Second Order Methods and Conjugate Gradient Method in Optimization

Ariyan Zarei

University of Arizona

ariyanzarei@email.arizona.edu

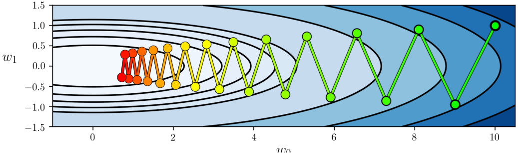
September 21, 2020

Overview

Review

Gradient Descent: a 1st order method for optimization

- Calculate Gradient of loss function and take step
- Issues
  - Convergence to a local minimum can be very slow
  - Not getting the best direction (zig zagging)
  - No guarantee to get to global minimum
  - Saddle points (min and max at the same time)



See Animation

Gradient Descent: a 1st order method for optimization

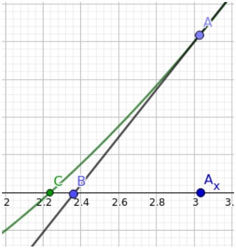
- Different variations of the same 1st order method
  - SGD and Momentum
  - RMSProp
  - Adam
  - AdaGrad
  - ...

Second order methods and Newton's method

Introduction to Newton's method

- Newton-Raphson's method
- Iterative second order algorithm
- Different applications
  - Zero finding of a function
  - Optimizing a function (minimum and maximum finding)

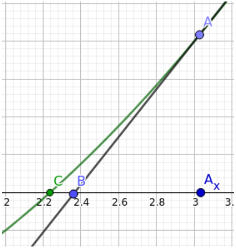
Newton's method for zero finding



$$f(x') = 0, f := \mathbb{R} \to \mathbb{R}$$

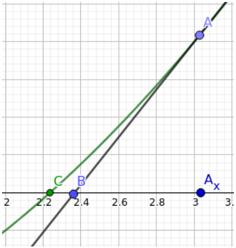
$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}, x_0 \in D_f$$

Newton's method for zero finding



- Intuition: Use a linear approximation of f and check where it intersects x axis.
- Tangent line as a first order approximate.

Newton's method for zero finding



$$\frac{B_y - A_y}{B_x - A_x} = f'(A_x)$$

$$\frac{0 - f(x_t)}{x_{t+1} - x_t} = f'(x_t)$$

Newton's method for zero finding

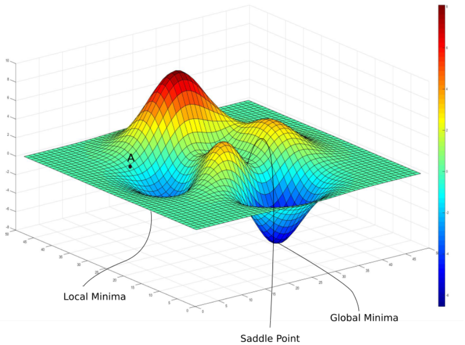
$$\frac{0 - f(x_t)}{x_{t+1} - x_t} = f'(x_t)$$

$$f'(x_t)x_{t+1} - f'(x_t)x_t = -f(x_t)$$

$$x_{t+1} = \frac{f'(x_t)x_t - f(x_t)}{f'(x_t)}$$

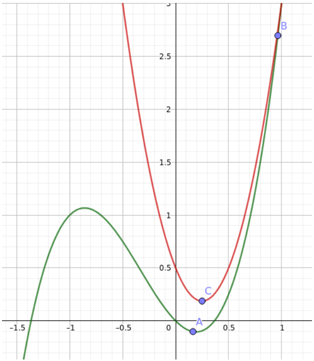
$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

Newton's method for optimization



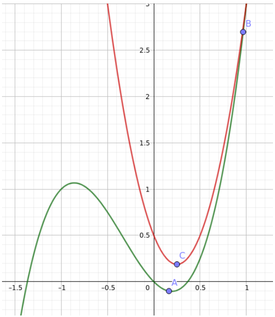
- Minimum of a function
    - Gradient equal to zero
    - Closed form
    - Iterative methods

Newton's method for optimization



$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$$

Newton's method for optimization



- Idea: Estimate a second order function at x_t and find it's minimum. This is a good direction.
- Second order methods are much faster than first order methods. The also provide a better direction.
- Reason is second order is a more accurate approximation than first order.

