# Semi-Definite Programming (SDP)

$$\text{minimize} \quad c^T x$$
$$\text{subject to} \quad F(x) \preccurlyeq 0$$

- $F(x) = G + \sum_{i=1}^{n} x_i F_i$
- $G$, $F_i$, $i = 1, 2, \ldots, n$ are $m \times m$ matrices
- $F(x) \preccurlyeq 0$ is a.k.a. **Linear Matrix Inequality**
- This is a **convex** optimization problem since

$$\{x : F(x) \preccurlyeq 0\} \text{ is convex}$$

Proof: If $F(x) \preccurlyeq 0$, $F(y) \preccurlyeq 0$, then $\forall \alpha \in [0, 1]$

$$F(\alpha x + (1-\alpha) y) = G + \sum_{i=1}^{n} (\alpha x_i + (1-\alpha) y_i) F_i$$
$$= \alpha F(x) + (1-\alpha) F(y) \preccurlyeq 0$$

- Multiple LMI constraints can be combined:

$$\tilde{F} = \tilde{G} + \sum_{i=1}^{n} x_i \tilde{F}_i \preccurlyeq 0$$
$$\hat{F} = \hat{G} + \sum_{i=1}^{n} x_i \hat{F}_i \preccurlyeq 0$$

$$\equiv \quad F = G + \sum x_i F_i \preccurlyeq 0$$

with $G = \begin{bmatrix} \tilde{G} & 0 \\ 0 & \hat{G} \end{bmatrix}$, $F_i = \begin{bmatrix} \tilde{F}_i & 0 \\ 0 & \hat{F}_i \end{bmatrix}$

SDP framework captures many interesting convex programs. Efficient computational methods to solve SDP's (barrier, interior-point).

Example 1 Linear Programming (LP)

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{s.t. } Ax \leq b \end{aligned}$$

SDP

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{s.t. } \operatorname{diag}(Ax-b) \preccurlyeq 0 \end{aligned}$$

Example 2

$$\begin{aligned} &\text{minimize } \frac{(c^T x)^2}{d^T x} \\ &\text{s.t. } Ax \leq b \end{aligned}$$

$d^T x > 0$ whenever $Ax \leq b$

Trick: use auxilliary variable $t$ as upper bound on objective

$$\equiv \quad \begin{aligned} &\text{minimize } t \\ &\text{s.t. } Ax \leq b \\ &\frac{(c^T x)^2}{d^T x} \leq t \end{aligned}$$

$$\equiv \quad \begin{aligned} &\text{minimize } t \\ &\text{s.t. } \begin{bmatrix} \operatorname{diag}(Ax-b) & 0 & 0 \\ 0 & -t & -c^T x \\ 0 & -c^T x & -d^T x \end{bmatrix} \preccurlyeq 0 \end{aligned}$$

# Alternating Direction Method of Multipliers (ADMM)

Recall Method of Multipliers:

$$\text{minimize} \quad f(x)$$
$$\text{s.t.} \quad h(x) = 0$$

$$L_c(x, \lambda) = f(x) + \lambda^T h(x) + c \|h(x)\|^2$$

$$x^{(k)} \in \arg\min_x L_c(x, \lambda^{(k)})$$

$$\lambda^{(k+1)} = \lambda^{(k)} + c\, h(x^{(k)})$$

- Special case with $f$ convex and $h(x) = Ax - b$ can show convergence to opt. for $c$ large enough

Now suppose the function being optimized and the linear constraint are decomposable:

$$\text{minimize} \quad f(x) + g(z)$$
$$\text{s.t.} \quad Ax + Bz - b = 0$$

$$L_c(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - b) + \frac{c}{2} \|Ax + Bz - b\|^2$$

ADMM:

$$x^{(k+1)} = \arg\min_x L_c(x, z^{(k)}, \lambda^{(k)})$$

$$z^{(k+1)} = \arg\min_z L_c(x^{(k)}, z, \lambda^{(k)})$$

$$\lambda^{(k+1)} = \lambda^{(k)} + c\,(Ax^{(k+1)} + Bz^{(k+1)} - b)$$

# Accelerated Gradient Descent

Recall Gradient Descent: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

If $f$ is convex and has $L$-Lipschitz gradients,

$$f(x_N) - f(x^*) \sim O\left(\frac{1}{N}\right) \quad \text{if } \alpha_k = \alpha \in \left(0, \frac{2}{L}\right)$$

Is this the best rate of convergence under these assumptions?

