An Introduction Of Conjugate Gradient Method

Saiyue Lyu

Conjugate gradient method is an algorithm to solve the linear system equation $Ax = b$ for a positive definite matrix A. This report summarizes the motivation, theoretical analysis, detailed algorithm and the implementation of conjugate gradient method based on CO367 I took in Fall 2018 at UWaterloo.

1. Motivation

Definition 1.1. *A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is said to be positive definite if*

$$h^T Ah > 0, \forall h \in \mathbb{R}^n\backslash\{0\}.$$

It is easy to see that a square real matrix A is positive definite if and only if its eigenvalues are all positive.

For any positive definite matrix $A \in \mathbb{R}^{n\times n}$ and any vector $b \in \mathbb{R}^n$, we want to find the solution $x \in \mathbb{R}^n$ for the system of linear equation $Ax = b$.

One of the methods to solve this system is to use Gauss-Jordan Elimination. When A is invertible, the system has one unique solution $x = A^{-1}b$ and when A is not invertible, the system has no solution or has a solution set. In both cases Gauss-Jordan Elimination requires a complexity of $\mathcal{O}(n^3)$ and the matrix multiplication $A^{-1}b$ has arithmetic complexity of $\mathcal{O}(n^2)$, so in total, solving this system requires a complexity of $\mathcal{O}(n^3)$. Unless this matrix A is well-structured, this method will be computationally expensive. Due to this difficulty, we want to transfer this question to a more solvable problem instead of solving this linear system directly.

Note that if A is positive definite, then all of its eigenvalues are positive and 0 is not an eigenvalue of A, this means the system $Ax = 0$ has no non-trivial solution, i.e. A is invertible. So under the assumption that A is positive definite, we want to find the unique solution of $Ax = b$ but not computing $A^{-1}b$ directly.

2. Optimization Conditions

Consider the optimization problem with the quadratic objective function

$$\min_x \; h(x) = \frac{1}{2}x^T Ax - b^T x + c,$$

where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$. I will show next that if A is positive definite, then solving this optimization problem is equivalent to solving the system $Ax = b$.

First we introduce the basic definitions and results in optimization :

Definition 2.1. *We say that x^* is a local minimizer of f if*

$$\exists\,\delta > 0 \textit{ such that } f(x^*) \le f(x), \forall x \in B_\delta(x^*).$$

We say that x^* is a strict local minimizer of f if

$$\exists\,\delta > 0 \textit{ such that } f(x^*) < f(x), \forall x \in (B_\delta(x^*)\backslash x^*).$$

We say that x^* is a global minimizer if

$$f(x^*) \le f(x), \forall x \in \mathbb{R}^n.$$

We say that x^* is a critical or stationary point if

$$\nabla f(x) = 0.$$

Note that all local minimizers are critical points, but not all critical points are local minimizers.

Theorem 2.1 (First Order Necessary Conditions For Optimality). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be C^1-smooth. If x^* is a local minimizer, then $\nabla f(x^*) = 0$.*

Proof. Let $B_\delta(x^*)$ be such that $f(x^*) \le f(x), \forall x \in B_\delta(x^*)$, i.e.

$$\forall i, \forall |k| < \delta, f(x^* + k \cdot e_i) - f(x^*) \ge 0,$$

$$\text{which gives} \begin{cases} \frac{f(x^*+k\cdot e_i)-f(x^*)}{k} \ge 0 \text{ if } k > 0 \\ \frac{f(x^*+k\cdot e_i)-f(x^*)}{k} \le 0 \text{ if } k < 0. \end{cases}$$

Since $f \in C^1$, then $\lim_{k\to 0} \frac{f(x^*+k\cdot e_i)}{k}$ exists.
If both $\ge 0, \le 0$ inequalities hold, then the equality holds, thus $\frac{\partial f}{\partial x_i}(x^*) = 0, \forall i$.
Therefore $\nabla f(x^*) = 0$. ∎

Theorem 2.2 (Second Order Necessary Conditions For Local Optimality). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be C^2-smooth. If x^* is a local minimizer, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi-definite.*

