Conjugate Gradient Method

Marco Zanotti

University Milano-Bicocca

Contents

1. Line Search - Review

Problem

$$min\ f(x) = \frac{1}{2}x^T Ax - b^T x \quad (1)$$

where A is an $n \times n$ symmetric and positive definite matrix, that is f is a convex quadratic function.

Solving w.r.t x implies

$$\nabla f(x) = Ax - b = 0 \implies Ax = b$$

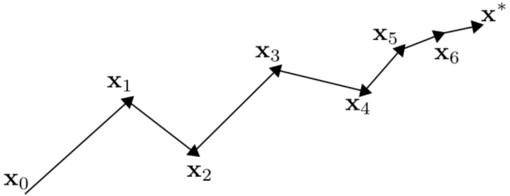
hence, at point $x = x_k$

$$\nabla f(x_k) = Ax_k - b = 0 \implies Ax_k = b$$

Line Search

Each iteration is given by

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



where α_k is the step length and p_k is the direction.

Line Search

The direction often has the form

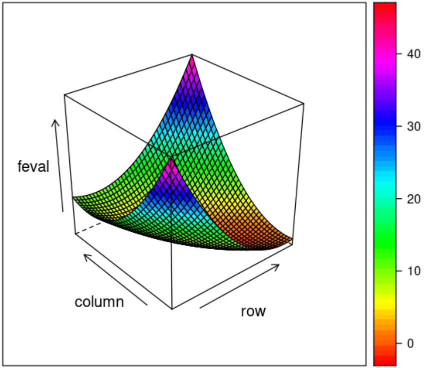
$$p_k = -B_k^{-1} \nabla f_k$$

where B_k is a symmetric, nonsingular matrix.

In the Steepest Descent $B_k = I$.
In the Newton's method $B_k = \nabla^2 f_k$.

Steepest Descent - Simulation

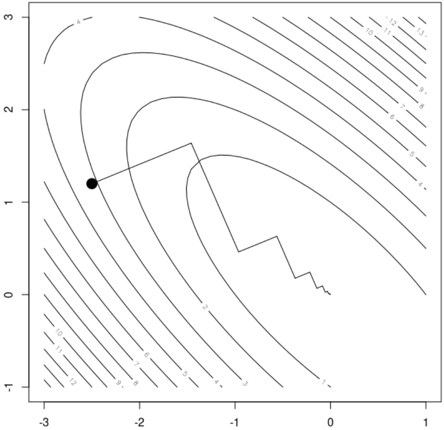
$$f(x_1, x_2) = x_1^2 + x_2^2 + \frac{3}{2}x_1 x_2 = \frac{1}{2}x^T \begin{bmatrix} 2 & \frac{3}{2} \\ \frac{3}{2} & 2 \end{bmatrix} x - \begin{bmatrix} 0 \\ 0 \end{bmatrix} x$$



Steepest Descent - Simulation

$$f(x_1, x_2) = x_1^2 + x_2^2 + \frac{3}{2}x_1 x_2$$

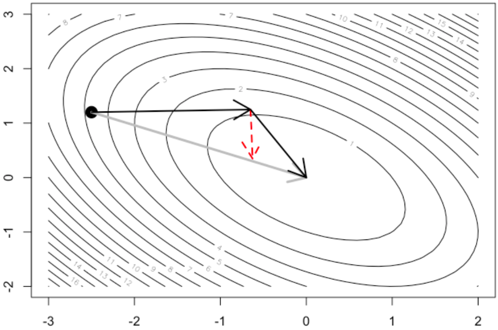
- SD is inefficient and slow to converge since it often requires many iterations to reach the optimum.
- 15 iterations to reach a tolerance of 0.01



2. Conjugate Direction

Intuition

$$f(x_1, x_2) = x_1^2 + x_2^2 + \frac{3}{2}x_1x_2$$



The red dashed arrow is the SD direction at the second step.

Definition: Conjugate Vectors

A set of nonzero vectors $\{p_0, p_1, ..., p_n\}$ is said to be **conjugate** (or A-orthogonal) with respect to a symmetric positive definite matrix A if and only if

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

Moreover, any set of vectors satisfying the **conjugacy** property is also **linearly independent**.

Why is conjugacy relevant?

