Upto 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 $\gamma$ in practice, and it's tuning in SGD is a little more "messy" than GD.

E.g. 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 SfD.

Q: How to choose $\gamma$ for SfD?

A: Find one that maximizes convergence!

... But how?

Reminder:

$E\|w_{k+1} - w^*\|^2 \leq E\|w_k - w^*\|^2 - 2\gamma E\langle \nabla f(w_k), w_k - w^* \rangle \gamma^2 \text{"progress" terms} + \gamma^2 E\|\nabla f_{sk}(w_k)\|^2$

Goal: Converge as fast as possible

i.e., maximize the "progress" term

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

Observe $p(\gamma)$ is quadratic in $\gamma$!
Finding the opt $\gamma$ is equivalent to

$$\arg\min_{\gamma \geq 0} p(\gamma) = \gamma^{opt} = \frac{E[\mathcal{D}(w_k), w_k - w^*]^2}{E \| f_{sk}(w_k) \|^2}$$

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

But provides a useful intuition!

![Graph showing the function $p(\gamma)$ with $\gamma^{opt}$ and $\gamma_{div}$ marked.]
A meaningful heuristic:

Find $\gamma_{\text{div}}$ and use a slightly smaller stepsize, i.e.

$$\gamma_{\text{opt}} = \frac{\gamma_{\text{div}}}{2}$$

This principle informs your design process in practice.

"Principle": Choose the largest $\gamma$ before divergence.

How to choose?
- Grid Search
- Random search

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

Or active learning, e.g. "HyperBand"
Part 2: Non differentiable functions

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

but \( \nabla f_i \) and \( \nabla f \) don’t always exist.

What to do in these cases?

Solution: Subgradients

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

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

Eg. subgradients of \( f \)
Fact: at each $x$ there might exist infinite subgradients.

Properties and Examples:

Def: The set of all subgradients of $f$ at $x$ is called the subdifferential at $x$.

$$\partial f(x) = \{ g(x) \mid g(x) \text{ is a subgradient of } f \text{ at } x \}$$

- if $f$ is $L$-Lipschitz

$$\|g\| \leq L$$
- \( f(x) = \max \{ f_i(x) \} \)

\[
f(x) = \text{convex hull}( \cup i \cdot f_i(x) ) \\
\text{convex polytope}
\]

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

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

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

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

**Lemma (informal):** When \( f \) is \( L \)-Lip SGD/CD 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:

\[ x_{k+1} = P_C (x_k - y \nabla f(x_k)) \]

\[ P_C (x) = \arg \min_{y \in C} \| x - y \| \]

eg. \( P_C (x) \) finds the closest point to \( x \) w.r.t Euclidean distance to \( C \).
Main property used for convergence:

When $A$ is convex

$$\|Px(y)-x^*\|^2 \leq \|y-x\|^2$$

Then all convergence guarantees follow through.

Q: What is the added cost of convergence?

$$\min \|x-y\|$$ is convex

Yes, so poly-time solvable, but how fast?

Some interesting cases are cheap!
Examples:

1) \( C = \delta \times j, \|x\| \leq \delta \) (\( L_2 \)-ball)

\[ P_c(x) = \frac{x}{\|x\|} \]

Cost: \( O(d) \)

2) \( \|x\|_{\infty} \leq 1 \) (\( L_\infty \)-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) \( \|x\|_1 \leq t \) (\( L_1 \)-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 expensive.

**Example:**

$$C = \{ X_{n \times n} : X \succ 0 \} \quad \text{(positive semidefinite matrices)}$$

**Projection:**

$$P_c(A) = \min X_{n \times n} - X_{n \times n} \| X \succ 0$$

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.