Proof. Let $z \in \mathbb{R}^n\backslash\{0\}$, we need to prove $z^T\nabla^2 f(x^*)z \ge 0$.
Let $B_\delta(x^*)$ be such that $f(x^*) \le f(x), \forall x \in B_\delta(x^*)$.
Let $y := k \cdot \frac{z}{||z||}$ with $0 < k < \delta$, then we have :

$$f(x^* + y) - f(x^*) \ge 0$$

$$f(x^*) + y^T\nabla f(x^*) + \frac{1}{2}y^T\nabla^2 f(x^*)y + \phi(x) - f(x^*) \ge 0 \text{ where } \lim_{y\to 0, y\neq 0} \frac{\phi(y)}{||y||} = 0.$$

By 1st order condition, we have $y^T\nabla f(x^*) = 0$, hence we have :

$$\frac{1}{2}\frac{k^2}{||z||^2}z^T\nabla^2 f(x^*)z + \phi(h\frac{z}{||z||}) \ge 0$$

$$z^T\nabla^2 f(x^*)z + 2||z||^2\frac{1}{k^2}\phi(k\frac{z}{||z||}) \ge 0.$$

Take the limit when $h \to 0$, by Talor's theorem, we have :

$$\lim_{k\to 0, k\neq 0} \frac{\phi(k \cdot \frac{z}{||z||})}{k^2} = 0.$$

Therefore we have $z^T\nabla^2 f(x^*)z \ge 0$. ∎

Theorem 2.3 (Second Order Sufficient Conditions For Local Optimality). *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^2\big(B_\delta(x^*)\big), x^* \in \mathbb{R}, \delta > 0$. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local minimizer.*

Proof. By Talor's 2nd order equation, $\forall y \in B_\delta(x^*)$, have :

$$f(x^* + y) = f(x^*) + y^T \nabla f(x^*) + \tfrac{1}{2} y^T \nabla^2 f(x^*) y + \phi(x) \text{ where } \lim_{y \to 0, y \neq 0} \tfrac{\phi(y)}{||y||} = 0.$$

Let $0 < \lambda_1 < \cdots < \lambda_n$ be the positive eigenvalues of $\nabla^2 f(x^*)$.
By the definition of limit, have

$$\exists r > 0 : \forall y \in B_r(x^*), |\frac{\phi(x)}{||y||^2}| \leq \frac{\lambda_1}{4} \iff |\phi(x)| \leq ||y||^2 \frac{\lambda_1}{4}.$$

Note that,

$$||y||^2 \cdot \lambda_1 \leq y^T \nabla^2 f(x^*) y \leq ||y||^2 \cdot \lambda_n.$$

Also by assumption, $\nabla f(x^*) = 0$, then we have :

$$\begin{aligned} f(x^* + y) &= f(x^*) + \frac{1}{2} y^T \nabla^2 f(x^*) y + \phi(y) \\ &\geq f(x^*) + \frac{1}{2} ||y||^2 \lambda_1 - ||y||^2 \frac{\lambda_1}{4} \\ &= f(x^*) + \frac{1}{4} ||y||^2 \cdot \lambda_1 \\ &> f(x^*) \text{ for all } y \in B_r(x^*) \backslash \{0\}. \end{aligned}$$

Therefore x^* is a strict local minimizer over $B_r(x^*)$. ∎

With above theorems, we obtain the following summary for optimality conditions :

$$\begin{cases} \nabla f(x^*) = 0 \\ \nabla^2 f(x^*) \; positive \; definite \end{cases} \Rightarrow x^* \text{ is a strict local minimizer}$$
$$\Rightarrow x^* \text{ is a local minimizer} \Rightarrow \begin{cases} \nabla f(x^*) = 0 \\ \nabla^2 f(x^*) \; positive \; semi-definite \end{cases}.$$

Lemma 2.4. *The gradient of $h(x)$ is $\nabla h(x) = Ax - b$ and the hessian of $h(x)$ is $\nabla^2 h(x) = A$.*

Proof.

