Minimization Algorithms

Solving linear system $Ax=b$ by minimizing a cost function $J(x) = \frac{1}{2} x^T A x - b^T x + c$

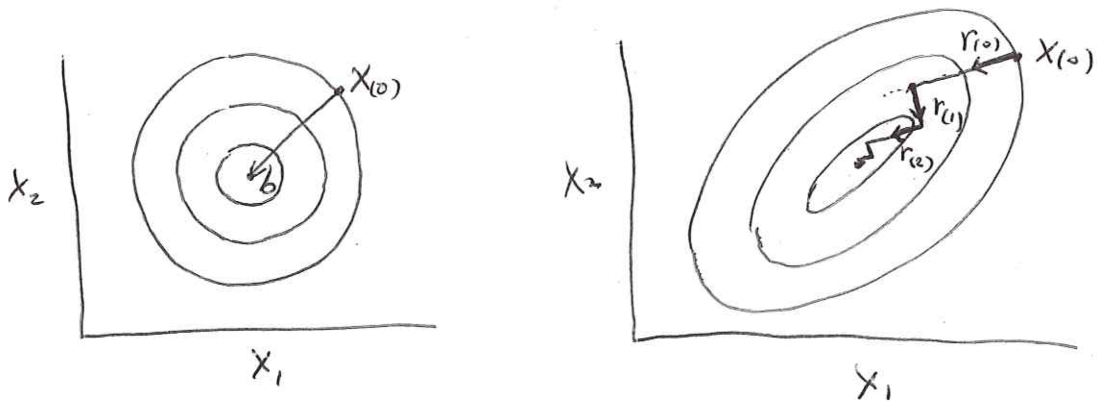
$\nabla J = Ax - b = 0$ is the gradient vector

$\nabla^2 J = A$ is the Hessian matrix that controls the shape of the cost function, "valley"

An example in two-variable case: $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

If $A = I$, $x = b$. $J = \frac{1}{2}(x_1^2 + x_2^2) - (b_1 x_1 + b_2 x_2) + c$

Contour of J in 2D:



If A has off-diagonal terms, the shape of contours becomes enlongated, (A is symmetric, positive-definite)

Start from a first guess $x_{(0)}$, several iterative methods can solve $Ax=b$ by stepping towards the solution

1 Gradient (steepest) Descent

Move along $-\nabla J$ to line minimum of J, $r_{(0)} = b - Ax_{(0)}$

for $i = 0, 1, 2, \cdots$ until $\|r_{(i)}\|$ is small enough:

$$r_{(i)} = -\nabla J(x_{(i)}) = b - Ax_{(i)}$$

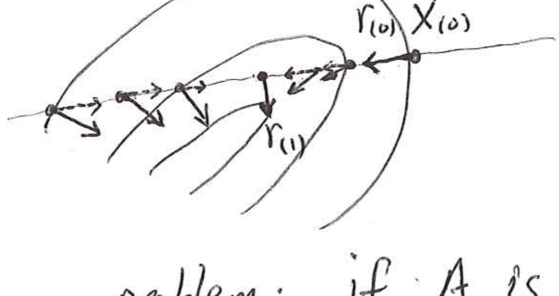
$$x_{(i+1)} = x_{(i)} + \alpha r_{(i)}, \quad \alpha = \frac{r_{(i)}^T r_{(i)}}{r_{(i)}^T A r_{(i)}}$$

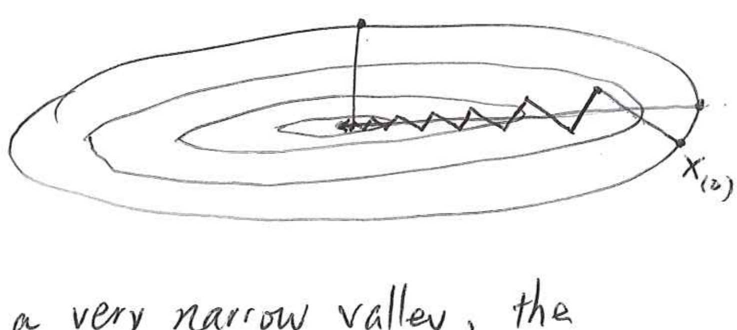
end

In each iteration (i), α is chosen so that $r_{(i)}$ is orthogonal to the next direction $r_{(i+1)}$:

$$\begin{aligned} r_{(i+1)}^T r_{(i)} &= (b - Ax_{(i+1)})^T r_{(i)} \\ &= (b - Ax_{(i)} - \alpha A r_{(i)})^T r_{(i)} \\ &= r_{(i)}^T r_{(i)} - \alpha r_{(i)}^T A r_{(i)} = 0 \end{aligned}$$

the point at which $r_{(i)} \perp r_{(i+1)}$ is where $J(x_{(i)} + \alpha r_{(i)})$ is minimum:

