Unconstrained Optimization

What is it?

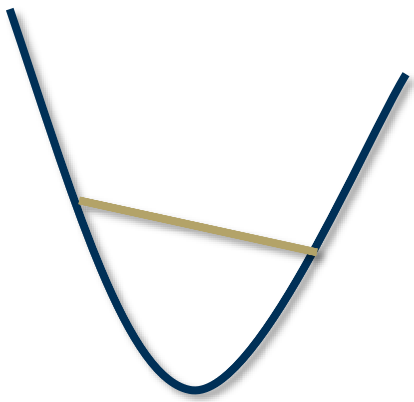
For our purposes today,

$$\underset{\boldsymbol{x} \in \mathbb{R}^d}{\text{minimize}} \; f(\boldsymbol{x})$$

The "Easy" Function Classes

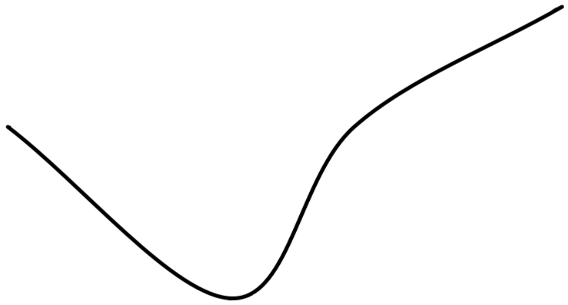
- Convex

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \;\leq\; \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$



- PL (Polyak-Łojasiewicz)

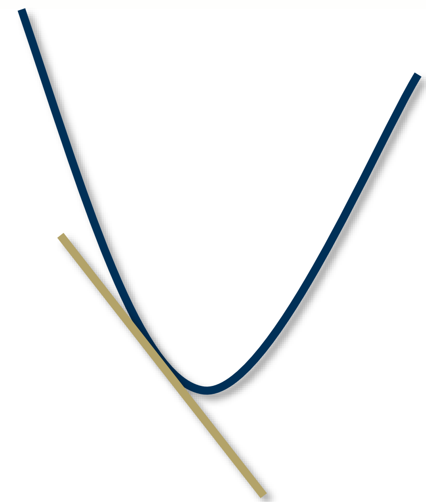
$$\|\nabla f(\boldsymbol{x})\|_2^2 \geq 2m\left(f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)\right)$$



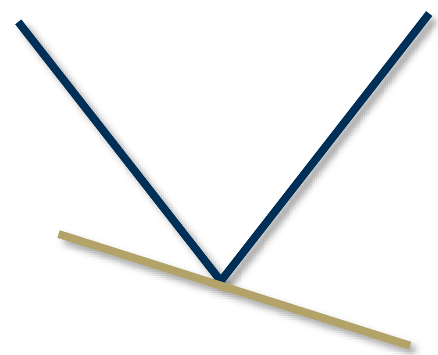
(no spurious local minima)

Non-Differentiable Convex Functions

- What is a gradient, really?
  - (for convex functions)
  - Linear lower bound



- Subgradient
  - Not necessarily unique at non-differentiable points
  - Subgradient analogs to gradient methods exist



Regularization

Change the problem from

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{x})$$

to

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{x}) + \lambda r(\boldsymbol{x})$$

e.g.

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{x}) + \lambda\, ||\boldsymbol{x}||^2$$

Some Benefits of Regularization

- Better optimization landscape for chosen optimization algo
- Infinitely many solutions -> one unique solution
- Lower variance in solution / less overfitting
- Prefer certain solutions / incorporate domain knowledge
  - e.g. L1 regularizer promotes sparse solution

Step-Size Line Search

- Sometimes gradient computations are expensive, but loss computations are cheap
- We need to make the most of every gradient computation we waited for

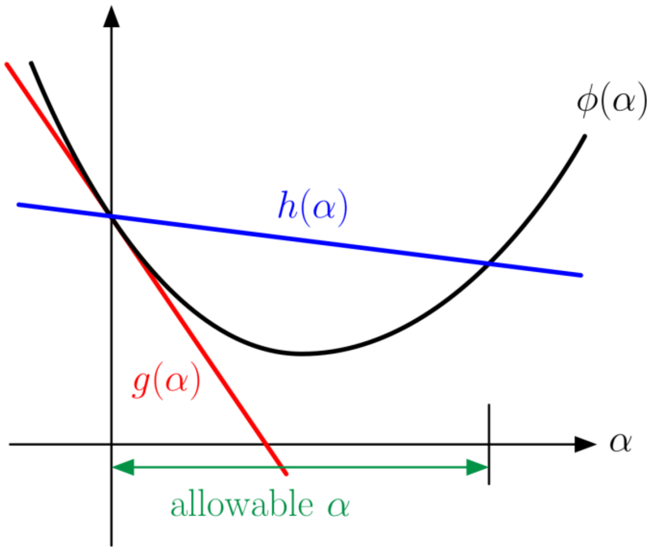
$$\underset{\alpha \geq 0}{\text{minimize}} \;\; \phi(\alpha) \qquad\qquad \phi(\alpha) = f\left(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k\right)$$

- Sometimes this subproblem can be solved in closed form (e.g. convex quadratic problems)
- Otherwise, need a different approach...