Recall on some math

- Gradient of f calculated at point a:

$$g = \nabla f(a) = \begin{bmatrix} \frac{\partial f(a)}{\partial x_1} \\ \frac{\partial f(a)}{\partial x_2} \\ \vdots \\ \frac{\partial f(a)}{\partial x_n} \end{bmatrix}$$

where $f := \mathbb{R}^n \to \mathbb{R}$ and $a = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$ is a point in the n dimensional space of $x_1, x_2, \dots, x_n$.

Recall on some math

- Hessian of f calculated at point a:

$$H = \nabla^2 f(a) = \begin{bmatrix} \frac{\partial^2 f(a)}{\partial x_1^2} & \frac{\partial^2 f(a)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(a)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(a)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(a)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(a)}{\partial x_2 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 f(a)}{\partial x_n \partial x_1} & \frac{\partial^2 f(a)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(a)}{\partial x_n^2} \end{bmatrix}$$

Hessian is the matrix of all possible second partial derivatives. It resembles the curvature of the function at a given point in each direction.

Recall on some math

- Eigenvectors of H correspond with directions where the curvature is independent of the other directions. Or in other words each eigenvector is the rate of change of gradient in one of the dimensions.
- Eigenvalues of H correspond with the amount of the curvature in each direction.

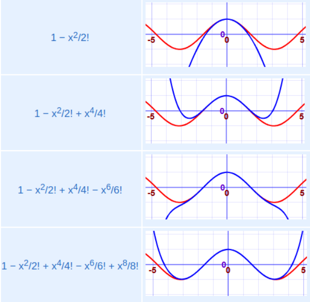
For the ith eigenvector and ith eigenvalue of matrix M we have $Mv_i = \lambda_i v_i$.

Recall on some math

- Taylor's Series of an univariate function f at a point a:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$$

Approximates an infinitely differentiable function $f := \mathbb{R} \rightarrow \mathbb{R}$ around a point a. We can truncate the series at each of the terms. For $cos(x)$ around zero:



Recall on some math

- Taylor's Series of a two variable function f at a point a, b:

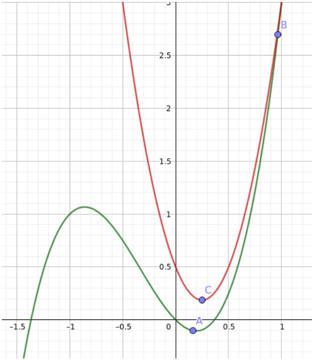
$$f(x, y) \approx f(a, b) + \frac{1}{1!}[f_x(a, b)(x - a) + f_y(a, b)(y - b)] +$$
$$\frac{1}{2!}[f_{xx}(a, b)(x - a)^2 + 2f_{xy}(a, b)(x - a)(y - b) + f_{yy}(a, b)(y - b)^2] + \dots$$

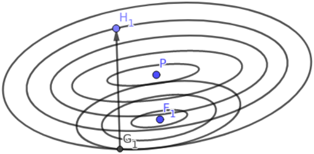
We can use dot products, matrices and vectors to simplify Taylor's series for multivariate functions:

$$f(X) = f(A) + \frac{1}{1!}\nabla f(A)(X - A) + \frac{1}{2!}(X - A)^\top \nabla^2 f(A)(X - A) + \dots$$

Where $X = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ and $A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$.

Newton's method for optimization





- Idea: Estimate a second order function at x_t and find it's minimum using closed form (derivative equal to zero). This is a good direction for finding the minimum of f.