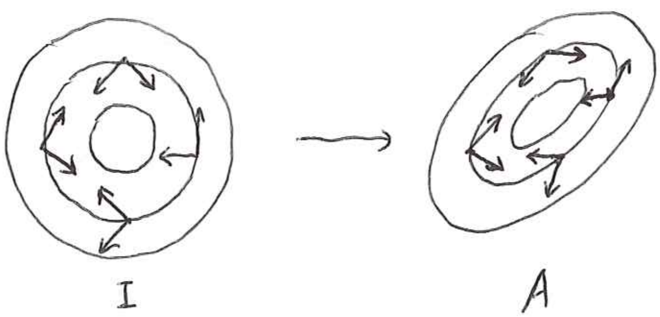




problem: if A is a very narrow valley, the method may need a lot of zigzags when initial point is not chosen properly.

2. Conjugate Gradient (CG)

- most popular solver. For n-dimensional problem, ($x \in \mathbb{R}^{n \times 1}$), it only requires n iterations.
- Two vectors x and y are "conjugate", or A-orthogonal, if $x^T A y = 0$.
- x and y are orthogonal in another space before transformed into the current space.



Algorithm: $r_{(0)} = b - Ax_{(0)}$, $p_{(0)} = r_{(0)}$

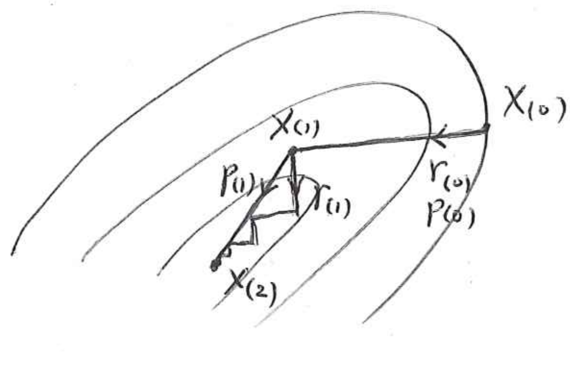
for $i = 0, 1, 2, \ldots$ until $\|r_{(i)}\|$ is small enough:

$$x_{(i+1)} = x_{(i)} + \alpha p_{(i)}, \quad \alpha = \frac{r_{(i)}^T r_{(i)}}{p_{(i)}^T A p_{(i)}}$$

$$r_{(i+1)} = r_{(i)} - \alpha A p_{(i)}$$

$$p_{(i+1)} = r_{(i+1)} + \beta p_{(i)}, \quad \beta = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$$

end



In each iteration (i), α is chosen so that $r_{(i)}$ is orthogonal to $r_{(i+1)}$ $\Rightarrow$ reach line minimum of J

β is chosen so that $p_{(i+1)}$ is conjugate with $p_{(i)}$.

$$r_{(i+1)}^T r_{(i)} = \left(r_{(i)} - \alpha A p_{(i)}\right)^T r_{(i)}$$

$$= r_{(i)}^T r_{(i)} - \alpha\, p_{(i)}^T A r_{(i)} = 0$$

$$\alpha = \frac{r_{(i)}^T r_{(i)}}{p_{(i)}^T A r_{(i)}} = \frac{r_{(i)}^T r_{(i)}}{p_{(i)}^T A p_{(i)}}$$

since $p_{(i)} = r_{(i)} + \beta p_{(i-1)}$

$$p_{(i+1)}^T A p_{(i)} = \left(r_{(i+1)} + \beta p_{(i)}\right)^T A p_{(i)}$$

$$= r_{(i+1)}^T A p_{(i)} + \beta \underbrace{p_{(i)}^T A p_{(i)}}_{= r_{(i)}^T A p_{(i)}} = 0$$

replace $A p_{(i)} = \frac{1}{\alpha}\left(r_{(i)} - r_{(i+1)}\right)$

$$\beta = -\frac{r_{(i+1)}^T \left(r_{(i)} - r_{(i+1)}\right)/\alpha}{r_{(i)}^T \left(r_{(i)} - r_{(i+1)}\right)/\alpha} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}},$$

3. Other methods

- Newton's method, for each iteration $\nabla^2 J \delta x_{(i)} = -\nabla J(x_{(i)})$

$$x_{(i+1)} = x_{(i)} + \delta x_{(i)}$$

- quasi-Newton methods. (e.g. BFGS)

  instead of calculating inverse of $\nabla^2 J$, approximate it with something easy to inverse, update the approx. $\nabla^2 J$ iteratively.

  (matlab: fminunc)