Step-Size Line Search

- Backtracking: start with a big step, then shrink until *allowable*
  - *Allowable*: Armijo condition

$$f\left(\boldsymbol{x}_k\right) - f\left(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k\right) \geq c\alpha \left\langle \boldsymbol{d}_k, \nabla f\left(\boldsymbol{x}_k\right)\right\rangle$$



$$\phi(\alpha) \leq h(\alpha) := f\left(\boldsymbol{x}_k\right) + c\alpha \left\langle \boldsymbol{d}_k, \nabla f\left(\boldsymbol{x}_k\right)\right\rangle$$

$$\phi(\alpha) \geq g(\alpha) := f\left(\boldsymbol{x}_k\right) + \alpha \left\langle \boldsymbol{d}_k, \nabla f\left(\boldsymbol{x}_k\right)\right\rangle$$

Step-Size Line Search

Backtracking line search

Input: $\boldsymbol{x}_k$, $\boldsymbol{d}_k$, $\bar{\alpha} > 0$, $c \in (0, 1)$, and $\rho \in (0, 1)$.

Initialize: $\alpha = \bar{\alpha}$

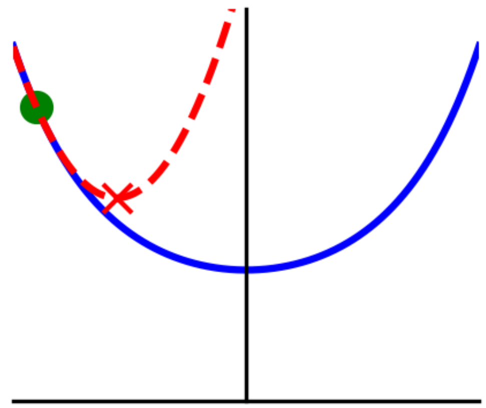
while $\phi(\alpha) > h(\alpha)$ **do**

$\alpha = \rho\alpha$

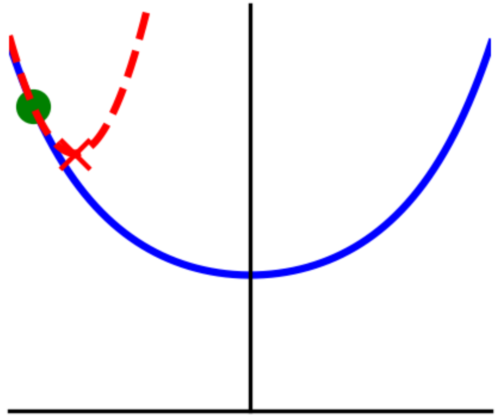
end while

Proximal Methods

Recall Newton's method uses local quadratic, which costs a Hessian computation



What if we just used *some* quadratic (that still agrees with the gradient)

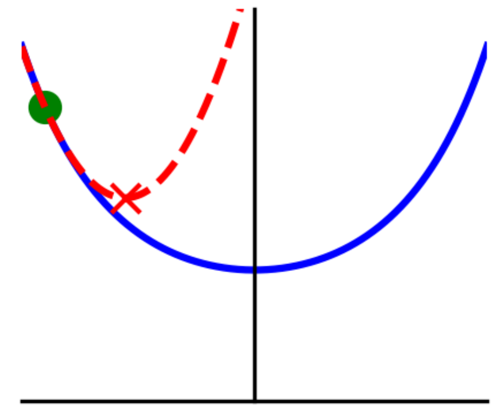


Proximal Methods

$$||\boldsymbol{x}||_{\boldsymbol{A}}^2 := \langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x} \rangle$$

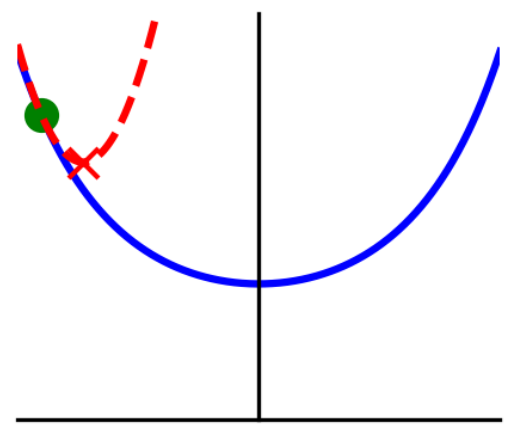
$$\underset{\boldsymbol{x}_{k+1}}{\text{minimize}}\, f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{1}{2} ||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||_{\nabla^2 f(\boldsymbol{x}_k)}^2$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left(\nabla^2 f(\boldsymbol{x}_k)\right)^{-1} \nabla f(\boldsymbol{x}_k)$$



$$\underset{\boldsymbol{x}_{k+1}}{\text{minimize}}\, f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{1}{2} ||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||_{\frac{1}{\eta}\mathbf{I}}^2$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \nabla f(\boldsymbol{x}_k)$$



(This is gradient descent!)

