ML and Optimization Lecture 13

- Drawbacks of GD
  - stepsize choice: assumption $f$ is Lipschitz, and we know its Lipschitz constant
    - one soln (GD): backtracking line search
  - the curvature of our functions: suboptimal estimates from using Lipschitzicity of gradients
    - use Hessian information: Newton's method
  - the scalability of GD: generically, if have $n$ datapoints in $\mathbb{R}^d$ each cost of GD is $O(nd)$!
    - use estimates of the gradient: SGD $\underbrace{O(d)}_{\text{cost per iteration}}$

# Stepsize choice via Line search

Recall for GD that $-\nabla f(x)$ points towards the minimizer

$$f(x^*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle$$

$$\Rightarrow \quad \langle -\nabla f(x), x^* - x \rangle \geq 0$$


So far we've been choosing to move to

$$\hat{x} = x - \alpha \nabla f(x)$$

where stepsize $\alpha$ is chosen based on a priori knowledge of $f$ being $\beta$-smooth ($\|\nabla f(x) - \nabla f(y)\|_2 \leq \beta \|x - y\|_2$ for all $x, y$): $\alpha = \frac{1}{\beta}$

Line search chooses step-sizes in an adaptive (but expensive) way to decrease the total cost of optimization.

Idea: minimize along the ray $x - \alpha \nabla f(x)$ the function value

$$\alpha_t = \underset{\alpha \geq 0}{\operatorname{argmin}} \; f(x_t - \alpha \nabla f(x_t))$$

This called exact line search and $\alpha_t$ is called the Curry step length. Not typically used, but can be useful for specific problems.

More generally we use backtracking line search.

Idea of backtracking line search:

T.S. expansion w/ remainder

$$f(x) = f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + r(x - x_t), \quad \text{where } r(\Delta) = o(\|\Delta\|) \text{ as } \Delta \to 0$$

so take $x = x_t + \alpha d_t$ where $d_t$ is a descent direction:

negative

$$f(x_t + \alpha d_t) = f(x_t) + \alpha \langle \nabla f(x_t), d_t \rangle + r(\alpha d_t)$$

By the definition of $o(\|\Delta\|)$, for any $\gamma > 0$, there is an $\varepsilon > 0$ such that

$$|r(\alpha d_t)| < \gamma \|\alpha d_t\|_2 = \alpha \gamma \|d_t\|_2. \quad \text{when } \alpha < \varepsilon$$

Take $\gamma = (c-1)\langle \nabla f(x_t), \frac{d_t}{\|d_t\|_a} \rangle$, where $c \in (0,1)$, so $\gamma > 0$, then

$$\alpha \langle \nabla f(x_t), d_t \rangle + r(\alpha d_t) < \alpha \langle \nabla f(x_t), d_t \rangle +$$
$$(c-1)\alpha \langle \nabla f(x_t), d_t \rangle$$
$$= c\alpha \langle \nabla f(x_t), d_t \rangle$$

so we can guarantee that <u>if</u> $\alpha$ is small enough, we get decrease:

$$f(x_t + \alpha d_t) < f(x_t) + c\alpha \langle \nabla f(x_t), d_t \rangle$$

- Backtracking line search "backtracks" to find a small enough $\alpha$ to ensure decrease.

## Backtracking Line Search

Given: $x_t$, $d_t$ - descent direction (i.e. a vector s.t. $\langle d_t, -\nabla f(x_t) \rangle \geq 0$)

$c \in (0,1)$ - amount of requested decrease

$\beta \in (0,1)$ - backtracking parameter

Return: $\alpha$ s.t.

$$f(x_t + \alpha d_t) < f(x_t) + c\alpha \langle \nabla f(x_t), d_t \rangle$$

Alg

$\alpha \leftarrow 1$

while $f(x_t + \alpha d_t) \geq f(x_t) + c\alpha \langle \nabla f(x_t), d_t \rangle$

$\alpha \leftarrow \beta \alpha$

return $\alpha$

The cost of BTLS depends on:

1) The cost of evaluating $f$
2) How much decrease you ask for (how large $c$ is)
3) The aggressiveness of your backtracking (how small $\beta$ is)

Idea: balance aggressiveness & decrease asked for to make maximal progress per call to BTLS

(Boyd & Vandenberghe. Convex Optimization)