Newton's method for optimization

Second order/quadratic approximation of f around a point A:

$$q(X) \approx f(A) + \nabla^\top f(A)(X - A) + \frac{1}{2}(X - A)^\top \nabla^2 f(A)(X - A)$$

$$\approx f(A) + \nabla^\top f(A)X - \nabla^\top f(A)A + \frac{1}{2}X^\top \nabla^2 f(A)X - A^\top \nabla^2 f(A)X + \frac{1}{2}A^\top \nabla^2 f(A)A$$

$$\approx \frac{1}{2}X^\top \nabla^2 f(A)X + \left[\nabla^\top f(A) - A^\top \nabla^2 f(A)\right] X + \left[f(A) - \nabla^\top f(A)A + \frac{1}{2}A^\top \nabla^2 f(A)A\right]$$

Newton's method for optimization

$$q(X) \approx \frac{1}{2} X^\top \nabla^2 f(A) X + \left[\nabla^\top f(A) - A^\top \nabla^2 f(A)\right] X + \left[f(A) - \nabla^\top f(A) A + \frac{1}{2} A^\top \nabla^2 f(A) A\right]$$

To minimize q, we need to have $\nabla q = 0$.

$$\nabla^2 f(A) X + \left[\nabla^\top f(A) - A^\top \nabla^2 f(A)\right] = 0$$

Now solve for X:

$$X = [\nabla^2 f(A)]^{-1} [A^\top \nabla^2 f(A) - \nabla^\top f(A)]$$

$$X = A - [\nabla^2 f(A)]^{-1} [\nabla^\top f(A)] = A - H^{-1} G^\top$$

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$$

Newton's method problems

- The Hessian might not be invertible (if Hessian is Positive Semi Definite and at least one of the eigenvalues is zero).
- The Hessian is Negative Definite. It will direct to the incorrect direction.
- In these cases we can switch to Gradient Descent. LevenbergMarquardt algorithm switches wisely between 1st and 2nd order methods.
- Computing Hessian and its inverse is expensive.
- Solve $Hy = G$ for y numerically instead of calculating inverse of H.

Conjugate Gradients

Quadratic Form and Linear System of Equations

- Conjugate Gradients is a method for solving **Sparse** linear system of equations in the form $Ax = b$. This is equivalent to minimizing a Quadratic Form.
- Quadratic Form
  - Quadratic function of a vector

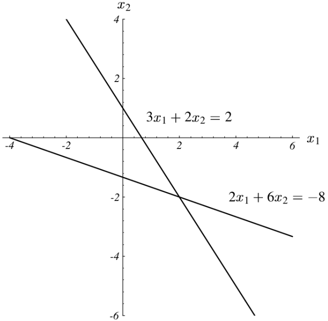
$$f(x) = \frac{1}{2}x^\top Ax - b^\top x + c$$

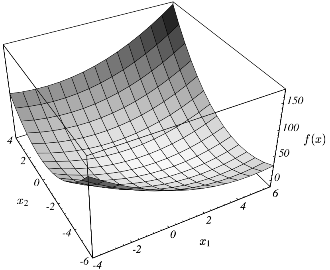
    where A is a matrix, b and x are vectors and c is a scalar.
- If A is symmetric and positive definite, minimizing the quadratic form is equal to solving $Ax = b$.

Quadratic Form and Linear System of Equations

Example:

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 8 \end{bmatrix}$$





Quadratic Form and Linear System of Equations

- Positive Definiteness → upward paraboloid bowl (intuition)
- Minimize the Quadratic form? set the gradient to zero and solve for x.

$$f(x) = \frac{1}{2}x^\top Ax - b^\top x + c$$

$$f'(x) = \frac{1}{2}A^\top x + \frac{1}{2}Ax - b$$

If A is symmetric:

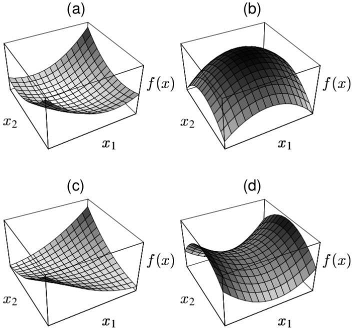
$$f'(x) = Ax - b$$

To minimize:

$$f'(x) = Ax - b = 0$$
$$Ax = b$$

Quadratic Form and Linear System of Equations

- So, the solution to $Ax = b$ is a critical point of the $f(x)$.
- Since A is positive definite (upward shape) as well as symmetric, the solution of the system of equation is a minimum of $f(x)$.
- Intuition: A tells us the shape of the surface. b and c tell us the minimum point (if any)

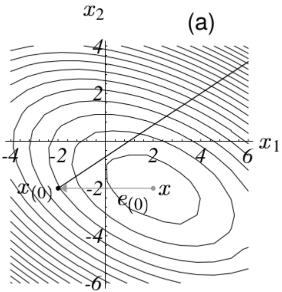


Steepest Descent Method

- Take series of steps until you are satisfied that you are close enough to minimum.
- Take each step in the direction which f decreases most quickly ($-\nabla f(x)$).
- Step size: use line search to gain maximum possible reduction in f in the direction of the $-\nabla f(x)$. This is where Gradient Descent and Steepest Descent are different from each other.

Steepest Descent Method

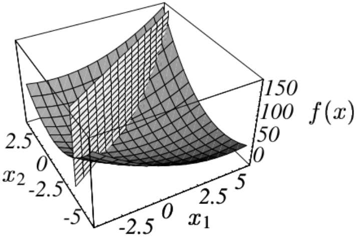
- Some Definitions:
  - error: $e_i = x_i - x^*$. How far we are from the solution.
  - residual: $r_i = b - Ax_i = -Ae_i$. How far we are from the value of b. In other words, residual is the error transformed by A into the space of b.
- Note: $r_i = -f'(x_i) = -\nabla f(x_i)$
- Update rule will be $x_{i+1} = x_i + \alpha r_i$
- Use line search to get the best value for α at each step.



Steepest Descent Method

- Line Search
  - Minimize $f(x_{i+1})$ along a line (the direction of the gradient).
  - $f(x_{i+1}) = f(x_i + \alpha r_i)$ is the parabola that is the intersection of the plain and paraboloid. It is a function of α.
  - Minimize? $\frac{d}{d\alpha} f(x_i + \alpha r_i) = 0$

(b)



Steepest Descent Method

Minimize? $\frac{d}{d\alpha} f(x_i + \alpha r_i) = 0$

$$\frac{d}{d\alpha} f(x_i + \alpha r_i) = f'(x_i + \alpha r_i)^\top \frac{d}{d\alpha}(x_i + \alpha r_i)$$

$$= f'(x_i + \alpha r_i)^\top r_i = f'(x_{i+1})^\top r_i$$

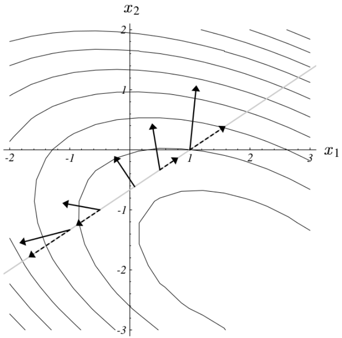
Remember $r_{i+1} = -f'(x_{i+1})$:

$$f'(x_{i+1})^\top r_i = -r_{i+1}^\top r_i = 0$$

Steepest Descent Method

$$\frac{d}{d\alpha} f(x_i + \alpha r_i) = f'(x_{i+1})^\top r_i = -r_{i+1}^\top r_i = 0$$

Intuition: The gradient at the minimum of the parabola, should be orthogonal to the previous gradient.



