Optimization & Operational Research : Second Part

Antoine Gourru

Slides built by Guillaume Metzler, updated by Ievgen Redko and Anna Karolin

January 2024 - Semester II

Algorithms

Setup

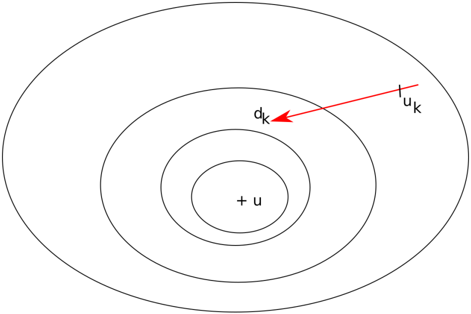
Given a function f and a non empty set $\mathcal{U}$ and **knowing there is a solution** to the problem : $f(u) = \min_{v \in \mathcal{U}} f(v)$.

Idea : build a series $(u_k)_{k \in \mathbb{N}}$ which converges to u.

Setup

Given a function f and a non empty set $\mathcal{U}$ and **knowing there is a solution** to the problem : $f(u) = \min_{v \in \mathcal{U}} f(v)$.

Idea : build a series $(u_k)_{k \in \mathbb{N}}$ which converges to u.

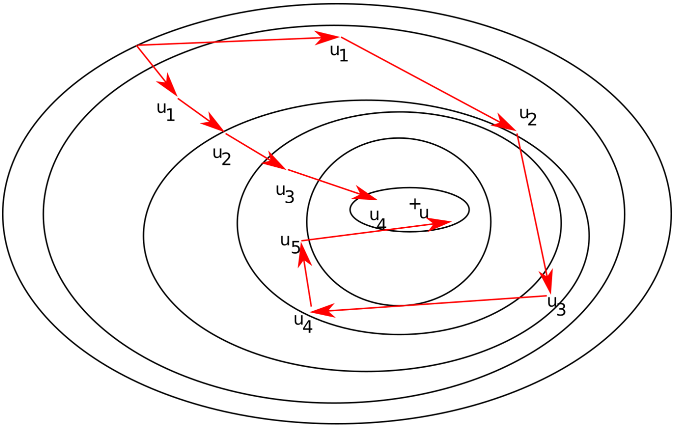


Algorithm :

- Take an initial value u_0.
- $u_k \rightarrow u_{k+1}$: Choose a direction d_k and minimize the function f along this direction.
- Solve $\arg\min_{\rho > 0} f(u_k - \rho d_k) = \rho_k$
- $u_{k+1} = u_k - \rho_k d_k$

Direction of descent

How to choose the direction d_k ?



Some ways seem to be **faster** than others to reach the solution

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0

2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle\nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle\nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Direction of descent

1. Recall that

$$f(u_k - \rho d_k) = f(u_k) - \rho\langle \nabla f(u_k), d_k\rangle + \rho\varepsilon(\rho)$$

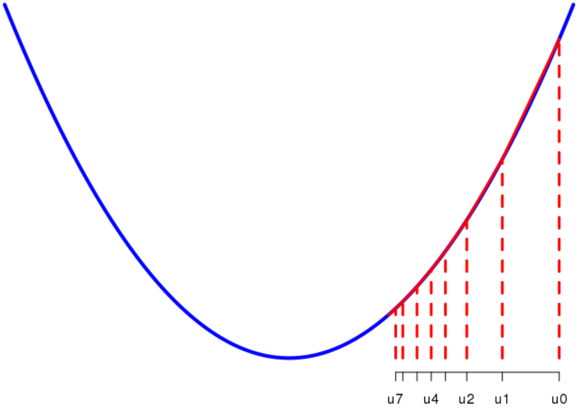
   when ρ is close to 0
2. To minimize f we choose d_k that maximizes $\langle \nabla f(u_k), d_k\rangle$
3. Due to Cauchy-Scwhartz Inequality, we have $d_k = \frac{\nabla f(u_k)}{||f(u_k)||}$ (assuming $\|d_k\| = 1$)
4. Leads to the algorithm
   - Choose u_0 to initialize the algorithm,
   - set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$ for $\rho_k > 0$
   - till $\|\nabla f(u_k)\| \leq \varepsilon$.

