Recall :

GD : $$\boxed{x^{(t+1)} = x^{(t)} - \mu^{(t)} \nabla f(x^{(t)})}$$

can be fixed, or variable

e.g. decreasing

or chosen via linesearch

GD w/ momentum :

$$\boxed{x^{(t+1)} = \underbrace{x^{(t)} - \mu \nabla f(x^{(t)})} + \underbrace{\beta (x^{(t)} - x^{(t-1)})}}$$

Variation : Nesterov's Accelaration

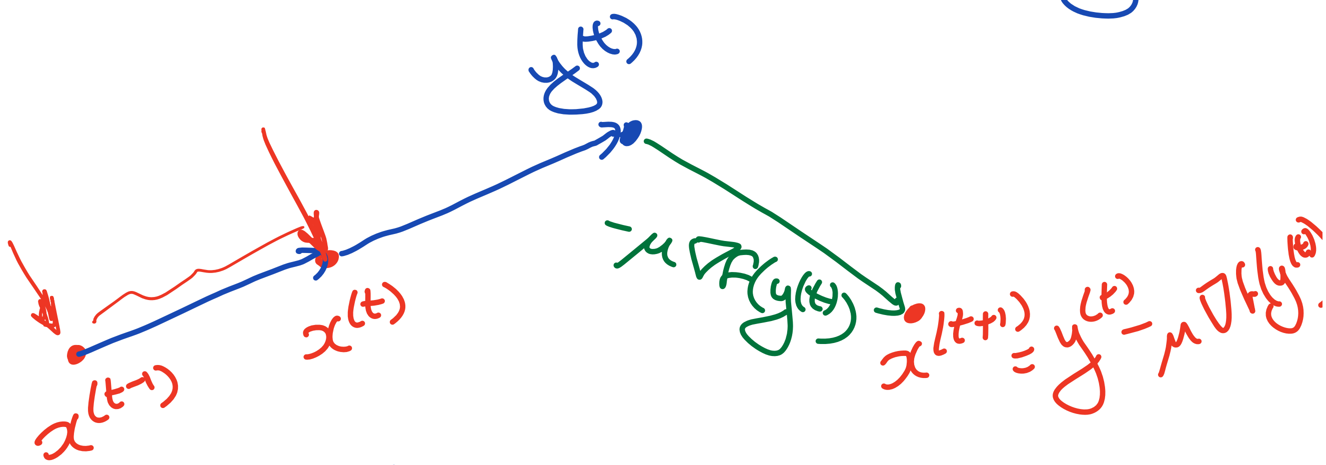
$$y^{(t)} = \underbrace{x^{(t)} + \beta (x^{(t)} - x^{(t-1)})} \quad (1)$$

$$x^{(t+1)} = y^{(t)} - \mu \nabla f(y^{(t)}) \quad (2)$$

Interpretation:

① Take a "momentum step" so you land at $y^{(t)}$

② Take a GD step from $y^{(t)}$.



We can of course combine ① & ② to get

$$\boxed{x^{(t+1)} = x^{(t)} + \beta\left(x^{(t)} - x^{(t-1)}\right) - \mu \nabla f\left(x^{(t)} + \beta\left(x^{(t)} - x^{(t-1)}\right)\right)}$$

↳ nesterov's accelaration

$\boxed{\text{Converges at an accelerated rate for } \underline{\text{any}} \text{ convex problem}}$

with rate $= \sqrt{\dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}}}$

$f(x) = \frac{1}{2} x^T A x$

$\kappa = \dfrac{\lambda_{max}}{\lambda_{min}}$

arises with the optimal choice of

$\mu = \dfrac{1}{\lambda_{max}}, \quad \beta = \dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$

in the quadratic case

compare to GD, GD/momentum

- used in practice
- speed up GD significantly.

Conjugate Gradient Methods

- Originally developed in the 50's by Hestenes & Steifel for solving large linear systems i.e. $Ax = b$

- First nonlinear conjugate gradient (CG) methods were developed in the 60's by Fletcher & Reeves for solving non-linear optimization problems.

Basic Idea:

Suppose that we want to minimize

$$\boxed{\Phi(x) = \frac{1}{2} x^T A x - b^T x} \longrightarrow (*)$$

where A is a symmetric positive semidefinite $n \times n$ matrix.

Notice: • Φ is convex &

• $\nabla \Phi(x) = Ax - b$

$\Rightarrow$ the optimizer satisfies $Ax^* = b$

(which links solving linear system with optimizing (*).)

Where does the word "conjugate" in CG methods come from?

Def'n: A set of vectors $\{p_1, \dots, p_\ell\}$

$p_i \neq p_j \ \forall i \neq j$

is conjugate with respect to a sym. P~~S~~D matrix A if

$$p_i^T A p_j = 0 \quad \forall i \neq j$$

Example: The standard basis vector $e_1, e_2, \dots, e_\ell \in \mathbb{R}^n$ where $\ell \leq n$ are conjugate w.r.t to I

$$\left(\text{bec} \quad e_i^T I e_j = 0 \quad \forall i \neq j\right)$$

How does conjugacy help us?

It turns out we can minimize $\Phi(x)$ in $(*)$ in at most n steps by successively minimizing it along the individual directions in a conjugate set.

