Up to now:

- Stepsize was selected to "optimize" conv. bounds.
- We had diff. functions
- Didn't need to worry about constraints

This lecture:

- How to choose stepsize in practice
- How to deal with non. diff. functions
- " " " " constraints.

Tuning your learning rates:

Up to now we selected step sizes that were useful for analyzing convergence rates, but these are not practical for implementation.

- There are several ways to tune γ in practice, and it's tuning in SGD is a little more "messy" than GD

Eg for GD we do:

- Line search: (direct solve)

$$\gamma_{k+1} = \arg\min_{\gamma} f(w_k - \gamma \nabla f(w_k))$$

Requires many $f(\cdot)$ evals

- Backtracking search / Armijo method
  More efficient.

These approaches are not suitable for SGD.

Q: How to choose γ for SGD?

A: Find one that maximizes convergence!
... But how?

Reminder:

$$\mathbb{E}\|w_{k+1}-w^*\|^2 \leq \left.\begin{array}{l}\mathbb{E}\|w_k - w^*\|^2 - 2\gamma\mathbb{E}\langle\nabla f(w_k), w_k - w^*\rangle \\ \quad + \gamma^2\mathbb{E}\|\nabla f_{S_k}(w_k)\|^2\end{array}\right\} \text{"progress" terms}$$

Goal: Converge as fast as possible

i.e., maximize the "progress" term

$$p(\gamma) = \gamma\mathbb{E}\langle\nabla f(w_k), w_k - w^*\rangle - \gamma^2\mathbb{E}\|\nabla f_{S_k}(w_k)\|^2$$

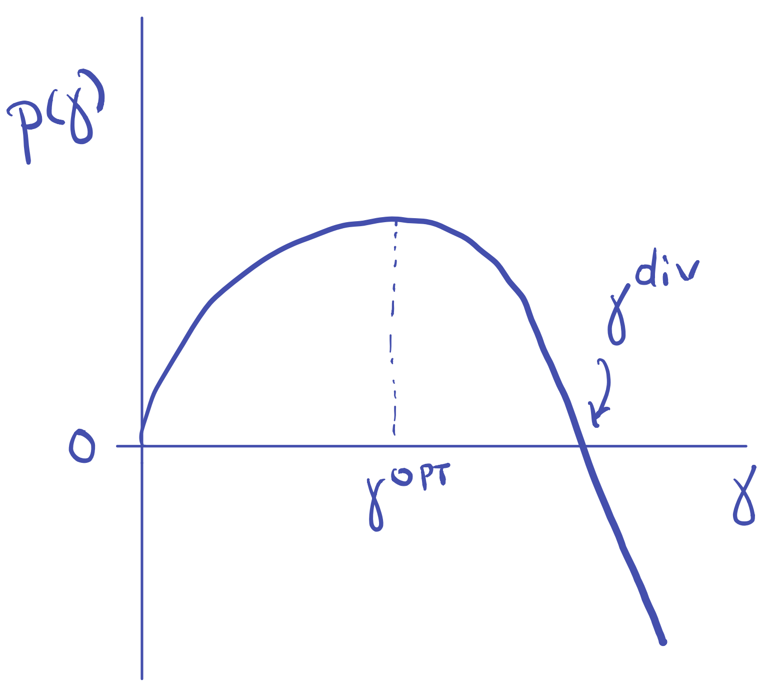
Observe $p(\gamma)$ is quadratic in γ!

Finding the opt γ is equivalent to

$$\underset{\gamma \geq 0}{\operatorname{argmin}}\ p(\gamma) = \gamma^{OPT} = \frac{E\{\langle \nabla f(w_k), w_k - w^* \rangle\}}{E\|\nabla f_{S_k}(w_k)\|^2}$$

Unfortunately we can't compute γ^{OPT} because it requires knowledge of $w_k - w^*$!

But provides a useful intuition!