It is possible to solve (1) in exactly n steps by successively minimizing it along the individual **conjugate** directions.

Theorem

Let the following be a (simple) **conjugate** direction method: given a starting point $x_0 \in R^n$ and a set of **conjugate** directions $\{p_0, p_1, ..., p_{n-1}\}$, at each iteration k a point is chosen such that

$$x_{k+1} = x_k + \alpha_k p_k \quad (2)$$

where α_k is the step length and p_k is the **conjugate** direction.

The method converges to the solution x^* of (1) in at most n steps.

Proof

First, α_k is the one-dimensional minimizer of (1) along $x_k + \alpha_k p_k$ and can be computed explicitly by

$$\nabla f_\alpha(x_k + \alpha_k p_k) = (x_k + \alpha_k p_k)^T A p_k - b^T p_k$$

setting equal to 0 and solving for α

$$(x_k + \alpha_k p_k)^T A p_k - b^T p_k = 0$$

$$\alpha_k = \frac{(b^T - Ax_k)p_k}{p_k^T A p_k}$$

$$\alpha_k = \frac{-\nabla f(x_k) p_k}{p_k^T A p_k} \quad (3)$$

Proof - Continue

It can be observed that, since the directions $\{p_i\}$ are linearly independent, they form a basis on R^n, implying they span the whole space.

Hence, the solution x^* can be represented as

$$x^* = x_0 + \delta_0 p_0 + \delta_1 p_1 + ... + \delta_{n-1} p_{n-1}$$

Proof - Continue

For some choice of the scalars δ_k and premultiplying by $p_k^T A$

$$p_k^T A(x^* - x_0) = p_k^T A(\delta_0 p_0 + \delta_1 p_1 + ... + \delta_k p_k)$$

and using the conjugacy property $p_i^T A p_j = 0$

$$p_k^T A(x^* - x_0) = p_k^T A \delta_k p_k$$

$$\delta_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} \quad (4)$$

Proof - Continue

Now, suppose that x_k is generate by (2) and (3), then

$$x_k = x_0 + \alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{k-1} p_{k-1}$$

premultiplying by $p_k^T A$ and using the conjugacy $p_i^T A p_j = 0$

$$p_k^T A(x_k - x_0) = p_k^T A(\alpha_0 p_0 + \alpha_1 p_1 + \ldots + \alpha_{k-1} p_{k-1})$$

$$p_k^T A(x_k - x_0) = 0$$

if this holds true for x_k it must hold also for x^*, hence

$$p_k^T A(x^* - x_0) = p_k^T A(x^* - x_k) = p_k^T (b - A x_k) = -p_k^T \nabla f(x_k)$$

Proof - End

Now, using the fact that $p_k^T A(x^* - x_0) = -p_k^T \nabla f(x_k)$

$$(4) \quad \delta_k = \frac{p_k^T A(x^* - x_0)}{p_k^T A p_k} = \frac{-p_k^T \nabla f(x_k)}{p_k^T A p_k} = \alpha_k \quad (3)$$

The coefficients δ_k coincide with the step lengths α_k, proving the theorem.

3. Conjugate Gradient

How to find the conjugate directions?

To use (2)-(3), it remains to find n A-orthogonal vectors p_k.

One way, use

$$\{v : Av = \lambda v\}$$

the set of eigenvectors of A.

These are mutually orthogonal as well as conjugate with respect to A and could be used as the conjugate directions $\{p_0, p_1, ..., p_{n-1}\}$.

In general to find the eigenvectors of a matrix is inefficient since it requires an excessive amount of computations.

Conjugate Gradient - Basic Property

The Conjugate Gradient method is a conjugate direction method with a very special property:

In generating the set of conjugate directions, it can compute a new direction p_k by using only the previous direction p_{k-1}.

It does not need to know all the previous elements $p_0, p_1, \dots, p_{k-1}$ of the conjugate set, since p_k is automatically conjugate to all the previous directions.

