Problem:

we wish to compute

$$\underset{x}{\operatorname{argmin}} \; f(x) = \frac{x^T A x}{2} - b^T x$$

where A is Positive semi definite

1) this is a local model of some function

2) soln when $Ax = b$
(Newton, linear alg)

We assume that solving $Ax = b$ is hard (eg can't store A)

Conjugate directions

a set of dirns $p_0 \cdots p_N$ is

conjugate if

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

and $$p_i^T A p_i \neq 0$$ ~~Otherwise~~

Notice: It is quite straightforward to make conjugate dirns

$$p_0 = e_0$$

$$p_1 = e_1 - \frac{(e_1^T A p_0)}{(p_0^T A p_0)} p_0$$

$$p_i = e_i - \sum_{j<i} \frac{(e_i^T A p_j)}{(p_j^T A p_j)} p_j$$

Notice

- there are many sets of conj. directions (~~start~~ run alg. above w/ a different set of basis vectors)

- each set contains N c.d.'s if A is positive definite. You can get this by diagonalizing A

Conjugate directions simplify problem

write $x = \sum_i \alpha_i p_i$ ← conj. wrt A.

then $f(x) = \sum_i \alpha_i^2 p_i^T A p_i - \sum_i \alpha_i b^T p_i$

↑ no $\alpha_i \alpha_j$ terms

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

so we can minimize the directions independently.

Algorithm skeleton

$x_0 = 0$

→ (get conj dir p_i)

$x_{i+1} = x_i + \alpha_i p_i$

α_i ← chosen to minimize in p_i

- Need go round only N times because the p_i form a basis

$\text{span}\{p_0, \ldots p_N\} = \text{span}\{e_0 \ldots e_N\}$

the basis you chose to form C.D.s in.

α_i is <u>easy</u> to get

$$\underset{\alpha_i}{\operatorname{argmin}} \; f(x_i + \alpha_i p_i) = \underset{\alpha_i}{\operatorname{argmin}} \; x_i^T \frac{A}{2} x_i + \alpha_i x_i^T A p_i + \alpha_i^2 p_i^T \frac{A}{2} p_i - b^T x_i - \alpha_i b^T p_i$$

<u>BUT</u> $x_i = [\text{linear comb of } p_0 \cdots p_{i-1}]$

so $x_i^T A p_i = 0$

so we must minimize

$$\alpha_i^2 p_i^T \frac{A}{2} p_i - \alpha_i b^T p_i + [\text{stuff that doesn't depend on } \alpha_i]$$

so $$\alpha_i = \frac{b^T p_i}{p_i^T A p_i}$$

<u>Notice</u> if A is P.D., then $p_i^T A p_i > 0$

– this will come in useful.

Algorithm

$x_0 = 0$

(get new conj. dir)

$$\alpha_i = \frac{b^T p_i}{p_i^T A p_i}$$

$$x_{i+1} = x_i + \alpha_i p_i$$

This is the bad bit

- we currently need to store all old conj. directions
- we must fix this.

Write

$$r_i = Ax_i - b \quad \leftarrow \underline{\text{residual}}$$

Start at $x_0 = 0$

$$r_0 = -b$$

then assume we have $p_0 \dots p_k$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$\alpha_k = \frac{b^T p_k}{p_k^T A p_k}$$

Notice:

$$r_k^T p_k = x_k^T A p_k - b_k^T p_k$$

but $x_k \in \text{span}\{p_0 \dots p_{k-1}\}$

So $-r_k^T p_k = b_k^T p_k$

So $$\alpha_k = \frac{-r_k^T p_k}{p_k^T A p_k}$$

Now

$$r_{k+1} = Ax_{k+1} - b$$
$$= A(x_k + \alpha p_k) - b$$
$$= r_k + \alpha_k A p_k$$

So that:

$$r_{k+1}^T p_k = r_k^T p_k - \frac{r_k^T p_k}{p_k^T A p_k} \cdot (p_k^T A p_k)$$

$$= 0$$

i.e r_{k+1} is perp to p_k.