A meaningful heuristic:

Find γ^{div} and use a slightly smaller stepsize, i.e.

$$\gamma^{OPT} \approx \gamma^{div}/2$$

This principle informs your design process in practice.

"Principle": Choose the largest γ before divergence.

How to choose?
- Grid Search
- Random search.

E.g. Set $\gamma = [10^{-5}\ 10^{-4} \dots 10^{1}\ 1]$

Or active learning, eg "HyperBand"

Part 2: Non differentiable functions

Let $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$

but ∇f_i and ∇f don't always exist.

What to do in these cases?

Solution: Subgradients

$\forall f$ convex, and $\forall x, y$ $\exists$ $g(x)$ s.t.

$$f(y) \geq f(x) + \langle g(x), y-x \rangle$$

Eg.

Fact: at each x there migth exist infinite subgrads

Properties and Examples:

Def: The set of all subgads of f at x is called the subdifferential at x

$$\partial(x) = \{ g(x) \text{ s.t } g(x) \text{ is a subgrad of } f \text{ at } x \}$$

- if f is L-Lipschitz

$$\|g\| \leq L$$

- $f(x) = \max f_i(x)$

$$\partial f(x) = \text{convexhull}\left(\bigcup_{i;\, f_i(x)=f(x)} \partial f_i(x)\right)$$

convex polytope

- $f(x) = \|x\|_1$

$$[g(x)]_i = \begin{cases} \text{sign}(x_i), & x_i \neq 0 \\ [-1, 1], & x_i = 0 \end{cases}$$

- $f(x) = \frac{1}{n} \sum_{i=1}^{n} |a_i^T x - b|$

$$g_i(x) = a_i \cdot \text{sign}(a_i^T x - b_i)$$

<u>Lemma (informal)</u>: When f is L-Lip SGD/GD achieve same rate for diff ~~and~~ non-diff functions.

Part 3: Projections

Q: What happen when we have constraints?

$$\min_{x \in C} f(x)$$

Projected (S)GD:

$$x_{k+1} = P_C(x_k - \gamma \nabla f(x_k))$$

$$P_C(x) = \underset{y \in C}{\operatorname{argmin}} \|x - y\|$$

eg $P_C(x)$ finds the closest point to x wrt Euclidean distance to C.

Main property used for convergence:

When C is convex

$$\| P_C(y) - x^* \|^2 \le \| y - x \|^2$$

Then all convergence guarantees follow through

Q: What is the added cost of convergence?

$$\min_{y \in C} \| x - y \| \text{ is convex}$$

so poly-time solvable, but how fast?

Some interesting cases are cheap!

Examples:

1) $C = \{ x ; \|x\| \leq 1 \}$ (ℓ_2-ball)

$$P_C(x) = \frac{x}{\|x\|}$$

Cost: $O(d)$

2) $\|x\|_\infty \leq 1$ (L_∞-ball)

$$[P(x)]_i = \begin{cases} x_i, & \text{if } |x_i| \leq 1 \\ \text{sign}(x_i), & \text{if } |x_i| > 1 \end{cases}$$

Cost: $O(d)$

3) $\|x\|_1 \leq 1$ (L_1-ball)

[Duchi et. al]

Cost: $O(d)$

So for many important problems the cost is small, but there are interesting cases where $P_C(y)$ is <u>expensive</u>

<u>Example:</u>

$$C = \{ X_{n\times n} : X \succeq 0 \} \quad \left(\begin{array}{c}\text{positive}\\ \text{semidefinite}\\ \text{matrices}\end{array}\right)$$

<u>Projection:</u>

$$P_C(A) = \min_{X \succeq 0} \| A_{n\times n} - X_{n\times n} \|^2$$

when A is symmetric $P_C(A)$ is equivalent to solving EVD and keeping only the positive eigenvalues of A while setting the negative ones to 0.

Cost of EVD $O(d^3)$!

Q: Can we avoid this high cost?

A: Sometimes, by using algorithms similar to Frank-Wolfe.