Find p_k

In the basic CG method, each direction p_k is chosen to be a linear combination of the SD direction and the previous direction p_{k-1}.

$$p_k = -\nabla f(x_k) + \beta_k p_{k-1} \quad (5)$$

where the scalar β_k is derived from (5) imposing $p_i^T A p_j = 0$

$$p_{k-1}^T A p_k = p_{k-1}^T A(-\nabla f(x_k) + \beta_k p_{k-1})$$

$$0 = p_{k-1}^T A(-\nabla f(x_k) + \beta_k p_{k-1})$$

$$\beta_k = \frac{p_{k-1}^T A \nabla f(x_k)}{p_{k-1}^T A p_{k-1}} \quad (6)$$

Algorithm

Given x_0, set $\nabla f(x_0) = Ax_0 - b$, $p_0 = -\nabla f(x_0)$, $k = 0$

while $\nabla f(x_k) \neq 0$

$$\alpha_k = -\frac{\nabla f(x_k)^T p_k}{p_k^T A p_k} \quad (3)$$

$$x_{k+1} = x_k + \alpha_k p_k \quad (2)$$

$$\nabla f(x_{k+1}) = Ax_{k+1} - b$$

$$\beta_{k+1} = \frac{\nabla f(x_{k+1})^T A p_k}{p_k^T A p_k} \quad (6)$$

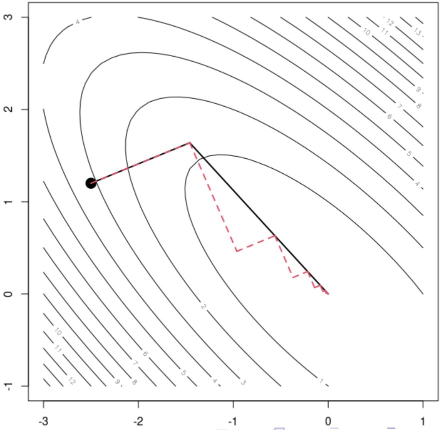
$$p_{k+1} = -\nabla f(x_{k+1}) + \beta_{k+1} p_k \quad (5)$$

$$k = k + 1$$

Conjugate Gradient - Simulation

$$f(x_1, x_2) = x_1^2 + x_2^2 + \frac{3}{2}x_1 x_2$$

- CG is much more efficient than SD and it takes at most n iterations to reach the optimum.
- 2 iterations to reach a tolerance of 0.01



Conclusions

- CG is more efficient than SD since it reaches the optimum in at most n iterations
- It is suitable especially for large scale optimization problems since it requires minimum storage and computation
- The method is sensitive to its starting position
- The method works with quadratic or quadratic-like functions, or where the function is approximately quadratic near the optimum

CG method has been improved and adapted to minimize general convex functions and even general nonlinear functions.

4. Application: Linear Regression

Dataset: GapMinder

lifeExp	Intercept	pop	gdpPercap	Asia	Europe	Americas
28.80	1	-0.20	-0.65	1	0	0
30.33	1	-0.19	-0.65	1	0	0
32.00	1	-0.18	-0.65	1	0	0
34.02	1	-0.17	-0.65	1	0	0
36.09	1	-0.16	-0.66	1	0	0
38.44	1	-0.14	-0.65	1	0	0

CG algorithm implementation

```r
beta_vec <- matrix(0, nrow = ncol(x_mat), ncol = 1) # initial guess
A <- t(x_mat) %*% x_mat # derive matrix A
b <- t(x_mat) %*% y_vec # derive vector b
fg <- A %*% beta_vec - b # residuals = gradient of f
p <- -fg # search direction
k <- 0 # number of iterations

while (norm(fg, "2") > 0.01) {

  alpha <- as.numeric((t(fg) %*% fg) / (t(p) %*% A %*% p))
  beta_vec <- beta_vec + alpha * p # update beta coefficients
  fg1 <- fg + alpha * A %*% p # update gradient of f
  beta <- as.numeric((t(fg1) %*% fg1) / (t(fg) %*% fg))
  p1 <- -fg1 + beta * p # update search direction (p)
  fg <- fg1
  p <- p1
  k <- k + 1
}
```

CG results

	beta_CG	beta_lm
Intercept	51.25188	51.25188
pop	0.69744	0.69744
gdpPercap	4.43098	4.43098
Asia	8.19263	8.19263
Europe	17.47269	17.47269
Americas	13.47594	13.47594
Oceania	18.08330	18.08330

method	median	mem_alloc	n_itr
LM	785us	752KB	971
GC	256us	192KB	990

References

J. Nocedal and S. Wright, Numerical Optimization, 2006, Springer

Thank you!