Summing up

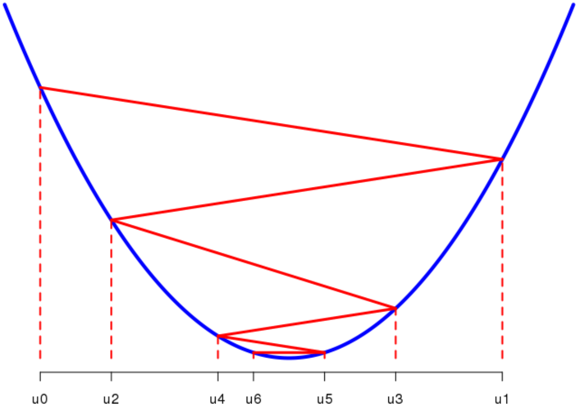
1. There exists several ways to use the gradient
2. We focus on gradient descent algorithms and their variants.
   - **Gradient Descent, Line Search, Newton's Method,...**

Other algorithms that **do not rely on** the derivatives of the function.

Gradient descent : choose the step 1/3



Gradient descent : choose the step 2/3



Gradient descent : choose the step 3/3

- If the step is too large, the sequence oscillates near the optimum.
- If the step is too small, the algorithm needs a large number of iterations.

Can **choose the step** for the gradient descents method **optimally** !

Gradient descent : with optimal step

Idea : choose the step that minimizes the objective function along a given direction.

Gradient descent : with optimal step

Idea : choose the step that minimizes the objective function along a given direction.

- Choose u_0 to initialize the algorithm,
- for $k = 0, 1, \ldots$ solve $\arg\min_{\rho > 0} f(u_k - \rho \nabla f(u_k))$,
- set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$
- till $\|\nabla f(u_k)\| \leq \varepsilon$.

Gradient descent : with optimal step

Idea : choose the step that minimizes the objective function along a given direction.

- Choose u_0 to initialize the algorithm,
- for $k = 0, 1, \dots$ solve $\arg\min_{\rho > 0} f(u_k - \rho \nabla f(u_k))$,
- set $u_{k+1} = u_k - \rho_k \nabla f(u_k)$
- till $\|\nabla f(u_k)\| \leq \varepsilon$.

This algorithm is called the Gradient Descent with optimal step.

Gradient descent : with optimal step

Definition

Let f be a convex and continuously differentiable function on $\mathbb{R}^n$. We say that f is strongly convex or α-elliptical if it exists $\alpha > 0$ such that

$$\langle \nabla f(v) - \nabla f(u), v - u \rangle \geq \alpha \|v - u\|, \ \forall u, v \in \mathbb{R}^n$$

What can we say about $\langle \nabla f(u_{k+1}), \nabla f(u_k) \rangle$ based on $\rho_k = \arg\min_{\rho > 0} f(u_k - \rho d_k)$?

Gradient descent : with optimal step

If ρ_k minimize $f(u_k - \rho_k d_k)$ we have :

$$\frac{\partial}{\partial \rho} f(u_k - \rho \nabla f(u_k))\big|_{\rho=\rho_k} = 0,$$

$$\iff \langle \nabla f(u_k - \rho_k \nabla f(u_k), \nabla f(u_k)) \rangle = 0,$$

$$\iff \langle \nabla f(u_{k+1}), \nabla f(u_k)) \rangle = 0.$$

The last equality is called the optimality condition.

Proposition

If f is a **strongly convex** then GD with optimal step converges

Gradient descent : with optimal step

Let A be a symmetric and PSD and $b \in \mathbb{R}^n$. We want to optimize

$$f(v) = \frac{1}{2}\langle Av, v\rangle - \langle b, v\rangle$$

- Calculate the gradient : $\nabla f(u_k) = Au_k - b$
- We then have to solve : $\rho_k = \arg\min_{\rho>0} f(u_k - \rho d_k)$. The optimality condition gives us : $\langle \nabla f(u_k), \nabla f(u_{k+1})\rangle = 0$

$$\begin{aligned} \nabla f(u_{k+1}) &= Au_{k+1} - b \\ &= A(u_k - \rho_k(Au_k - b) - b \\ &= Au_k - b - \rho_k A(Au_k - b) \end{aligned}$$