How? Conjugate direction method:

$$\boxed{x^{(t+1)} = x^{(t)} + \underbrace{\alpha_t}_{\text{scalar}} \underbrace{P_t}_{\text{vector} \in \mathbb{R}^n}} \quad (1)$$

where $\alpha_t = \underset{a \in \mathbb{R}}{\operatorname{argmin}} \; \Phi(\underbrace{x^{(t)}}_{\text{known}} + \overset{\text{unknown}}{a} \underbrace{P_t}_{\text{known}})$

(i.e. pick the best a)

(i.e. perform an exact line search)

Since Φ is quadratic, we can solve for α_t exactly:

$$\underline{\Phi}(x + aP) = \frac{1}{2}(x+ap)^T A (x+ap) - (x+ap)^T b$$

x given, P given, a: trying to optimize

To find optimal a, solve $\frac{d\underline{\Phi}}{da}(\underbrace{x+ap}) = 0$

Chain rule $\Rightarrow$ $\underbrace{\underline{\Phi}'(x+ap)}_{[A(x+ap)-b]^T} \cdot \underbrace{(x+ap)'}_{P} = 0$

thus $(A(x+ap)-b)^T p = 0$

$$\Rightarrow a^* = \boxed{\alpha_t = \frac{(b - Ax^{(t)})^T P_t}{P_t^T A P_t}}$$

Notice!

$$\boxed{\alpha_t = \frac{-\nabla\underline{\Phi}(x^{(t)})^T P_t}{P_t^T A P_t}}$$

(bec $b - Ax = -\nabla \underline{\Phi}(x)$)

Theorem: For any $x^{(0)}$ the sequence $\{x^{(t)}\}$ generated by 1 converges to x^*, the optimizer of Φ in at most n-steps.

$\Phi: \mathbb{R}^n \to \mathbb{R}$

Won't prove it: Proof is in "Nocedal & Wright (2006)" Chapter 5.

What is the conjugate gradient method?

Unlike the conjugate directions method, here we compute P_t as part of the algorithm.

- Can compute P_t using only P_{t-1}
- So, don't need to store or compute with $P_0, P_1, \ldots, P_{t-2}$

- P_t is automatically conjugate to $P_0, P_1, \ldots, P_{t-1}$!

How? :

$$\underbrace{P_t}_{\text{vector}} = - \underbrace{\nabla \Phi(x^{(t)})}_{\text{vector}} + \underbrace{\overbrace{\beta_t}^{\text{scalar}}}_{\text{chosen to enforce conjugacy}} \underbrace{P_{t-1}}_{\text{vector}}$$

How to pick β_t then?

$$P_{t-1}^T A \left[P_t = - \nabla \Phi(x^{(t)}) + \beta_t P_{t-1} \right]$$

$$\Rightarrow \underbrace{P_{t-1}^T A P_t}_{0 \text{ by conjugacy}} = - \overbrace{P_{t-1}^T A \nabla \Phi(x^{(t)})}^{\text{scalar}} + \beta_t \underbrace{P_{t-1}^T A P_{t-1}}_{\text{scalar}}$$

$$\Rightarrow \beta_t = \frac{P_{t-1}^T A \nabla \Phi(x^{(t)})}{P_{t-1}^T A P_{t-1}}$$

What about β_0? Just set $\beta_0 = 0$

$\Rightarrow$ we have an algorithm.

- Initialize $x^{(0)}$, $P_0 = -\nabla \Phi(x^{(0)})$
- for $t = 1, 2, \ldots$

  * $\beta_t = \dfrac{P_{t-1}^T A \nabla \Phi(x^{(t)})}{P_{t-1}^T A P_{t-1}}$

  * $P_t = -\nabla \Phi(x^{(t)}) + \beta_t P_{t-1}$

  * $d_t = -\dfrac{\nabla \Phi(x^{(t)})^T P_t}{P_t^T A P_t}$

  * $x^{(t+1)} = x^{(t)} + d_t P_t$

Should be reminiscent of momentum

CG: Version 0

Theorem: If $\Phi(x) = \frac{1}{2} x^T A x - b^T x$ with A being $n \times n$ symmetric P~~S~~D, then CG converges to x^* in at most n steps

Proof: See reference

More efficient implementation (needs properties of CG to derive, won't do it) is

- Initialize $x^{(0)}$; $r_0 = Ax^{(0)} - b = \nabla \Phi(x^{(0)})$
  $p_0 = -r_0$

- $\alpha_t = \dfrac{r_t^T r_t}{p_t^T A p_t}$

- $x^{(t+1)} = x^{(t)} + \alpha_t p_t$

- $r_{t+1} = r_t + \alpha_t A p_t$
- $\beta_{t+1} = \dfrac{r_{t+1}^T r_{t+1}}{r_t^T r_t}$
- $p_{t+1} = -r_{t+1} + \beta_{t+1} p_t$

CG (version 1)

Cost per iteration:

- matrix vector mult: $A p_t$
- $p_t^T \underbrace{(A p_t)}_{\text{already computed}}$
- $r^T r$

- not much more expensive than GD

Can we improve CG (in the quadratic setting $\phi(x) = \frac{1}{2}x^T A x - b^T x$?

$\uparrow$ PD, sym.

Idea : Preconditioning

First, some background :

* Recall that we seek x^* : $Ax^* = b$

$\uparrow$ sym. PD, $n \times n$

* It will turn out that the performance of CG depends on the eigenvalues of A.

How? Define $\|z\|_A = \sqrt{z^T A z}$

$z \in \mathbb{R}^n$

e.g. If $A = I$, then $\|z\|_A = \|z\|$
$= \sqrt{z^T z}$

Theorem: If A has eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$, then

$$\|x^{(t+1)} - x^*\|_A^2 \leq \left(\frac{\lambda_{n-t} - \lambda_1}{\lambda_{n-t} + \lambda_1}\right)^2 \|x^{(0)} - x^*\|_A^2$$

This implies, choosing $t+1 = n \Leftrightarrow t = n-1$

$$\|x^{(n)} - x^*\|_A^2 \leq 0 \cdot \|x^{(0)} - x^*\|_A^2 = 0$$

That is, convergence within n-steps!

Additionally, if for example $\lambda_{n-1} = \lambda_n$, then convergence to x^* takes only $n-1$ steps!