Steepest Descent Method

$$\begin{aligned}
r_{i+1}^\top r_i &= 0 \\
(b - Ax_{i+1})^\top r_i &= 0 \\
(b - A(x_i + \alpha r_i))^\top r_i &= 0 \\
(b - Ax_i - \alpha A r_i)^\top r_i &= 0 \\
(b - Ax_i)^\top r_i - \alpha (A r_i)^\top r_i &= 0 \\
(b - Ax_i)^\top r_i &= \alpha (A r_i)^\top r_i \\
r_i^\top r_i &= \alpha r_i^\top A^\top r_i \\
\alpha &= \frac{r_i^\top r_i}{r_i^\top A r_i}
\end{aligned}$$

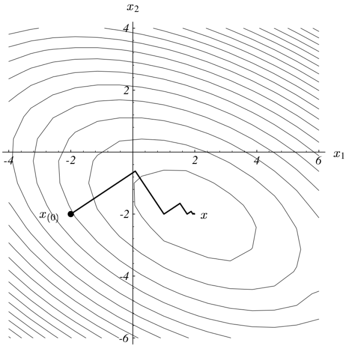
Steepest Descent Method

Summary:

$$r_i = b - Ax_i$$

$$\alpha_i = \frac{r_i^\top r_i}{r_i^\top A r_i}$$

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



Steepest Descent Method

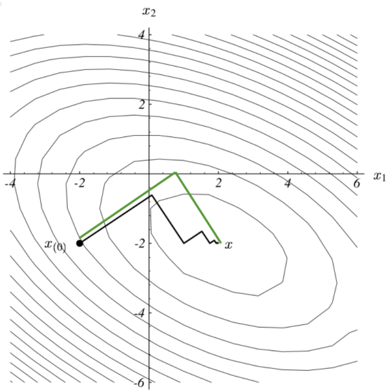
- Better and faster than GD
- Two matrix multiplication per iteration (expensive)
- Proof of convergence (Homework!)

Conjugacy and Conjugate Directions

- Remember: Steepest Descent takes steps in the same directions as two previous steps. (zig zagging path).
- Would be a lot faster if we took correct step size for each direction so that we never need to take step in that direction.
- Convergence and speed of minimization would be a lot faster (linear to the number of dimensions?)

Conjugacy and Conjugate Directions

- Idea? Find $n-1$ orthogonal directions and take exactly one step in each direction with the correct length to end up at minimum.



Conjugacy and Conjugate Directions

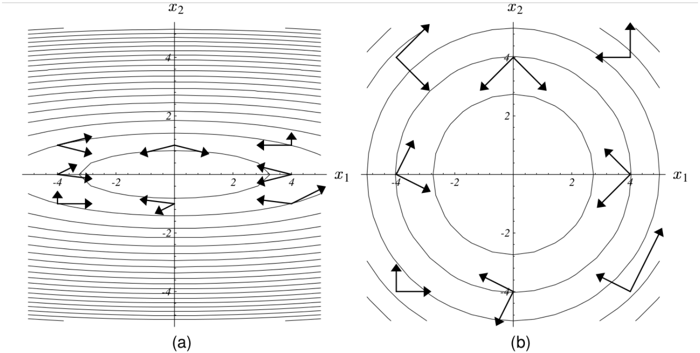
$$x_{i+1} = x_i + \alpha d_i$$

- Idea? Error vector at step $i+1$ be orthogonal to d_i.
- problem? We need to know the answer to calculate the error!! $(e_{i+1} = x_{i+1} - x^*)$
- Solution? Use A-orthogonality or conjugacy instead of orthogonality.
- Two vectors u and v are A-orthogonal or conjugate if:

$$u^\top A v = 0$$

- Remember that orthogonality of two vector u and v is $u^\top v = 0$.