Gradient descent : with optimal step

$$\begin{aligned} \Rightarrow \langle Au_k - b, Au_k - b - \rho_k A(Au_k - b)\rangle &= 0 \\ \Rightarrow \langle Au_k - b, Au_k - b\rangle &= \langle Au_k - b, \rho_k A(Au_k - b)\rangle \\ \Rightarrow \rho_k &= \frac{\langle Au_k - b, Au_k - b\rangle}{\langle Au_k - b, A(Au_k - b)\rangle} \end{aligned}$$

We finally have the following algorithm :

- Initialize $u_0 \in \mathbb{R}^n$
- At each step, calculate $\rho_k = \dfrac{\|Au_k - b\|^2}{\|Au_k - b\|_A^2}$
- Set $u_{k+1} = u_k - \rho_k(Au_k - b)$
- Stop if $\|\nabla J(u_{k+1})\| = \|Au_{k+1} - b\| \leq \epsilon$

Gradient descent : with optimal step

Exercise

Consider the matrices $A = \begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ and the application f defined by $f(v) = \langle Av, v \rangle + \langle b, v \rangle$

1. Explain why f in convex.
2. Solve the problem $u = \underset{v \in \mathbb{R}^2}{\arg\min}\ f(v)$.
3. For a given vector u_k, calculate ∇f_{u_k} and ρ_k.
4. Implement the presented method to solve this problem.

Correction

- f is defined as a quadratic function where A is PSD, so f is convex.
- We have to solve :

$$J_{f(x,y)} = \begin{pmatrix} 12x + 4y + 2, & 4x + 8y + 3 \end{pmatrix} = (0, 0).$$

The solution is $\left(-\frac{1}{20}, -\frac{7}{20}\right)$.

- Let set $u_k = (v_1, v_2)$ then :

$$\nabla f_{u_k} = \begin{pmatrix} 12v_1 + 4v_2 + 2, & 4v_1 + 8v_2 + 3 \end{pmatrix},$$

and $\rho_k = \dfrac{\|2Au_k + b\|_2^2}{\|2Au_k + b\|_A^2}$

Exercise

Let f be the function defined by : $f(x, y) = 4x^2 - 4xy + 2y^2$.

1. Is the function f convex ?
2. Apply the gradient descent with optimal step to calculate the three first steps of the algorithm using $(x_0, y_0) = (1, 1)$.

Correction 1/3

- The function f can be rewritten as : $f(u) = \frac{1}{2}u^T Au - b^T u$, where $b = (0,0)^T$ and $A = \begin{pmatrix} 8 & -4 \\ -4 & 4 \end{pmatrix}$. The function f is a quadratic function, furthermore the matrix A is PSD so the function f is convex.
- The optimal learning rate is given by :

$$\rho_k \frac{\|Au_k - b\|_2^2}{\|Au_k - b\|_A^2},$$

where the matrix A and the vector b were previously introduced.

Correction 2/3

- The function f can be rewritten as : $f(u) = \frac{1}{2}u^T Au - b^T u$, where $b = (0\ 0)^T$ and $A = \begin{pmatrix} 8 & -4 \\ -4 & 4 \end{pmatrix}$. The function f is a quadratic function, furthermore the matrix A is PSD so the function f is convex.
- The optimal learning rate is given by :

$$\rho_k = \frac{\|Au_k - b\|_2^2}{\|Au_k - b\|_A^2},$$

where the matrix A and the vector b were previously introduced. Recall that the process is defined by :

$$u_{k+1} = u_k - \rho_k \nabla f(u_k).$$

We will now apply this process to compute the three first iterations.

