Machine Learning

Instructor: Prof. Ganesh Ramakrishnan

Barrier methods

- Consider the objective

$$\min f(x)$$
$$\text{s.t. } g_i(x) \leq 0, \forall i$$

- Indicator function for $g_i(x)$

$$I_{g_i}(x) = \begin{cases} 0, & \text{if } g_i(x) \leq 0 \\ \infty, & \text{otherwise} \end{cases}$$

  - We have shown that this is convex
- We will use subgradient descent to solve this optimization

Option 1: Sum of indicators

- Convert our objective to the following unconstrained optimization problem
- Let $C_i = \{x \mid g_i(x) \leq 0\}$
- We take

$$\min_x F(x) = \min_x f(x) + \sum_i I_{C_i}(x)$$

- Consider the subgradient of F:

$$g_F(x) = g_f(x) + \sum_i g_{I_{C_i}}(x)$$

- Recall that $g_{I_{C_i}}(x)$ is $d \in \mathbf{R}^n$ s.t. $d^\top x \geq d^\top y$, $\forall y \in C_i$
- $g_{I_{C_i}}(x) = 0$ if x is in the interior of C_i, and has other solutions if x is on the boundary

Option 1: More General

- Consider the following sum of a differentiable function $f(x)$ and a nondifferentiable function $c(x)$
- We take

$$\min_x F(x) = \min_x f(x) + c(x)$$

- Like gradient descent, consider the first order approximation for $f(x)$ around x^k leaving $c(x)$ alone:

$$\min_x f(x^k) + \nabla^T f(x^k)(x - x^k) + \frac{1}{2t}||x - x^k||^2 + c(x)$$

- Adding $f(x^k)^2$ to the objective (without any loss) to complete squares

$$x^{k+1} = \operatorname{argmin}_x \frac{1}{2t}||x - (x^k - t\nabla f(x^k))||^2 + c(x)$$

- In general, such a step is called a *proximal* step

$$x^{k+1} = prox_t\left(||x^k - t\nabla f(x^k))||^2 + c(x)\right)$$

Option 1: Generalized Gradient Descent

- Interesting because in many settings, $prox_t(x)$ can be computed efficiently

$$prox_t(z) = \operatorname*{argmin}_x \frac{1}{2t}||x - z||^2 + c(x)$$

- Illustration on Lasso[1]
-
-
-
-

[1] How did we come up with the iterative algo for Lasso on page 8 of http://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture23a.pdf?

Illustration on Lasso[2]

[2] Justification of the iterative algo for Lasso on page 8 of
http://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture23a.pdf

Illustration on Lasso[3]

[3] Justification of the iterative algo for Lasso on page 8 of
http://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture23a.pdf

Option 1: Generalized Gradient Descent

- Recall
$$prox_t(z) = \operatorname*{argmin}_x \frac{1}{2t}||x - z||^2 + c(x)$$
- Gradient Descent: $c(x) = 0$
- Projected Gradient Descent: $c(x) = \sum_i g_{I_{C_i}}(x)$
- Proximal Minimization: $f(x) = 0$
- Convergence: If $f(x)$ is convex, differentiable, and ∇f is Lipschitz continuous with constant $L > 0$ AND $c(x)$ is convex and $prox_t(x)$ can be solved exactly then convergence result (and proof) is similar to that for gradient descent

$$f(x^k) - f(x^*) \leq \frac{1}{k}\sum_{i=1}^{k}\left(f(x^i) - f(x^*)\right) \leq \frac{\left\|x^{(0)} - x^*\right\|^2}{2tk}$$

Eg: Projected Gradient Descent

- Let

$$dist(x, C_i) = \min_{u \in C_i} \|x - u\|^2$$

- We define

$$D(x) = \max_i dist(x, C_i)$$

  - If C_i is closed and convex, a unique minimizer $P_{C_i}(x)$ exists (projection of x on C_i)
  - $dist(x, C_i) = 0$ if $x \in C_i$

- Recall discussion on subgradient descent for this problem in class notes[4]

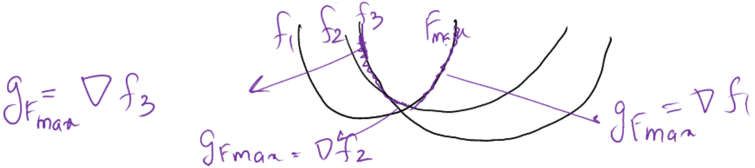
[4] http://www.cse.iitb.ac.in/~cs709/notes/enotes/lecture22a.pdf

