previous iterations
- $h = 0$ gives accelerated gradient method

Convergence analysis

As usual, we are minimizing $f(x) = g(x) + h(x)$, assuming:

- g is convex, differentiable, $\text{dom}(f) = \mathbb{R}^n$, and ∇g is Lipschitz continuous with constant $L > 0$
- h is convex, prox function can be evaluated

Theorem: Accelerated proximal gradient method with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)}) - f^\star \leq \frac{2\|x^{(0)} - x^\star\|_2^2}{t(k+1)^2}$$

Achieves the optimal rate $O(1/k^2)$ for first-order methods! I.e., a rate of $O(1/\sqrt{\epsilon})$

Backtracking line search

A few ways to do this with acceleration ... here's a simple method (more complicated strategies exist): fix $\beta < 1$, $t_0 = 1$. At iteration k, start with $t = t_{k-1}$, and while

$$g(x^+) > g(v) + \nabla g(v)^T(x^+ - v) + \frac{1}{2t}\|x^+ - v\|_2^2$$

shrink $t = \beta t$, and let $x^+ = \text{prox}_t(v - t\nabla g(v))$. Else keep x^+

Under same assumptions, we get the same rate

Theorem: Accelerated proximal gradient method with backtracking line search satisfies

$$f(x^{(k)}) - f^\star \leq \frac{2\|x^{(0)} - x^\star\|_2^2}{t_{\text{min}}(k+1)^2}$$

where $t_{\text{min}} = \min\{1, \beta/L\}$

Is acceleration always useful?

Acceleration can be a very effective speedup tool ... but should it always be used?

In practice the speedup of using acceleration is diminished in the presence of warm starts. I.e., suppose want to solve lasso problem for tuning parameters values

$$\lambda_1 > \lambda_2 > \ldots > \lambda_r$$

- When solving for λ_1, initialize $x^{(0)} = 0$, record solution $\hat{x}(\lambda_1)$
- When solving for λ_j, initialize $x^{(0)} = \hat{x}(\lambda_{j-1})$, the recorded solution for λ_{j-1}

Over a fine enough grid of λ values, proximal gradient descent can often perform just as well without acceleration

23.1 Stochastic gradient descent

Consider minimizing the average of a bunch of functions:

$$\min_x \frac{1}{n} \sum_{i=1}^n f_i(x)$$

Recall that gradient descent goes as follows:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^n \nabla f_i(x^{(k-1)})$$

The stochastic gradient descent replaces the averaged gradient with the gradient of a randomly chosen function:

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f_{i_k}(x^{(k-1)})$$

where i_k is randomly drawn from $\{1, ..., n\}$.

Note that:

$$\mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x^{(k-1)})$$

So the gradient used is an unbiased, albeit higher variance estimate of the gradient used for GD.

To reduce the variance, mini-batch stochastic gradient descent can be used, in which we sample $I_k \in [n]$ with $|I_k| = b$ functions and use the average of their gradients:

$$\mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{b} \sum_{i \in I_k}^n \nabla f_i(x^{(k-1)})$$

Note that this is again an unbiased estimate of the gradient, b times more expensive than SGD but having $1/b$th the variance.

Condition	GD Rate	SGD Rate
Convex	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$
+ Lipschitz Gradient	$O(1/k)$	$O(1/\sqrt{k})$
+ Strongly Convex	$O(\gamma^k)$	$O(1/k)$

Theorem 1. *If f is convex and G-bounded then after T steps of stochastic gradient descent with $\eta = \frac{R(\sigma+G)}{\sqrt{T}}$ we have, $E[\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) - f(x^*)] \leq \frac{R(\sigma+G)}{\sqrt{T}}$ where $\sigma^2 = \max_x[||\hat{\nabla} f(x) - \nabla f(x)||^2]$ and $R = \|x_0 - x^*\|$.*

Proof. By convexity, $f(x^*) \geq f(x_t) + \nabla f(x_t)(x^* - x_t)$. This implies

$$f(x_t) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) = \hat{\nabla} f(x_t)(x_t - x^*) + (\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*)$$

$$= \frac{1}{\eta}(x_t - x_{t+1})(x_t - x^*) + (\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*)$$

Using the fact that $(a-b)\cdot(a-c) = \frac{1}{2}[||a-b||^2 + ||a-c||^2 - ||b-c||^2]$ we get that

$$f(x_t) - f(x^*) \leq \frac{1}{2\eta}[||x_t - x^*||^2 + ||x_t - x_{t+1}||^2 - ||x_{t+1} - x^*||^2] + (\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*)$$

$$= \frac{1}{2\eta}[||x_t - x^*||^2 - ||x_{t+1} - x^*||^2] + \frac{1}{2\eta}||x_t - x_{t+1}||^2 + (\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*)$$

Since $x_{t+1} = x_t - \eta \hat{\nabla} f(x_t)$ we get that

$$f(x_t) - f(x^*) \leq \frac{1}{2\eta}[||x_t - x^*||^2 - ||x_{t+1} - x^*||^2] + \frac{\eta}{2}||\hat{\nabla} f(x_t)||^2 + (\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*)$$