Correction 3/3

1. For the first iteration : $\rho_0 = \frac{\|Au_0\|_2^2}{\|Au_0\|_A^2} = \frac{16}{128} = \frac{1}{8}$. And $\nabla f(u_0) = Au_0 = (4\ 0)^T$.

$$u_1 = (1\ 1)^T - \frac{1}{8}(4\ 0)^T = (0.5\ 1)^T.$$

2. For the second iteration : $\nabla f(u_1) = Au_1 = (0\ 2)^T$. The learning rate is given by : $\rho_1 = \frac{\|Au_1\|_2^2}{\|Au_1\|_A^2} = \frac{4}{16} = \frac{1}{4}$. Thus u_2 is given by :

$$u_2 = (0.5\ 1)^T - \frac{1}{4}(0\ 2)^T = (0.5\ 0.5)^T.$$

3. For the third iteration : $\nabla f(u_2) = Au_2 = (2\ 0)^T$. The learning rate is given by : $\rho_2 = \frac{\|Au_2\|_2^2}{\|Au_2\|_A^2} = \frac{4}{32} = \frac{1}{8}$. Thus u_3 is given by :

$$u_3 = (0.5\ 0.5)^T - \frac{1}{8}(2\ 0)^T = (0.25\ 0.5)^T.$$

Gradient Descent : Armijo Criterium

Idea : use a linear search to find the **learning rate**.
Given a $\theta \in]0, 1[$, choose the greatest ρ such that :

$$f(u_k - \rho \nabla f(u_k)) \leq f(u_k) - \theta \rho \|\nabla f(u_k)\|^2.$$

At each step, we reduce the function's value of at least $\theta \|\nabla f(u_k)\|$.

Gradient Descent : Armijo Criterium

Idea : use a linear search to find the **learning rate**.
Given a $\theta \in]0,1[$, choose the greatest ρ such that :

$$f(u_k - \rho \nabla f(u_k)) \leq f(u_k) - \theta\rho \|\nabla f(u_k)\|^2.$$

At each step, we reduce the function's value of at least $\theta\|\nabla f(u_k)\|$.

Armijo's condition :

- Choose $\alpha_0 > 0$ and $0 < \theta < 1$,
- Choose the greatest $s \in \mathbb{Z}$ such that :

$$f(u_k - \alpha_0 2^s \nabla f(u_k)) \leq f(u_k) - 2^s \alpha_0 \theta \|\nabla f(u_k)\|^2.$$

- Set $u_{k+1} \leftarrow u_k - \alpha_0 2^s \nabla f(u_k)$.

Gradient Descent : Armijo Criterium and Wolfe's Criteria

Theorem

If the function f is **strictly convex** and if its gradient ∇f is **Lipschitz**, then the Armijo's algorithm **converge**.

If we add the following condition to the previous one, given $0 < \theta < \eta < 1$:

$$\langle \nabla f(u_k), \nabla f(u_k - \rho \nabla f(u_k)) \rangle \geq \eta \|\nabla f(u_k)\|^2,$$

we get the Wolfe's Criteria

Conjugate Gradient

Definition

Let A be a **symmetric PD** matrix and u, v two vectors. u, v are **conjugate** with respect to A if

$$\langle Au, v \rangle = 0$$

Conjugate Gradient

Definition

Let A be a **symmetric PD** matrix and u, v two vectors. u, v are **conjugate** with respect to A if

$$\langle Au, v \rangle = 0$$

Let A be a **symmetric PD** matrix and f the function defined by

$$f(v) = \frac{1}{2}\langle Av, v \rangle - \langle b, v \rangle.$$

The objective is to build a series of **conjugate descent direction**

Conjugate Gradient

- Let $u_0 \in \mathbb{R}^n$, define a first direction of descent $d_0 = \nabla f(u_0)$ and minimize f along this direction :

$$\arg\min_{\alpha_0} f(u_0 - \alpha_0 d_0).$$

- Solving this problem we get :

$$\alpha_0 = \frac{\langle \nabla f(u_0), d_0 \rangle}{\langle A d_0, d_0 \rangle}.$$

- We set $u_1 = u_0 - \alpha_0 d_0$
- To build $d_1 = \nabla f(u_1) + \beta_0 d_0$, we need to find the value of $\beta_0 \in \mathbb{R}$ such that

$$\langle A d_1, d_0 \rangle = 0.$$

Conjugate Gradient

- We then have to solve $\langle A\nabla f(u_1), d_0\rangle + \langle A\beta_0 d_0, d_0\rangle = 0$. The solution is given by

$$\beta_0 = -\frac{\langle A\nabla f(u_1), d_0\rangle}{\langle Ad_0, d_0\rangle}.$$

Once it's done, you'll do as before.