Which means we can cook up a new ~~conj~~ direction out of r_{k+1} and p_k

and make it conj. to p_k

our new dirn

$$p_{k+1} = -r_{k+1} + \beta_k p_k$$

now, we want

$$p_{k+1}^T A p_k = 0$$

so $$r_{k+1}^T A p_k = \beta_k p_k^T A p_k$$

i.e

$$\beta_k = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

This direction is special in important ways

IF we start with

$$p_0 = -r_0$$

~~It turns~~

I have already shown that this construction yields

$$p_{k+1}^T A p_k = 0$$

But we can also show that

$$p_{k+1}^T A p_j = 0 \qquad 0 \le j \le k-1.$$

We do this by showing

(1) that

$$r_{k+1}^T p_j = 0 \quad , \; 0 \le j \le k.$$

(2)

$$\text{span}\{r_0, r_1, r_2 \cdots r_k\}$$

$$\text{span}\{r_0, Ap_0, Ap_1, \cdots Ap_{k-1}\}$$

$$\text{Span}\{p_0, p_1, p_2, \cdots p_k\}.$$

Now assume (1) and (2)

then

$$p_{K+1}^T A p_j = \begin{cases} 0 & \text{for } j = K \\ -r_{K+1}^T A p_j + \beta_K \left(p_K^T A p_j \right) & \\ & j < K. \end{cases}$$

Now we apply an induction.

(1) we have $p_2^T A p_1 = 0$

(2) assume $p_K^T A p_j = 0$, $j < K-1$

(we're OK for $j = K-1$)

(3) then

$$p_{K+1}^T A p_j = -r_{K+1}^T A p_j$$

But $r_{K+1}^T p_j = 0$ ← by (1)

so $r_{K+1} \perp \text{Span}\{p_0, \ldots p_K\}$

↑ at right angles to

But

$$\text{span}\{p_0, \cdots p_k\}$$

$$= \text{span}\{r_0, Ap_0, \cdots Ap_{k-1}\}$$

So $r_{k+1} \perp \text{span}\{r_0, Ap_0, \cdots Ap_{k-1}\}$

So $r_{k+1}^T A p_j = 0 \quad , \quad j \leq k-1$

So

$p_{k+1} A p_j = 0$ and we are done

Must now prove ① and ②

(2) $r_{k+1}^T p_k = 0$ Already established this

must show

$$r_{k+1}^T p_j = 0 \quad , \quad j < k.$$

Notice

$$r_{k+1} = r_k + \alpha_k A_k p_k$$

So $r_{k+1} = r_{j+1} + [\text{a sum of } \alpha A p \text{ terms for } j+1 \cdots k]$

So $r_{k+1}^T p_j = r_{j+1}^T p_j + [\quad]^T p_j$

$[\quad] = 0$ by conjugacy

$r_{j+1}^T p_j = 0$ from above

So we are done on (1)

We must deal with claim 2.

$$V_k^p = \text{Span}\{p_0, \cdots p_k\}$$

$$V_k^r = \text{Span}\{r_0, \quad , r_k\}$$

$$V_k^A = \text{Span}\{r_0, Ap_0, \cdots Ap_{k-1}\}$$

Example

$$V_0^p = \text{Span}\{p_0\}; \quad V_0^r = \text{Span}\{r_0\}; \quad V_0^A = \text{Span}\{r_0\}$$

but $p_0 = -r_0$

$$V_1^p = \text{Span}\{p_0, p_1\} = \text{Span}\{p_0, r_1 + \beta_1 p_0\}$$

$$V_1^r = \text{Span}\{r_0, r_1\} = \text{span}\{r_0, r_0 + \alpha_0 Ap_0\}$$

$$V_1^A = \text{Span}\{r_0, Ap_0\} =$$

Now $V_1^r = V_1^p$ by inspection (Notice p has nothing to do with it)

$V_1^r = V_1^A$ by inspection

AND THAT IS THE FORM OF THE INDUCTION

We have

start

$$x_0 = 0$$
$$r_0 = -b$$
$$p_0 = -r_0$$

..