Now take expectation on both sides. The last term will have an expected value of 0[1].

$$E[f(x_t) - f(x^*)] \leq \frac{1}{2\eta}[||x_t - x^*||^2 - ||x_{t+1} - x^*||^2] + \frac{\eta}{2}E[||\hat{\nabla} f(x_t)||^2] + 0$$

Using $E[y^2] = E[y]^2 + \sigma^2$,

$$E[f(x_t) - f(x)] \leq \frac{1}{2\eta}[||x_t - x^*||^2 - ||x_{t+1} - x^*||^2] + \frac{\eta}{2}(\sigma^2 + G^2)$$

Summing up from $t = 0$ to $T - 1$ we get

$$E[\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) - f(x^*)] \leq \frac{1}{2\eta}\frac{||x_0 - x^*||^2}{T} + \frac{\eta}{2}(\sigma^2 + G^2)$$

$$\leq \frac{R^2}{2\eta T} + \frac{\eta}{2}(\sigma^2 + G^2)$$

Setting $\eta = \frac{R}{\sqrt{T(\sigma^2+G^2)}}$, we get that the RHS is at most

$$\leq \frac{R(\sigma + G)}{\sqrt{T}}$$

□

Theorem 2. *If f is convex and β-Lipschitz then after T steps of stochastic gradient descent with $\eta = \frac{1}{\beta + \sigma\frac{\sqrt{T}}{R}}$ we have, $E[\frac{1}{T}\sum_{t=0}^{T-1} f(x_t) - f(x^*)] \leq \frac{\beta R^2}{T} + \frac{\sigma R}{\sqrt{T}}$, where $\sigma^2 = \max_x[||\hat{\nabla} f(x) - \nabla f(x)||^2]$ and $R = \|x_0 - x^*\|$.*

Proof. We will use the property of Lipschitz functions that $f(y) \leq f(x) + \nabla f(x)(y-x) + \frac{\beta}{2}||x-y||^2$. We have

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)(x_{t+1} - x_t) + \frac{\beta}{2}||x_{t+1} - x_t||^2.$$

Also by convexity, $f(x^*) \geq f(x_t) + \nabla f(x_t)(x^* - x_t)$. From this we get that

$$f(x_{t+1}) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) + \nabla f(x_t)(x_{t+1} - x_t) + \frac{\beta}{2}||x_{t+1} - x_t||^2$$

Writing this in terms of the noisy gradient:

$$f(x_{t+1}) - f(x^*) \leq \nabla f(x_t)(x_t - x^*) - \eta \nabla f(x_t)\hat{\nabla} f(x_t) + \frac{\beta\eta^2}{2}||\hat{\nabla} f(x_t)||^2$$

$$\leq \hat{\nabla} f(x_t)(x_t - x^*) - \eta||\hat{\nabla} f(x_t)||^2 + \frac{\beta\eta^2}{2}||\hat{\nabla} f(x_t)||$$

$$+(\nabla f(x_t) - \hat{\nabla} f(x_t))(x_t - x^*) - \eta(\nabla f(x_t) - \hat{\nabla} f(x_t))\hat{\nabla} f(x_t)$$

Note that we had the first three terms above when there was no noise. The last two terms above are new in the noisy case. When you take the expectation, the new noisy case terms are 0 and $\eta\sigma^2$, respectively. Lets rewrite the above equation with this in mind. We get

$$f(x_{t+1}) - f(x^*) \leq \hat{\nabla} f(x_t)(x_t - x^*) - \eta||\hat{\nabla} f(x_t)||^2 + \frac{\beta\eta^2}{2}||\hat{\nabla} f(x_t)||$$

$$+\eta\sigma^2 + \frac{1}{2\eta}||x_t - x^*||^2 - \frac{1}{2\eta}||x_t - x^*||^2$$

$$\leq -\frac{1}{2\eta}[||x_t - x^* - \eta\hat{\nabla} f(x_t)||^2 - \frac{\eta}{2}||\hat{\nabla} f(x_t)||^2 + \frac{\beta\eta}{2}||\hat{\nabla} f(x_t)||^2 + \eta\sigma^2 + \frac{1}{2\eta}[||x_t - x^*||^2]$$

$$\frac{1}{2\eta}[||x_t - x^*||^2 - ||x_{t+1} - x^*||^2] + \eta\sigma^2$$

Since $\frac{\eta}{2} > \frac{\beta\eta^2}{2}$, then $\eta < \frac{1}{\beta}$ and we get that

$$E[\frac{1}{T}\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*)] \leq \frac{R^2}{2\eta T} + \eta\sigma^2$$

Set $\eta = \frac{1}{\beta + \frac{\sigma\sqrt{T}}{R}}$

$$E[\frac{1}{T}\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*)] \leq \frac{\beta R^2}{T} + \frac{\sigma R}{\sqrt{T}}$$

□

Variance Reduction

23.2.1 Stochastic average gradient

- Maintain a table, containing gradient g_i of f_i, for $i = 1, ..., n$
- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i(x^{(0)})$, for $i = 1, ..., n$
- At steps $k = 1, 2, 3...$ pick random $i_k \in \{1, ..., n\}$ and set:

$$g_{i_k}^{(k)} = \nabla f_{i_k}(x^{(k-1)})$$

  while setting $g_i^{(k)} = g_i^{(k-1)}$ for $i \neq i_k$.

- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot \frac{1}{n} \sum_{i=1}^{n} g_i^{(k)}$$

SAG gradient estimates are no longer unbiased, but they have greatly reduced variance.

For the update step, at each iteration, it only requires an $O(1)$ addition for the aggregate gradient from the previous step to be modified to become the aggregate gradient for the current step:

$$x^{(k)} = x^{(k-1)} - t_k \cdot (\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n} + \frac{1}{n} \sum_{i=1}^{n} g_i^{(k-1)})$$

The above update can be seen as a moving average over a sliding window which is a constant time calculation.

Suppose for each f_i, ∇f_i is Lipschitz with constant L. Write $\bar{x}^{(k)} = \frac{1}{k} \sum_{l=0}^{k-1} x^{(l)}$, then we have the below theorem for convergence rate:

Theorem 23.1 ***cite****: SAG, with a fixed step size* $t = \frac{1}{16L}$*, and initialization* $g_i^{(0)} = \nabla f_i(x^{(0)}) - \nabla f(x^{(0)})$ *for* $i = 1, ..., n$ *is such that:*

$$\mathbb{E}[f(\bar{x}^{(k)})] - f^\star \leq \frac{48n}{k}(f(x^{(0)}) - f^\star) + \frac{128L}{k}||x^{(0)} - x^\star||_2^2$$

This gives $O(1/k)$ convergence rate which matches that for GD. The only difference is that the constant factor for GD is $O(L)$ where it is $O(n + L)$ for SAG. Notice that the initialization $g_i^{(0)}$ centers the gradients around zero by removing the average from every element of the gradient; in other words, the average of $g_i^{(0)}$ is 0. Moreover, the first term in the above convergence bound depends on n; the authors of the work have suggested mitigating this by warm starting with the result of n SGD steps.

Furthermore, under the even stronger assumption of strong convexity where each f_i is strongly convex with parameter m:

Theorem 23.2 ***cite****: SAG, with a fixed step size* $t = \frac{1}{16L}$*, and initialization* $g_i^{(0)} = \nabla f_i(x^{(0)}) - \nabla f(x^{(0)})$ *for* $i = 1, ..., n$ *is such that:*

$$\mathbb{E}[\bar{x}^{(k)}] - f^\star \leq (1 - \min\{\frac{m}{16L}, \frac{1}{8n}\})^k \cdot (\frac{3}{2} f(x^{(0)}) - f^\star) + \frac{4L}{n}||x^{(0)} - x^\star||_2^2)$$

This gives $O(\gamma^k)$ convergence rate and again matches that for GD. The convergence analysis proofs for SAG is extremely complicated because of the biased estimator in SAG update rule.

23.2.2 SAGA

- Maintain a table, containing gradient g_i of f_i, for $i = 1, ..., n$
- Initialize $x^{(0)}$ and $g_i^{(0)} = \nabla f_i(x^{(0)})$, for $i = 1, ..., n$
- At steps $k = 1, 2, 3...$ pick random $i_k \in \{1, ..., n\}$ and set:

$$g_{i_k}^{(k)} = \nabla f_{i_k}(x^{(k-1)})$$

while setting $g_i^{(k)} = g_i^{(k-1)}$ for $i \neq i_k$.

- Update

$$x^{(k)} = x^{(k-1)} - t_k \cdot (g_{i_k}^{(k)} - g_{i_k}^{(k-1)} + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)})$$

Notice that the only difference between SAG and SAGA is the heavier weight on the updated gradient at step k. We have:

$$[g_{i_k}^{(k)} - g_{i_k}^{(k-1)}] + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

instead of

$$[\frac{g_{i_k}^{(k)}}{n} - \frac{g_{i_k}^{(k-1)}}{n}] + \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

Interestingly, the SAGA gradient is unbiased. This is because

$$\mathbb{E}[g_{i_k}^{(k-1)}] = \frac{1}{n}\sum_{i=1}^{n} g_i^{(k-1)}$$

and

$$\mathbb{E}[g_{i_k}^{(k)}] = \mathbb{E}[\nabla f_{i_k}(x^{(k-1)})] = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x^{(k-1)})$$

AdaGrad (Adam)

One popular adaptive step size method is called *AdaGrad*. Let $g^{(k)} = \nabla f_{i_k}(x^{k-1}))$, for $j = 1, ..., p$ do:

$$x_j^{(k)} = x_j^{(k-1)} - \alpha \frac{g_j^{(k)}}{\sqrt{\sum_{l=1}^{k} (g_j^{(l)})^2}}$$

The advantage of the above update rule is that, we do not need to tune our learning rate anymore as α is now a fixed hyperparameter. It is noted that in sparse problems, AdaGrad performs much better than standard SGD. Several extensions of AdaGrad exists, *viz.*, Adam, RMSProp etc.