$$
\begin{aligned}
\frac{\partial}{\partial x_k} b^T x &= b_k \\
\nabla b^T x &= b \\
\frac{\partial}{\partial x_k} x^T A x &= \frac{\partial}{\partial x_k} \sum_{i,j} A_{ij} x_i x_j \\
&= \frac{\partial}{\partial x_k} \Big(\sum_{j \neq k} A_{kj} x_k x_j + \sum_{i \neq k} A_{ik} x_i x_k + A_{kk} x_k^2 \Big) \\
\text{(as } A \text{ is symmetric)} \qquad &= \frac{\partial}{\partial x_k} \Big(\sum_{j \neq k} A_{kj} x_k x_j + \sum_{i \neq k} A_{ki} x_i x_k + A_{kk} x_k^2 \Big) \\
&= \frac{\partial}{\partial x_k} \Big(2 \sum_{j \neq k} A_{kj} x_k x_j + A_{kk} x_k^2 \Big) \\
&= 2 \sum_{j \neq k} A_{kj} x_j + 2 A_{kk} x_k = 2 \sum_{j} A_{kj} x_j \\
&= k\text{th row of } 2Ax \\
\nabla x^T A x &= 2Ax
\end{aligned}
$$

Similarly, we can get $\nabla^2 h(x) = \nabla(Ax - b) = A$.
Therefore $\nabla h(x) = Ax - b$ and $\nabla^2 h(x) = A$. ∎

Theorem 2.5. *The quadratic function $h(x) = \frac{1}{2} x^T A x - b^T x + c$ with positive definite A has a unique global minimizer.*

Proof. Since A is positive definite, then A^{-1} exists. There is a unique critical point (i.e. point where $\nabla h = 0$) $x^* = A^{-1} b$. Since $\nabla^2 h(x^*) = A$ is positive definite, then x^* is a local minimizer. Note that for any $y \in \mathbb{R}^n$, have:

$$
{x^*}^T A y = ({x^*}^T A y)^T = y^T A^T x^* = y^T A x^*. \tag{1}
$$

Hence have:

$$
\begin{aligned}
h(x^* + y) &= \frac{1}{2}(x^* + y)^T A (x^* + y) - b^T (x^* + y) + c \\
&= \frac{1}{2} {x^*}^T A x^* + \frac{1}{2} {x^*}^T A y + \frac{1}{2} y^T A x^* + \frac{1}{2} y^T A h - b^T x^* - b^T y + c \\
&= \Big(\frac{1}{2} {x^*}^T A x^* - b^T x^* + c\Big) + \Big(\frac{1}{2} {x^*}^T A y + \frac{1}{2} y^T A x^*\Big) + \frac{1}{2} y^T A y - b^T y \\
&= h(x^*) + y^T A x^* + \frac{1}{2} y^T A y - b^T y \text{ by (1)} \\
&= q(x^*) + y^T A (A^{-1} b) + \frac{1}{2} y^T A y - b^T y \\
&= q(x^*) + y^T b + \frac{1}{2} y^T A y - b^T y \\
&= q(x^*) + \frac{1}{2} y^T A y \\
&\geq q(x^*).
\end{aligned}
$$

Therefore x^* is a global minimizer of $h(x)$. ∎

Then using the summary for optimality conditions, we can derive the following theorem:

Theorem 2.6. *The solution of the optimization problem is exactly the solution of* $Ax = b$, *i.e.* x^* *is the global minimizer of* $\min_x h(x)$ *if and only if* x^* *is the solution of* $Ax = b$.

Proof. x^* is the solution of $Ax = b$.
$\Leftrightarrow \nabla h(x^*) = 0$.
$\Leftrightarrow x^*$ is a local minimizer (since $\nabla^2 h(x)$ is positive definite).
$\Leftrightarrow x^*$ is the unique global minimizer (by theorem(2.5)). ∎

Therefore solving $Ax = b$ can be turned into solving the optimization problem $\min_x h(x)$, which leads to the well-known **Conjugate Gradient Method**.