General prescriptions:

$$c \in [.01, .3]$$

$$\beta \in [.5, .8]$$

# Gradient Descent w/ Exact Line Search

(on $\mu$-strongly convex & $\beta$-smooth convex functions)

$$f(x_T) - f(x_*) \leq \left(1 - \frac{1}{\kappa}\right)^{T-1} \left(f(x_1) - f(x^*)\right)$$

where $\kappa = \frac{\beta}{\mu}$ is the condition number of the convex function $f$

$\beta$ = an upper bound on the eigenvalues of $\nabla^2 f$ on the entire domain

$\mu$ = a lower bound on the eigenvalues of $\nabla^2 f$ on the entire domain

$$\kappa \leq \frac{\beta}{\mu} \iff \mu I \preceq \nabla^2 f(x) \preceq \beta I$$

Strong convexity If

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|x - y\|_2^2$$

then we say $f$ is $\mu$-strongly convex

---

Clear that $\mu \leq \beta$, so $K = \frac{\beta}{\mu} \geq 1$

---

The convergence bound says

$$f(x_T) - f(x_*) \leq c^{T-1}\left(f(x_1) - f(x_*)\right)$$

where $c = 1 - \frac{1}{K}$, so $K \approx 1$ implies steepest descent (GD + line search) converges fast.

# Newton's Method

We saw that steepest descent works well when $K = \frac{\beta}{\mu}$ is small, which means $f$ looks locally like a quadratic w/ the same amount of curvature in every direction.

What if we don't have this: e.g. consider

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 \quad \text{where} \quad \nabla^2 f(x) = A^T A$$

is ill-conditioned

Then steepest descent has poor convergence. Notice we solve this in one step with

$$\begin{aligned} x_{OLS} &= (A^T A)^{-1} A^T b \\ &= x_0 - (A^T A)^{-1} A^T (A x_0 - b) \\ &= x_0 - \nabla^2 f(x_0)^{-1} \nabla f(x_0) \end{aligned}$$

Note that $-\nabla^2 f(x_t)^{-1} \nabla f(x_t) = d_t$ is a descent direction.

Why?

We know $\nabla^2 f(x_t) \succ 0$ so its eigenvalues are $> 0$

this means $\nabla^2 f(x_t)^{-1} \succ 0$ as well, so

$$\langle d_t, -\nabla f(x_t) \rangle = \langle \nabla^2 f(x_t)^{-1} \nabla f(x_t), \nabla f(x_t) \rangle$$
$$\geq 0$$

∎

Another view of Newton's method: we explicitly model f's curvature

$$f(x) \approx f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2}(x - x_t)^T \nabla^2 f(x_t)(x - x_t)$$
$$= m(x)$$

Since the approximation $m(x)$ is convex we can approximately minimize f by setting

$$\nabla m(x)_{t+1} = 0 \iff \nabla f(x_t) + \nabla^2 f(x_t)(x_{t+1} - x_t) = 0$$
$$\iff x_{t+1} = x_t - \nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

This suggests we can use $\nabla^2 f(x_t)^{-1} \nabla f(x_t)$ as our search direction.

As before we can line search to choose how far to move in that direction

## Damped/guarded Newton's method

Input: $x_t$, oracles for $f, \nabla f, \nabla^2 f$

Output: $x_T$

Alg

For $t = 1, \dots, T-1$

$$d_t = -\nabla^2 f(x_t)^{-1} \nabla f(x_t)$$

$$\alpha_t \leftarrow \text{BTLS}$$

$$x_{t+1} = x_t - \alpha_t d_t$$

What is the benefit of using curvature info?

There are two phases of the algorithm:

- damped phase: suboptimality goes down by a constant factor at every iteration ($\alpha_t < 1$)
  in this phase: $f(x_t) - f(x_*) \leq \varepsilon$ $O(\log(\frac{1}{\varepsilon}))$
- purely Newton phase: at each iteration, the number of digits of accuracy doubles! ($\alpha_t = 1$, BTCS does nothing)
  in this phase: $f(x_t) - f(x_*) \leq \varepsilon$ $O(\log\log(\frac{1}{\varepsilon}))$

see Boyd & Vandenberghe for details and proof

Drawbacks:

— need to form $\nabla^2 f(x_t) \in \mathbb{R}^{d \times d}$

<u>and</u> solve a linear system

In general this costs $O(d^2)$ memory and $O(d^3)$ operations

so when $d$ is large (say 100K) this is not feasible.