You set $\alpha_1 = \arg\min_{\alpha} f(u_1 - \alpha d_1)$.

Set $u_2 = u_1 - \alpha_1 d_1$. And so on ...

Conjugate Gradient : Summary

Algorithm :

- Choose $u_0 \in \mathbb{R}^n$ and $d_0 = \nabla f(u_0)$.
- Set $\alpha_0 = \dfrac{\langle \nabla f(u_0), d_0 \rangle}{\langle A d_0, d_0, \rangle}$ and $u_1 = u_0 - \alpha_0 d_0$.
- $\beta_0 = -\dfrac{\langle A \nabla f(u_1), d_0 \rangle}{\langle A d_0, d_0 \rangle}$.

For $k \geq 1$ do,

- Set $d_k = \nabla f(u_k) + \beta_{k-1} d_{k-1}$.
- Set $\alpha_k = \dfrac{\langle \nabla f(u_k), d_k \rangle}{\langle A d_k, d_k, \rangle}$ and $u_{k+1} = u_k - \alpha_k d_k$.
- Set $\beta_k = \dfrac{\langle A \nabla f(u_{k+1}), d_k \rangle}{\langle A d_k, d_k \rangle}$

Untill $\|\nabla f(u_{k+1})\| \leq \varepsilon$.

Conjugate Gradient : Results

Proposition

For all $1 \leq k \leq n$ such that $\nabla f(u_0), \ldots, \nabla f(u_n)$ are non equal to zero, we have the following relations for all $0 \leq l \leq k-1$:

$$\langle \nabla f(u_k), \nabla f(u_l) \rangle = 0$$

and

$$\langle Ad_k, d_l \rangle = 0.$$

Theorem

If A is a symetric positive and definite matrix, then the conjugate gradient method converges with **at most** n **steps**.

Try to prove the proposition by induction at home

Newton's Method

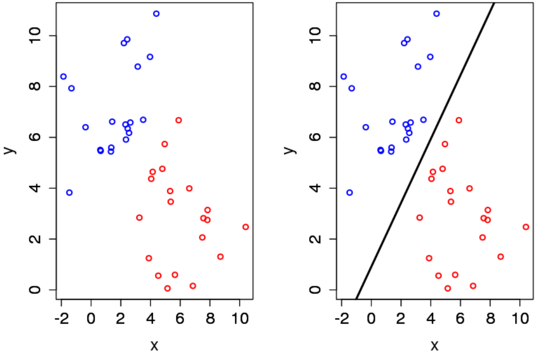
The Newton's Method is a gradient descent algorithm that refines the direction of the descent as follows :

$$u_{k+1} \leftarrow u_k - \left(\nabla^2 f(u_k)\right)^{-1} \cdot \nabla f(u_k).$$

- ✓ Requires less iterations to converge
- × Requires the inverse of the Hessian of the function we want to optimize ($\Theta(n^3)$).
- × The Hessian is not always invertible at a given point.

Newton's Method

Let's come back to the logistic regression.
We want to find a model that predict the class of our data.



→ An example of straight line that separate the two classes using logistic regression.

Newton's Method

For Logistic Regression, we want to maximize $l(x, a)$ with a **possible** solution given by solving the equation :

$$\nabla_a l(x, a) = \nabla_a \left(\sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right) = 0,$$

where $p = \big(1 + \exp(-a^T x)\big)^{-1}$.

Explain why the log-likelihood is **concave**. Calculate the **first and second derivatives** of the function l.

Newton's Method

If we apply the Newton's Method to the logistic regression we have

$$\nabla_a l(x,a) = \sum_{i=1}^{n} (y_i - p_i)\, x_i, \quad \nabla_a^2 l(x,a) = -\sum_{i=1}^{n} p_i(1-p_i) x_i x_i^T$$

We can then write the algorithm :

- Choose a_0,
- Calculate $\nabla_a l(x,a)$ and $\left(\nabla_a^2 l(x,a)\right)^{-1}$
- Set $a_{k+1} \leftarrow a_k - \left(\nabla_a^2 l(x,a)\right)^{-1} \nabla_a l(x,a)$
- Stop when $\|\nabla_a l(x,a)\| \leq \varepsilon$.