3. Direction and Step Size of Conjugate Gradient Method

To find a feasible direction p_i and moves the iterate x_i along the direction p_i to make the value of objective function to decrease, we have the following algorithm:

input : x_0
for $i = 0, 1, \cdots,$ **do**
  Choose direction p_i;
  Choose step size $\alpha_i > 0$;
  $x_{i+1} = x_i + \alpha_i \cdot p_i$
end

Algorithm 1: Algorithm for Feasible Direction

Direction p_i determines the direction that the iterate x_i should move along and step size α_i determines how far it moves away from the current x_i, to select p_i, we use the 1st order Taylor expansion:

$$h(x_{i+1}) = h(x_i - \alpha_i p_i) = h(x_i) - \alpha_i p_i^T \nabla h(x_i) + \phi(\alpha_i p_i) \quad \text{with} \quad \lim_{\alpha_i \to 0} \frac{\phi(\alpha_i p_i)}{||\alpha_i p_i||} = 0.$$

What we want is $h(x_{i+1}) < h(x_i)$, i.e. $\alpha_i p_i^T \nabla h(x_i) < 0$ and $\phi(\alpha_i p_i)$ is small enough, hence this forces $p_i^T \nabla h(x_i) < 0$ and α_i being small, to choose the direction p_i, we use the **Steepest Descent Method**, staring from x_i, $h(x)$ deceases fastest along the direction of opposite of gradient of x_i, i.e. let $p_i = -\nabla h(x_i)$.

To determine the step size α_i, we will use the method of **Line Search**:

Definition 3.1. *The error* $e_i = x_i - x^*$ *is the vector showing how far the iterated solution now from the solution.*

Definition 3.2. *The residual* $r_i = b - Ax_i$ *is the vector showing how far the iterated* b_i *now from the value of* b.

Note that $r_i = b - Ax_i = -\nabla h(x_i) = p_i = Ax^* - Ax_i = -Ae_i$, which gives an idea that we can think of residual as the direction of the steepest descent.

By Theorem 2.1, α_i minimizes $h(x_i)$ when the directional derivative $\frac{d}{d\alpha_i}h(x_i) = 0$:

$$\begin{aligned}
0 &= \frac{d}{d\alpha_i}h(x_i + \alpha_i p_i) = \nabla h(x_i + \alpha_i p_i)^T \cdot p_i \\
&= \nabla h(x_i + \alpha_i p_i)^T r_i = \big(b - A(x_i + \alpha_i p_i)\big)^T r_i \\
&= \big(b - A(x_i + \alpha_i r_i)\big)^T r_i = (b - Ax_i)^T r_i - \alpha_i (Ar_i)^T r_i \\
\alpha_i &= \frac{(b - Ax_i)^T r_i}{(Ar_i)^T r_i} = \frac{r_i^T r_i}{r_i^T A r_i}.
\end{aligned}$$

And to update r_{i+1} and p_{i+1}, we calculate :

$$\begin{aligned}
r_{i+1} = b - Ax_{i+1} = b - A(x_i + \alpha_i p_i) = r_i - \alpha_i A p_i \\
p_{i+1} = -\nabla h(x_{i+1}).
\end{aligned}$$

Now we have the direction and step size with iterated relation:

$$\begin{aligned}
p_i &= -\nabla h(x_i) = -Ax_i + b_i = r_i \\
p_{i+1} &= r_{i+1} + \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} p_i \\
\alpha_i &= \frac{r_i^T r_i}{r_i^T A r_i} \\
r_{i+1} &= r_i - \alpha_i A p_i.
\end{aligned}$$

Therefore now we can get the algorithm for **Conjugate Gradient Method**.