- We get the subgradient of $D(x)$ as

$$g_D(x) = \nabla dist(x, C_i) \text{ if } D(x) = dist(x, C_i)$$

- For illustration, consider

$$g_{F_{max}}(x) = \nabla f_i(x) \text{ if } f_i(x) = \max_j f_j(x)$$



  - If f_i gives maximum value at a point, $g_{F_{max}}$ will be ∇f_i at that point
  - At the points of intersection of f_i and f_j, we will get some convex combination of ∇f_i and ∇f_j

Projection methods

- So far, we have dealt with simple projections during SMO and the general decomposition method
    - We considered $\alpha_i y_i + \alpha_j y_j = constant$, and solved a quadratic optimization problem for α_i and α_j
    - We then projected $(\alpha_i, \alpha_j) \rightarrow [0, C]^2$
- We will now 'scale up' these projections
- In active set methods, the working set changes slowly. Projection methods can solve bound constrained optimization problems with large changes in the working set at each iteration.

Overview

- We can find Δx as the change in x along some steepest descent direction of f without constraints
- Thus, let $x_u^{k+1} = x^k + \Delta x$ be the working set that reduces $f(x)$ without constraints (unbounded)
- To find the constrained working set, we project x_u^{k+1} onto Ω to get x^{k+1}

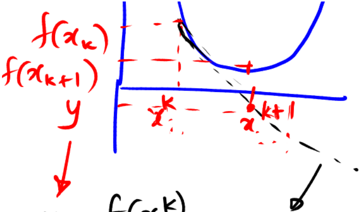
- To project x_u onto the non-empty closed convex set Ω to get the projected point x_p, we solve:

$$x_p = P_\Omega(x_u) = \underset{z \in \Omega}{\operatorname{argmin}} \|x_u - z\|_2^2$$

- That is, the projected point x_p is the point in Ω that is the closest to the unbounded optimal point x_u if Ω is a non-empty closed convex set

Descent direction for a convex function

- For a descent in a convex function f, we must have $f(x^{k+1}) \geq$ Value at x^{k+1} obtained by linear interpolation from x^k
- *ie.* $f(x^{k+1}) \geq f(x^k) + \nabla^\top f(x^k)(x^{k+1} - x^k)$



$$y = f(x^k) + \nabla^T f(x^k)(x^{k+1} - x^k)$$

- Thus, for Δx^k to be a descent direction, it is necessary that $\nabla^\top f(x^k)\Delta x^k \leq 0$
(where $\Delta x^k = x^{k+1} - x^k$)

We want that the point obtained after the projection of x_u^{k+1} to be a descent from x^k for the function f

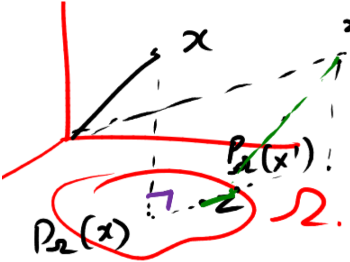
$$\nabla f(x^k) \cdot \Delta x_p \leq 0$$

(where $\Delta x_p = P_\Omega(x_u^{k+1}) - x^k$)

- **Claim:** If $P_\Omega(x)$ is a projection of x, then

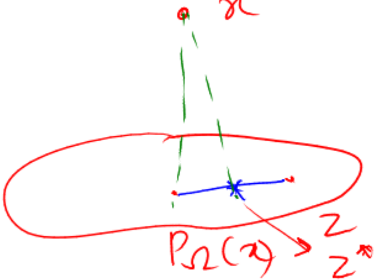
$$\left(z - P_\Omega(x)\right)^\top \left(x - P_\Omega(x)\right) \le 0, \; \forall\, z \in \Omega$$

- That is, the angle between $\left(z - P_\Omega(x)\right)$ and $\left(x - P_\Omega(x)\right)$ is obtuse (or right-angled for the projected point), $\forall z \in \Omega$



Proof for $\langle z - P_\Omega(x), x - P_\Omega(x) \rangle \leq 0$

- To be more general, let us consider an inner product $\langle a, b \rangle$ instead of $a^\top b$
- Let $z^* = (1-\alpha)P_\Omega(x) + \alpha z$, for some $\alpha \in (0,1)$, and $z \in \Omega$
  $\implies z^* = P_\Omega(x) + \alpha(z - P_\Omega(x))$, $z^* \in \Omega$