$$\alpha_k = \frac{-r_k^T p_k}{p_k^T A p_k}$$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$r_{k+1} = r_k + \alpha_k A p_k$$

$$\beta_k = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

$$p_{k+1} = -r_{k+1} + \beta_k p_k$$

Terminating if

- r_{k+1} is small "enough"
- of kN iterations

A little notation

we computed β_k as

$$\beta_k = \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$$

But we could use other expressions we require

$$p_{k+1}^T A p_k = 0$$

Notice: $r_{k+1} - r_k = \alpha A p_k$

so we could require

$$(r_{k+1} - r_k)^T p_{k+1} = 0$$

This gets us

~~$\beta_k = (r_{k+1} - r_k)$~~

$$(r_{k+1} - r_k)^T p_{k+1} = -(r_{k+1} - r_k)^T r_k + \beta_k (r_{k+1} - r_k)^T p_k$$

But

$$r_{k+1}^T p_k = 0$$

AND

$$r_k^T p_k = r_k^T (-r_k + \beta_{k-1} p_{k-1})$$
$$= -r_k^T r_k$$

So that:

$$\beta_k = \frac{(r_{k+1} - r_k)^T r_k}{r_k^T r_k}$$

This comes in useful in non-linear cases

This gives

Start:

$$x_0 = 0$$
$$r_0 = -b$$
$$p_0 = -r_0$$

Iterate:

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$$

$$x_{k+1} = x_k + \alpha_k p_k$$

$$r_{k+1} = r_k + \alpha_k A p_k$$

$$\beta_k = \frac{(r_{k+1} - r_k)^T r_k}{r_k^T r_k}$$

$$p_{k+1} = -r_{k+1} + \beta_k p_k$$

Terminate if:

- r_{k+1} is small enough.
- N iterations

Modified Newton w/ C.G.

Notice: every successful step of CG causes $x^T \frac{A}{2} x - b^T x$ to decrease

Pf:
- if A isn't p.d., we will know because $p_k^T A p_k$ will be 0/neg.

Strategy:
- Think of local model of fn.

$$f(u_i + \delta) \approx f(u_i) + \delta^T \frac{H_f}{2} \delta + \nabla f^T \delta$$

u_i: our original point

- we are looking for a δ that (ideally) minimizes $\delta^T \frac{H}{2} \delta + \nabla f^T \delta$
- but if $\delta^T \frac{H}{2} \delta + \nabla f^T \delta < 0$, δ is still a useful step. (descent).

- apply C.G., checking sign of $p_k^T A p_k$ at each step.

- Terminate if:
  - N rounds
  - r_k "small"
  - $p_k^T A p_k$ too small

- This yields a descent dir. AND its a Newton dir if H is pd

BUT NOTICE C.G.'s best feature

- You don't need to know or store A
- You just need to be able to form matrix-vector products.

We originally worked with

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

But imagine we have a function $g(x)$ that is <u>modelled</u> as

$$g(x) \approx g_0 + \alpha \cdot \nabla g|_0^T x + \frac{x^T H x}{2}$$

We can build a <u>powerful</u> analogy

f side	g side
A	H
b	$-\nabla g|_0$
$r_k = Ax + b$	$\nabla g \approx \nabla g|_0 + Hx$
α obtained by exact min	α – linesearch

This is not perfect, because as x changes, H will change
- we cannot expect to take N steps
- we may need to restart

This leads to 3 algorithms
(Polak-Ribiere; Fletcher-Reeves; ?)

I describe only Polak-Ribiere.

- Start at x_0

$$r_0 = +\nabla g\Big|_{x=x_0}$$
$$p_0 = -r_0$$

$$\alpha_k = \text{linesearch}\left(g(x_k + \alpha p_k)\right)$$
$$x_{k+1} = x_k + \alpha p_k$$
$$r_{k+1} = \nabla g\Big|_{x=x_{k+1}}$$
$$\beta_k = \frac{(r_{k+1} - r_k)^T r_k}{r_k^T r_k}$$
$$p_{k+1} = -r_{k+1} + \beta_k p_k$$

But we may need to reset

options

- every Nth iteration, restart (i.e. $x_0 \leftarrow$ current x_{k+1} and go again)
- OR if $\beta_k <$ small threshold ~~$\beta_k = \min$~~ $\beta_k \leftarrow 0$

(small threshold is often 0.2)

NOTICE:

- Method gets second order info, BUT we do not need to compute or store H!