4. Conjugate Gradient Method Algorithm

input **:** x_0
output: x^*
$r_0 = b$;
$p_0 = r_0$;
for $i = 0, 1, \cdots,$ **do**
  $\alpha_i = \frac{r_i^T r_i}{p_i^T A p_i}$;
  $x_{i+1} = x_i + \alpha_i p_i$;
  $r_{i+1} = r_i - \alpha_i A p_i$;
  $p_{i+1} = r_{i+1} + \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} p_i$
end

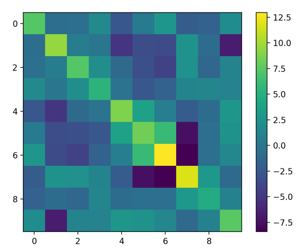
Algorithm 2: Algorithm for Conjugate Gradient Method

To implement this algorithm in python, first we need to generate an arbitrary symmetric positive definite matrix, to do so, let T be any nonsingular matrix, then $A = T \cdot T^T$ will be positive definite if A is square. The **Python** code is below:

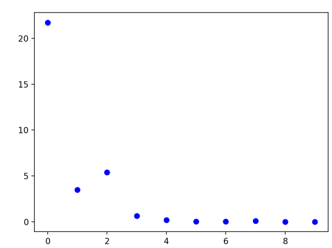
```
from scipy import random, linalg
import numpy as np
import matplotlib.pyplot as plt

matrixSize = 10
T = np.random.normal(size=[matrixSize,matrixSize])
A = np.dot(T,T.transpose())
b = np.ones(matrixSize)
x0=np.zeros(matrixSize)

if np.all(np.linalg.eigvals(A) > 0) :
    print ('the input matrix is positive definite\n')
else :
    print ('the input matrix is not positive definite\n')

detA = np.linalg.det(A)
print(detA)

x = x0
epsilon=1e-10   # tolerence
r = b   # residual
p = r  # direction
old = np.dot(np.transpose(r), r)
iter = 0
error = np.zeros(matrixSize)

for i in range(len(b)):
    Ap = np.dot(A,p)
    alpha = old / np.dot(np.transpose(p), Ap)   # step size
    x = x + alpha * p   # update gradient descent
    r = r - alpha * Ap   # update residual incrementally
    new = np.dot(np.transpose(r), r)
    if np.sqrt(new) < epsilon:
        break
    p = r + (new / old) * p   # determine new direction
    old = new
    error[i] = pow(np.linalg.norm(np.dot(A, x) - b),2)
    iter = iter +1


sol=x
print("solution is found in %s iterations" % iter)

finalError = error[iter-1]
print('final error is\n', finalError)
```

```
plt.imshow(A)
plt.colorbar()
plt.show()
plt.plot(error , 'bo')
plt.show()
```

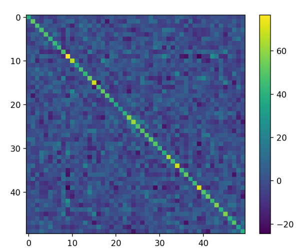
Which gives the graphs of error when the matrix size is 10 by 10, 50 by 50 and 100 by 100 :



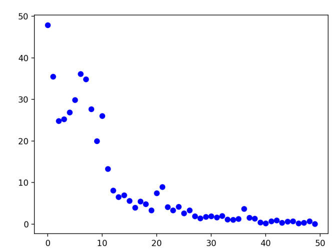
(a) A 10 by 10 matrix



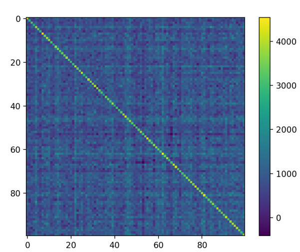
(b) Error for 10 by 10



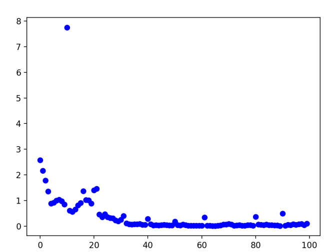
(c) A 50 by 50 matrix



(d) Error for 50 by 50



(e) A 100 by 100 matrix



(f) Error for 100 by 100

Figure 1: Errors of Conjugate Gradient Method

5. Discussion of Implementation

Theoretically the algorithm will work for any symmetric positive definite matrix, however when the eigenvalues of matrix A is very close to zero but not equal to zero, the result becomes very messy, the error will no longer behave a decreasing converging trend.