Proximal Methods

The general form for a proximal step is

$$\underset{\boldsymbol{x}_{k+1}}{\text{minimize}}\, f(\boldsymbol{x}_k) + \frac{1}{2\eta}\,||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k||^2$$

- To get GD, we substituted f(x) with its first-order approximation
- Sometimes it's better if we only use a first-order approximation for *part* of the objective function...

Proximal Methods

The LASSO problem is

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\, ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}||_2^2 + \lambda\, ||\boldsymbol{\theta}||_1$$

What if we only used a first-order approximation for the L2 term, and kept the L1 term as-is? This is **ISTA** (iterative soft-thresholding algorithm), and it's way better than the standard subgradient method

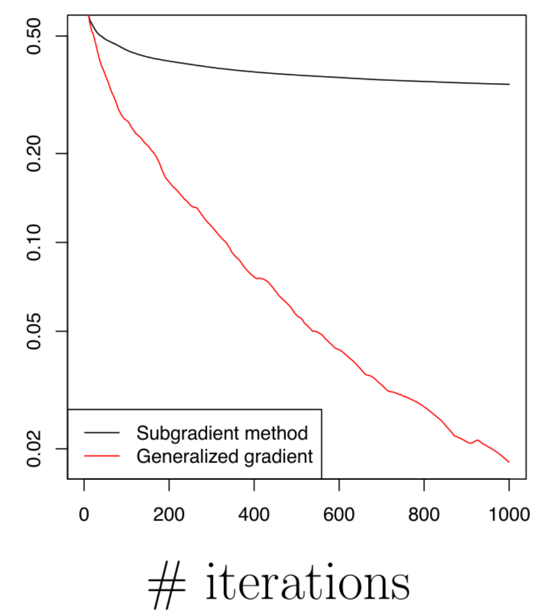


Chart taken from the lecture notes of Geoff Gordon and Ryan Tibshirani. ISTA is red.

How Good is GD, Really?

- Hard to say in general, lots of things matter

- Strong Convexity
  - Lower bounded by (nonzero) quadratic

$$\nabla^2 f(\boldsymbol{x}) \succeq m\mathbf{I}$$

- Smoothness
  - Upper bounded by quadratic

$$\nabla^2 f(\boldsymbol{x}) \preceq M\mathbf{I}$$

How Good is GD, Really?

- When f is smooth, error decreases as $O(1/k)$
- When f is smooth *and* strongly convex, error decreases as $O(r^k)$ for some $r<1$ (called *linear convergence*)

- For reference, Newton's method has *quadratic convergence*, which is faster than linear convergence

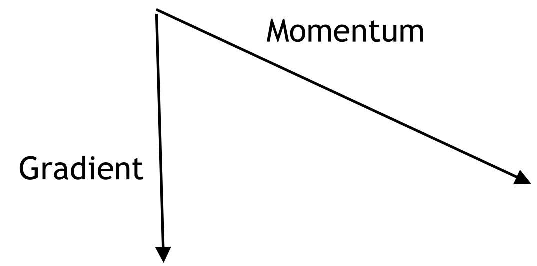
GD with Momentum

- Some possible reasons to use momentum:
  - Better convergence rates (not all problems)
  - Better at avoiding spurious local minima
  - Prefers certain solutions to others (not always desirable)
  - Computing stochastic gradients (in batches), so want an integral-component to the steps

GD with Momentum

- Polyak's heavy-ball method

$$
\begin{array}{l}
\textbf{for } k = 0, 1, 2, \ldots \textbf{ do} \\
\quad \textbf{if } k = 0 \textbf{ then} \\
\quad\quad \boldsymbol{b}_{k+1} \leftarrow \nabla f(\boldsymbol{x}_k) \\
\quad \textbf{else} \\
\quad\quad \boldsymbol{b}_{k+1} \leftarrow \mu \boldsymbol{b}_k + \nabla f(\boldsymbol{x}_k) \\
\quad \textbf{end if} \\
\quad \boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \boldsymbol{b}_{k+1} \\
\textbf{end for}
\end{array}
$$