## Accelerated (Momentum) Gradient:

Introduce memory in gradient descent iteration:

"Heavy Ball" Method (Polyak 1964):

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \underbrace{\beta(x_k - x_{k-1})}_{\text{momentum}}$$

momentum term accelerates GD near minimum

## Nesterov's Accelerated Gradient (1983+)

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$$

$$y_{k+1} = x_{k+1} + \underbrace{\frac{\lambda_k - 1}{\lambda_{k+1}}}_{\mu_{k+1}}(x_{k+1} - x_k)$$

$$\lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}, \qquad \lambda_0 = 0$$

$$x_{k+1} = x_k + \mu_k(x_k - x_{k-1}) - \frac{1}{L}\nabla f\left(x_k + \mu_k(x_k - x_{k-1})\right)$$

Can show that for NAG,

$$f(x_N) - f(x^*) \sim O\left(\frac{1}{N^2}\right) \leftarrow \text{Best rate}$$

# Stochastic Gradient Descent (SGD)

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x) = f(x)$$

Since $\nabla \left( \frac{1}{m} \sum_{i=1}^{m} f_i(x) \right) = \frac{1}{m} \sum_{i=1}^{n} \nabla f_i(x) = \nabla f(x)$

GD iteration is:

$$x_{k+1} = x_k - \frac{\alpha_k}{m} \sum_{i=1}^{m} \nabla f_i(x_k), \quad k = 0, 1, \ldots$$

In stochastic Gradient Descent:

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k), \quad k = 0, 1, \ldots$$

where $i_k \in \{1, \ldots, m\}$ is chosen index at time $k$.

Possible rules for choosing index:

1) Cyclic: $1, 2, \ldots, m, 1, 2, \ldots, m, \ldots$
2) Random: $i_k$ chosen uniformly at random.

$$\Rightarrow E[\nabla f_{i_k}(x)] = \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x) = \nabla f(x)$$

I.e. SGD uses unbiased estimate of $\nabla f(x)$ at each time step.

## Advantages of SGD

- Computation per iteration independent of $m$
- Less memory usage.

Disadvantage Slower convergence rate, especially near minimum.

SGD after m updates:

$$x_{k+m} = x_k - \alpha \sum_{i=1}^{m} \nabla f_i(x_{k+i-1})$$

Full gradient with step-size $\alpha m$,

$$x_{k+1} = x_k + \alpha \sum_{i=1}^{m} \nabla f_i(x_k)$$

SGD needs diminishing stepsize to converge

For convex $f$ with $L$-Lipschitz gradients:

Full GD: $f(x_N) - f(x^*) \sim O\left(\frac{1}{N}\right)$

SGD: $E[f(x_N)] - f(x^*) \sim O\left(\frac{1}{\sqrt{N}}\right)$

Even with strong convexity added:

$$E[f(x_N)] - f(x^*) \sim O\left(\frac{1}{N}\right)$$

How can we improve SGD convergence?

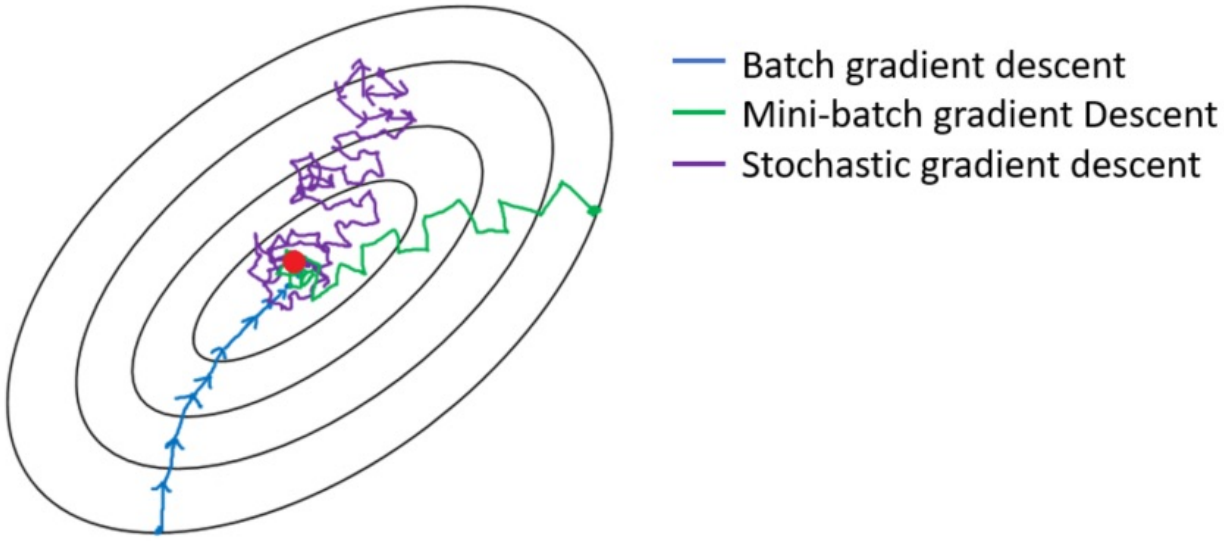
Mini-batches:

$$x_{k+1} = x_k - \alpha_k \frac{1}{b} \sum_{i \in I_k} \nabla f_i(x_k),$$

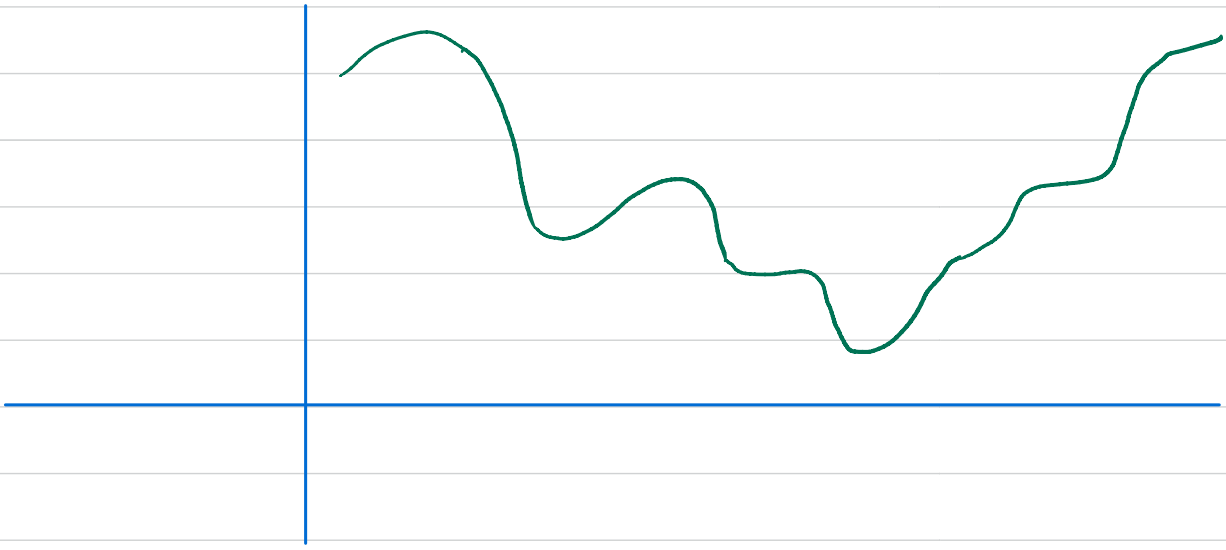
$I_k \subseteq \{1, \ldots, m\}$, $|I_k| = b \ll m$.

For Lipschitz gradients, with mini-batches

$$E[f(x_N)] - f(x^*) \sim O\left(\frac{1}{\sqrt{bN}} + \frac{1}{N}\right)$$




## SGD and Nonconvex optimization



SGD empirically shown to escape bad local minima and saddle points.

## Coordinate Descent

Initialize $x^0$

For $k = 1, 2, \ldots$

For $i = 1, \ldots n$

$$x_i^{k+1} = \arg\min_{x_i \in \mathbb{R}} f(x_i, w_{-i}^k)$$

end

## Gauss-Seidel Update

$$w_{-i}^k = (x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_{i+1}^k, \ldots x_n^k)$$

## Jacobi Update

$$w_{-i}^k = (x_1^k, \ldots, x_{i-1}^k, x_{i+1}^k, \ldots x_n^k)$$

## Remarks

1. Can choose to only one coordinate to update in each iteration
   - random
   - Gauss-Southwell: $i = \arg\max_j \left| \frac{\partial}{\partial x_j} f(x^k) \right|$
2. Replace arg min above by gradient update
3. Can "block" coordinates
4. Convergence under convexity and smoothness