- Since $P_\Omega(x) = \text{argmin}_{z \in \Omega} \|x - z\|_2^2$,
  $\left\|x - P_\Omega(x)\right\|^2 \leq \|x - z^*\|^2$

$$
\begin{aligned}
&\left\|x - z^*\right\|^2 \\
&= \left\| x - \left(P_\Omega(x) + \alpha(z - P_\Omega(x))\right) \right\|^2 \\
&= \left\|x - P_\Omega(x)\right\|^2 + \alpha^2 \left\|z - P_\Omega(x)\right\|^2 - 2\alpha \left\langle x - P_\Omega(x), z - P_\Omega(x) \right\rangle \\
&\geq \left\|x - P_\Omega(x)\right\|^2
\end{aligned}
$$

$$
\implies \left\langle x - P_\Omega(x), z - P_\Omega(x) \right\rangle \leq \frac{\alpha}{2} \left\|z - P_\Omega(x)\right\|^2, \; \forall \alpha \in (0, 1)
$$

- Thus, the LHS can either be 0 or a negative value. Any positive value of the LHS will lead to a contradiction for some small $\alpha \to 0$
- Hence, we proved that $\left\langle z - P_\Omega(x), x - P_\Omega(x) \right\rangle \leq 0$

- We can also prove that if $\langle x - x^*, z - x^* \rangle \leq 0$, $\forall z \in \Omega$ s.t. $z \neq x^*$, and $x^* \in \Omega$, then

$$x^* = P_\Omega(x) = \underset{\bar{z} \in \Omega}{\operatorname{argmin}} \|x - \bar{z}\|_2^2$$

- Consider $\|x - z\|^2 - \|x - x^*\|^2$
$= \left\|x - x^* + (x^* - z)\right\|^2 - \|x - x^*\|^2$
$= \|x - x^*\|^2 + \|z - x^*\|^2 - 2\langle x - x^*, z - x^* \rangle - \|x - x^*\|^2$
$= \|z - x^*\|^2 - 2\langle x - x^*, z - x^* \rangle$
> 0
- $\implies \|x - z\|^2 > \|x - x^*\|^2$, $\forall z \in \Omega$ s.t. $z \neq x^*$
- This proves that $x^* = P_\Omega(x)$

References

- Yu-Hong Dai, Roger Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. http://link.springer.com/content/pdf/10.1007%2Fs10107-005-0595-2.pdf

These approaches lead to a class of algorithms that start with small values of λ, iteratively increase λ $(\to \infty)$, and in each iteration, we use some descent algorithm to solve the unconstrained minimization problem

$$\min_x f(x) + \lambda B(x)$$

where B is a **barrier function** like

1. $B(x) = \sum I_{C_i}(x)$
2. $B(x) = \max_i \min_{u \in C_i} \|x - u\|^2$
3. $B(x) = \phi_{g_i}(x) = -\frac{1}{t} \log\left(-g_i(x)\right)$
    - Here, $-\frac{1}{t}$ is used instead of λ
    - Lets discuss this in more detail

Option 3: Log barrier function

- The log barrier function is defined as

$$B(x) = \phi_{g_i}(x) = -\frac{1}{t}\log\left(-g_i(x)\right)$$

- It looks like an approximation of $\sum I_{C_i}(x)$
- $f(x) + \sum_i \phi_{g_i}(x)$
  is convex if f and g_i are convex
- We've taken care of the inequality constraints, lets also consider an equality constraint $Ax = b$

- Our objective becomes

$$\min_x f(x) + \sum_i \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

$$\text{s.t. } Ax = b$$

- At different values of t, we get different x^*
- Let $\lambda_i^*(t) = \frac{-1}{t\, g_i\left(x^*(t)\right)}$
- First-order necessary conditions for optimality at $x^*(t)$:
  - $g_i\left(x^*(t)\right) \leq 0$
  - $Ax^*(t) = b$
  - $\nabla f\left(x^*(t)\right) + \sum_{i=1}^m \lambda_i^*(t) \nabla g_i\left(x^*(t)\right) + \nu^*(t)^\top A = 0$
  - $\lambda_i^*(t) \geq 0$
    - Since $g_i\left(x^*(t)\right) \leq 0$ and $t \geq 0$
- $\left(\lambda_i^*(t), \nu^*(t)\right)$ is dual feasible