  - torch.optim.SGD
  - O(1/k) for smooth, strongly convex functions
  - (Conjugate Gradients is a variant of heavy ball tuned for quadratics. It's the fast way to compute $A^{-1}b$ in high dimensions)

GD with Momentum

- Nesterov's method

for $k = 0, 1, 2, \ldots$ **do**

    if $k = 0$ **then**

        $\boldsymbol{b}_{k+1} \leftarrow \nabla f(\boldsymbol{x}_k)$

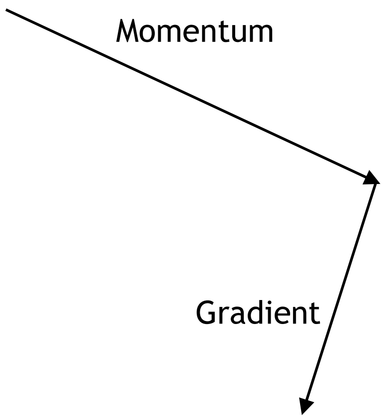
    else

        $\boldsymbol{b}_{k+1} \leftarrow \mu \boldsymbol{b}_k + \nabla f(\boldsymbol{x}_k)$

    end if

    $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \left(\mu \boldsymbol{b}_{k+1} + \nabla f(\boldsymbol{x}_k)\right)$

end for



- torch.optim.SGD
- O(1/k^2) for smooth, strongly convex functions

GD with Momentum

- AdaGrad

$$
\begin{aligned}
&\boldsymbol{s}_0 \leftarrow 0 \\
&\textbf{for } k = 0, 1, 2, \ldots \textbf{ do} \\
&\quad \boldsymbol{s}_{k+1} \leftarrow \boldsymbol{s}_k + (\nabla f(\boldsymbol{x}_k))^2 \qquad \text{(element-wise)} \\
&\quad \boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \frac{\nabla f(\boldsymbol{x}_k)}{\sqrt{\boldsymbol{s}_{k+1}} + \epsilon} \qquad \text{(element-wise)} \\
&\textbf{end for}
\end{aligned}
$$

  - torch.optim.Adagrad
  - More "equitable" trajectory
  - Step size naturally shrinks over time

GD with Momentum

- RMSprop

$$
\begin{aligned}
&\boldsymbol{v}_0 \leftarrow 0 \\
&\boldsymbol{b}_0 \leftarrow 0 \\
&\textbf{for } k = 0, 1, 2, \ldots \textbf{ do} \\
&\quad \boldsymbol{v}_{k+1} \leftarrow \alpha \boldsymbol{v}_k + (1-\alpha)\left(\nabla f(\boldsymbol{x}_k)\right)^2 \qquad \text{(element-wise)} \\
&\quad \boldsymbol{b}_{k+1} \leftarrow \mu \boldsymbol{b}_k + \frac{\nabla f(\boldsymbol{x}_k)}{\sqrt{\boldsymbol{v}_{k+1}} + \epsilon} \qquad \text{(element-wise)} \\
&\quad \boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \boldsymbol{b}_{k+1} \\
&\textbf{end for}
\end{aligned}
$$

  - torch.optim.RMSprop
  - Gradient history is gradually "forgotten"
  - Incorporates a momentum-like term

GD with Momentum

- Adam

$$
\begin{aligned}
&\boldsymbol{v}_0 \leftarrow 0 \\
&\boldsymbol{m}_0 \leftarrow 0 \\
&\textbf{for } k = 0, 1, 2, \dots \textbf{ do} \\
&\quad \boldsymbol{v}_{k+1} \leftarrow \beta_2 \boldsymbol{v}_k + (1 - \beta_2) \left(\nabla f(\boldsymbol{x}_k)\right)^2 \qquad \text{(element-wise)} \\
&\quad \boldsymbol{m}_{k+1} \leftarrow \beta_1 \boldsymbol{m}_k + (1 - \beta_1) \nabla f(\boldsymbol{x}_k) \qquad \text{(element-wise)} \\
&\quad \hat{\boldsymbol{v}}_{k+1} \leftarrow \tfrac{1}{1-\beta_2^{k+1}} \boldsymbol{v}_{k+1} \\
&\quad \hat{\boldsymbol{m}}_{k+1} \leftarrow \tfrac{1}{1-\beta_1^{k+1}} \boldsymbol{m}_{k+1} \\
&\quad \boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \gamma \tfrac{\hat{\boldsymbol{m}}_{k+1}}{\sqrt{\hat{\boldsymbol{v}}_{k+1}} + \epsilon} \qquad \text{(element-wise)} \\
&\textbf{end for}
\end{aligned}
$$

- torch.optim.Adam
- Uses a more classical momentum term
- A more dynamic approach increasing/decreasing step sizes