There are several other ways to generate "good enough" positive semidefinite matrix, here by "good enough", it means the eigenvalues of the matrix are very far from 0. The first thought is to adjust T by letting $T_{new} = a + b \cdot T$ with a, b are large positive integers, which will shifts the eigenvalues of A such that they are not close to 0. Another way is to think about Spectral Decomposition of A, first generates series of strict positive number that are not close to 0 to be the eigenvalues, then generate the corresponding orthogonal matrix. Note, to make A symmetric, we can always do $A_{new} = \frac{A^T + A}{2}$.

After several testing, it is found that the program always stops within n iteration, this leads to the discussion of convergence.

6. Convergence of Conjugate Gradient Method

Theorem 6.1. *For any e_i, it can be expressed as a linear combination of eigenvectors of A.*

Proof. Recall the fact that if A is symmetric, then the n eigenvectors of A are orthogonal, say $v_1, \cdots, v_n$. Since we can scale the eigenvectors arbitrarily, WLOG, we can assume $\|v_j\|_2 = 1, \forall j$. Hence have :

$$e_i = \sum_{j=1}^{n} \beta_j v_j.$$

∎

Hence we obtain the following equations :

$$r_i = -Ae_i = -\sum_{j=1}^{n} \beta_j \lambda_j v_j$$

$$\|r_i\|_2^2 = \sum_{j=1}^{n} \beta_j^2 \lambda_j^2$$

$$r_i^T A r_i = \sum_{j=1}^{n} \beta_j^2 \lambda_j^3$$

$$\|e_i\|_2^2 = \sum_{j=1}^{n} \beta^2$$

$$e_i^T A e_i = (\sum_{j=1}^{n} \beta_j v_j^T)(\sum_{j=1}^{n} \beta_j \lambda_j v_j) = \sum_{j=1}^{n} \beta_j^2 \lambda_j.$$

Hence we finally get :

$$\begin{aligned}
x_{i+1} &= x_i + \alpha_i \cdot p_i \\
e_{i+1} &= e_i + \frac{r_i^T r_i}{r_i^T A r_i} \cdot r_i \\
&= e_i + \frac{\sum_{j=1}^n \beta_j^2 \lambda_j^2}{\sum_{j=1}^n \beta_j^2 \lambda_j^3} \cdot r_i \\
&= e_i + \frac{\sum_{j=1}^n \beta_j^2 \lambda_j^2}{\sum_{j=1}^n \beta_j^2 \lambda_j^3} \cdot (-Ae_i).
\end{aligned}$$

Theorem 6.2. *Given any starting point, the conjugate gradient algorithm will achieve the minimizer x^* within n iterations.*

Proof. When e_i is one of the eigenvector of A with eigenvalue λ, then $r_i = -Ae_i = -\lambda e_i$ is also an eigenvectors, we have :

$$\begin{aligned}
e_{i+1} &= e_i + \frac{r_i^T r_i}{r_i^T A r_i} r_i \\
&= e_i + \frac{\lambda^2 \|e_i\|_2^2}{\lambda^2 e_i^T A e_i} \cdot (-\lambda e_i) \\
&= e_i + \frac{\lambda^2 \|e_i\|_2^2}{\lambda^2 e_i^T \cdot \lambda e_i} \cdot (-\lambda e_i) \\
&= e_i - e_i = 0.
\end{aligned}$$

This means choosing $\alpha_i = \lambda^{-1}$ will give the convergence immediately.

When e_i has more than one eigenvector component, all the eigenvectors $v_1, \cdots, v_n$ have a common eigenvalue λ, hence after n iterations :

$$\begin{aligned}
e^{i+1} &= e_i + \frac{\sum_{j=1}^n \beta_j^2 \lambda_j^2}{\sum_{j=1}^n \beta_j^2 \lambda_j^3} \cdot (-Ae_i) \\
&= e_i + \frac{\lambda^2 \sum_{j=1} \beta_j^2}{\lambda^3 \sum_{j=1} \beta_j^2} \cdot (-\lambda e_i) \\
&= 0.
\end{aligned}$$

Therefore after at most n iterations, the algorithm will get to the minimizer point. ∎