Conjugacy and Conjugate Directions



Conjugacy and Conjugate Directions

- Two A-orthogonal or conjugate vectors

$$u^\top Av = 0$$

  - u be orthogonal with v transformed to the space of b using matrix A? (my own interpretation)
- New requirement for the step size (to take steps with the proper length in each direction):
  - Error of the next step be conjugate / A-orthogonal to the direction of the current step

$$d_i^\top Ae_{i+1} = 0$$

Conjugacy and Conjugate Directions

- This intuition comes from line search along the d_i direction:

$$\frac{d}{d\alpha} f(x_{i+1}) = 0$$

$$f'(x_{i+1})^\top \frac{d}{d\alpha} x_{i+1} = 0$$

Remember that $f'(x_{i+1}) = -r_{i+1}$ and $x_{i+1} = x_i + \alpha_i d_i$. So:

$$-r_{i+1}^\top d_i = 0$$

$$(Ae_{i+1})^\top d_i = 0$$

$$e_{i+1}^\top A^\top d_i = 0$$

$$d_i^\top A e_{i+1} = 0$$

Conjugacy and Conjugate Directions

- If we open up and solve for α_i we get:

$$\alpha_i = \frac{d_i^\top r_i}{d_i^\top A d_i}$$

- Note that if the direction vectors were residuals (negative of gradients), this would be the same formula used by the Steepest Descent method. ($\alpha_i = \frac{r_i^\top r_i}{r_i^\top A r_i}$)
- This procedure computes x^* in n steps.

Conjugacy and Conjugate Directions

- Gram-Schmidt Conjugation / Conjugate Gram-Schmidt process
  - A way to get A-orthogonal search directions d_i.
  - Algorithm
    - Start with a set of n independent vectors (like the unit vectors along coordinate axis) $u_0, u_1, \ldots, u_{n-1}$.
    - Set $d_0 = u_0$
    - for each d_i where $i > 0$, take u_i and subtract out any components that are not A-orthogonal to the previous d vectors.

$$d_i = u_i + \sum_{k=0}^{i-1} \beta_{ik} d_k$$

where for $i > j$:

$$\beta_{ij} = -\frac{u_i^\top A d_j}{d_j^\top A d_j}$$

Conjugate Gradients Method

- Conjugate Gradients Method is built upon all previous techniques and methods.
- We use the method of Conjugate Directions where our search directions are the conjugated of the residuals (or the gradients)
- That is why this algorithm is called Conjugate Gradients (Conjugat**ed** Gradients is much better as the paper suggests).

Conjugate Gradients Method

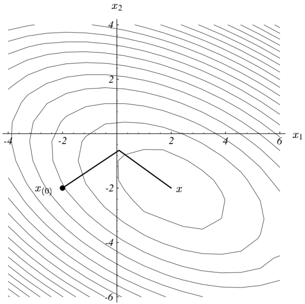
- We use r_i in the Gram-Schmidt process to get the search directions.
- Because of the orthogonality of the residuals and the fact that each residuals is a linear combination of previous residuals and Ad_{i-1}, Gram-Schmidt process (the summation) becomes a single term.
- Search directions can be calculated iteratively as each step runs.
- We can work around the math and find all the necessary equations for Conjugate Gradients method.

Conjugate Gradients Method

$$
\begin{aligned}
d_0 = r_0 &= b - Ax_0 \\
\alpha_i &= \frac{r_i^\top r_i}{d_i^\top A d_i} \\
x_{i+1} &= x_i + \alpha_i d_i \\
r_{i+1} &= r_i - \alpha_i A d_i \\
\beta_{i+1} &= \frac{r_{i+1}^\top r_{i+1}}{r_i^\top r_i} \\
d_{i+1} &= r_{i+1} + \beta_{i+1} d_i
\end{aligned}
$$

Conjugate Gradients Method

- Conjugate Gradient method is faster than Gradient Descent because of the conjugate search directions.
- It does not require calculating Hessian unlike Newton's method.



References

- Draw graphs: link
- 2nd Order Optimization material: link
- GD in NN and issues: link
- Hessian for DL: link
- Algs to train NN: link
- Youtube videos for Newton method: link and link
- Intro to Newton method: link
- Taylor series: link
- Taylor series for Multivariate functions: link
- Conjugate Gradient Method: link

Thank you for your attention

I will post the slides to my homepage at
`http://vision.cs.arizona.edu/ariyanzarei/`