Quasi-Newton's Method : Motivation

Idea : avoid calculating the inverse of the Hessian matrix H_k^{-1} as follows :

$$\begin{aligned} u_{k+1} &= u_k - M_k \nabla f(u_k), \\ M_{k+1} &= M_k + C_k. \end{aligned}$$

Approximate the H_k^{-1} by matrix M_k at which, we add a matrix of correction C_k at each step

Quasi-Newton's Method : Motivation

Recall that :

$$\nabla f(u_k) = \nabla f(u_{k+1} + (u_k - u_{k+1})) \sim \nabla f(u_{k+1}) + \nabla^2 f(u_{k+1})(u_k - u_{k+1}),$$

we then have :

$$\left(\nabla^2 f(u_{k+1})\right)^{-1} (\nabla f(u_{k+1}) - \nabla f(u_k)) \sim u_{k+1} - u_k.$$

If we set :

$$M_{k+1} = \left(\nabla^2 f(u_{k+1})\right)^{-1},\ \gamma_k = \nabla f(u_{k+1}) - \nabla f(u_k)$$

and $\delta_k = u_{k+1} - u_k$, we get the Quasi Newton's Condition :

$$M_{k+1}\gamma_k = \delta_k$$

Quasi-Newton's Method : Davidon-Fletcher-Powell

- Assume C_k is of rank 1, ie, C_k as $v_k v_k^T$ where $v_k \in \mathbb{R}^n$.
- The update becomes :

$$M_{k+1} = M_k + v_k v_k^T$$

.

- The Quasi Newton's Condition gives :

$$\begin{aligned} (M_k + v_k v_k^T)\gamma_k &= \delta_k, \\ M_k\gamma_k + v_k v_k^T \gamma_k &= \delta_k, \\ v_k v_k^T \gamma_k &= \delta_k - M_k\gamma_k, \\ v_k &= \frac{\delta_k - M_k\gamma_k}{v_k^T \gamma_k}. \end{aligned}$$

And the second line gives us :

$$v_k^T \gamma_k = (\gamma_k \delta_k - \gamma_k M_k \gamma_k)^{1/2}.$$

Quasi-Newton's Method : Broyden Algorithm

Broyden Algorithm

Algorithm

- Initialize $u_0 \in \mathbb{R}^n$ and M_0 (usually $M_0 = Id$),
- for $k \geq 0$ do
  - set $\rho_k = \underset{\rho \in \mathbb{R}}{\arg\min}\, f(u_k - \rho M_k \nabla f(u_k))$,
  - set $u_{k+1} = u_k - \rho_k M_k \nabla f(u_k)$,
  - set$M_{k+1} = M_k + \frac{(\delta_k - M_k\gamma_k)(\delta_k - M_k\gamma_k)^T}{(\delta_k - M_k\gamma_k)^T\gamma_k}$,

Untill $\|\nabla f(u_{k+1})\| \leq \varepsilon$.

Quasi-Newton's Method : Broyden-Fletcher-Goldfarb-Shanno

- Assume C_k is of rank 1, ie, C_k as $v_k v_k^T$ where $v_k \in \mathbb{R}^n$.
- The inverse of the Hessian, at each step, is then approximated by :

$$M_{k+1} = M_k + \left[1 + \frac{\langle M_k\gamma_k, \gamma_k\rangle}{\langle \delta_k, \gamma_k\rangle}\right] \frac{\delta_k \delta_k^T}{\langle \delta_k, \gamma_k\rangle} - \frac{\langle \delta_k, \gamma_k\rangle M_k + M_k \gamma_k \delta_k^T}{\langle \delta_k, \gamma_k\rangle}.$$

The algorithm is the same as the previous one.

Conclusion

- **Gradient descent with a constant learning rate :**
  - ✓ Easy to implement
  - × Convergence depends on the value of the learning rate
- **Gradient descent with an optimal step :**
  - ✓ Faster then simple gradient descent
  - × More costly in terms of time
- **Newton's Method :**
  - ✓ Faster than the two others.
  - ✓ Requires less iterations.
  - × Requires to invert the Hessian matrix

To go further

1. A more advanced **Adam algorithm** (used currently for DNNs)
2. **Projected gradient descent** seen later in the course