- $x^*(t)$ minimizes the Lagrangian

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \nu^\top (Ax - b)$$

  - $\nabla L = 0$ at $x^*(t)$

- Lagrange dual function

$$L^*(\lambda, \nu) = \min_x L(x, \lambda, \nu)$$

$$L^*\left(\lambda^*(t), \nu^*(t)\right) = f\left(x^*(t)\right) + \sum_{i=1}^{m} \lambda_i^*(t) g_i\left(x^*(t)\right) + \nu^*(t)^\top \left(Ax^*(t) - b\right)$$

$$= f\left(x^*(t)\right) - \frac{m}{t}$$

  - $\frac{m}{t}$ here is called the *duality gap*
  - As $t \to \infty$, duality gap $\to 0$

- At optimality, primal optimal = dual optimal
  i.e. $p^* = d^*$
- From weak duality,

$$f\left(x^*(t)\right) - \frac{m}{t} \leq p^*$$

$$\implies f\left(x^*(t)\right) - p^* \leq \frac{m}{t}$$

  - The duality gap is always $\leq \frac{m}{t}$
  - The more we increase t, the smaller will be the duality gap

Iterative algorithm

1. Start with $t = t^{(0)}$, $\mu > 1$, and consider ϵ tolerance
2. **Repeat**
   1. **Solve**

      $$x^*(t) = \operatorname*{argmin}_x f(x) + \sum_{i=1}^{m} \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

      $$\text{s.t. } Ax = b$$

   2. If $\frac{m}{t} < \epsilon$, **Quit**
      else, **set** $t = \mu t$

- In the process, we can also obtain $\lambda^*(t)$ and $\nu^*(t)$
- **Convergence of outer iterations:**
  We get ϵ accuracy after $\log\left(\frac{\left(m/\epsilon t^{(0)}\right)}{log(\mu)}\right)$ updates of t

- The inner optimization in the iterative algorithm using a barrier method,

$$x^*(t) = \underset{x}{\operatorname{argmin}}\, f(x) + \sum_i \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

$$\text{s.t. } Ax = b$$

  can be solved using (sub)gradient descent starting from older value of x from previous iteration
- We must start with a strictly feasible x, otherwise $-\log\left(-g_i(x)\right) \to \infty$

- If you set $t^{(0)} = \frac{m}{t}$, we will have only one iteration
- We want to run at least some iterations. Thus, we choose $t^{(0)} \ll \frac{m}{t}$
- We need not obtain $x^*(t)$ exactly at each outer iteration
- If not solving for $x^*(t)$ exactly, we will get ϵ accuracy after *more than* $\log\left(\frac{\left(m/\epsilon t^{(0)}\right)}{log(\mu)}\right)$ updates of t
  - However, solving the inner iteration exactly may take too much time
  - Fewer inner loop iterations correspond to more outer loop iterations

How to find a strictly feasible $x^{(0)}$?

- *Basic Phase I method*

$$x^{(0)} = \underset{x}{\text{argmin}}\, \Gamma$$

$$\text{s.t. } g_i(x) \leq \Gamma$$

- We solve this using the barrier method, and thus will also need a strictly feasible starting $\hat{x}^{(0)}$
- Here,

$$\Gamma = \max_{i=1\ldots m} g_i(\hat{x}^{(0)}) + \delta$$

  where, $\delta > 0$
  - *i.e.* Γ is slightly larger than the largest $g_i(\hat{x}^{(0)})$

- On solving this optimization for finding $x^{(0)}$,
  - If $\Gamma^* < 0$, $x^{(0)}$ is strictly feasible
  - If $\Gamma^* = 0$, $x^{(0)}$ is feasible (but not strictly)
  - If $\Gamma^* > 0$, $x^{(0)}$ is not feasible
- A slightly 'richer' problem can consider different Γ_i for each g_i, to improve numerical precision

$$x^{(0)} = \underset{x}{\operatorname{argmin}} \Gamma_i$$

$$\text{s.t. } g_i(x) \leq \Gamma_i$$

Choice of a good $\hat{x}^{(0)}$ or $x^{(0)}$ depends on the nature/class of the problem, use domain knowledge to decide it