Modern Optimization Techniques

Lucas Rego Drumond

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Newton's Method

Outline

1. Review
2. The Newton's Method

Outline

Unconstrained Optimization Problems

An **unconstrained optimization problem** has the form:

$$\text{minimize} \quad f_0(\mathbf{x})$$

Where:

- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is convex, twice differentiable
- An optimal $\mathbf{x}^*$ exists and $f(\mathbf{x}^*)$ is attained and finite

Descent Methods

The next point is generated using

- A step size μ
- A direction $\Delta \mathbf{x}$ such that

$$f_0(\mathbf{x}^t + \mu \Delta \mathbf{x}^{t-1}) < f_0(\mathbf{x}^{t-1})$$

procedure DescentMethod
  input: f_0
  Get initial point $\mathbf{x}$
  repeat
    Get Update Direction $\Delta \mathbf{x}$
    Get Step Size μ
    $\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t + \mu \Delta \mathbf{x}^t$
  until convergence
  return $\mathbf{x}$, $f_0(\mathbf{x})$
end procedure

Methods seen so far

- Gradient Descent:

$$\Delta \mathbf{x} = -\nabla f_0(\mathbf{x})$$

- Stochastic Gradient Descent:
  - If the function is if the form $f_0(\mathbf{x}) = \sum_{i=1}^{m} g(\mathbf{x}, i)$:
  -

$$\Delta_i \mathbf{x} = -\nabla g(\mathbf{x}, i)$$

Outline

An idea using second order approximations

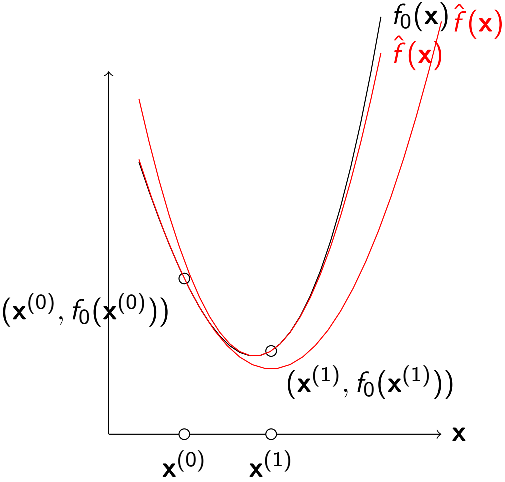
Be $f_0 : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}$:

$$\text{minimize} \quad f_0(\mathbf{x})$$

- Start with an initial solution $\mathbf{x}^{(t)}$
- Compute $\hat{f}$, a quadratic approximation of f_0 around $\mathbf{x}^{(t)}$
- Find $\mathbf{x}^{t+1} = \arg\min \hat{f}(\mathbf{x})$
- $t \leftarrow t + 1$
- Repeat until convergence

An idea using second order approximations

$$f_0(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-3)^2 + \frac{1}{10}\mathbf{x}^3$$



Taylor Approximation

Be $f : \mathbb{R}^n \to \mathbb{R}$ an infinitely differentiable function at some point $\mathbf{a} \in \mathbb{R}^n$

$f(\mathbf{x})$ can be approximated by the Taylor expansion of f, which is given by:

$$f(\mathbf{a}) + \frac{\nabla f(\mathbf{a})}{1!}(\mathbf{x}-\mathbf{a}) + \frac{\nabla^2 f(\mathbf{a})}{2!}(\mathbf{x}-\mathbf{a})^2 + \frac{\nabla^3 f(\mathbf{a})}{3!}(\mathbf{x}-\mathbf{a})^3 + \cdots$$
$$= \sum_{i=0}^{\infty} \frac{\nabla^i f(\mathbf{a})}{i!}(\mathbf{x}-\mathbf{a})^i$$

It can be shown that for a k large enough

$$f(\mathbf{x}) = \sum_{i=0}^{k} \frac{\nabla^i f(\mathbf{a})}{i!}(\mathbf{x}-\mathbf{a})^i$$

Second Order Approximation

Let us take the second order approximation of a twice differentiable function $f_0 : \mathbb{R}^n \to \mathbb{R}$ at a point $\mathbf{x}$:

$$\hat{f}(\mathbf{t}) = f_0(\mathbf{x}) + \nabla f_0(\mathbf{x})^T(\mathbf{t} - \mathbf{x}) + \frac{1}{2}(\mathbf{t} - \mathbf{x})^T \nabla^2 f_0(\mathbf{x})(\mathbf{t} - \mathbf{x})$$

We want to find the point $\mathbf{t} = \mathbf{x}^{(t+1)} = \arg\min \hat{f}$:

$$\begin{aligned}
\nabla_{\mathbf{t}} \hat{f}(\mathbf{t}) = \nabla f_0(\mathbf{x}) + \nabla^2 f_0(\mathbf{x})(\mathbf{t} - \mathbf{x}) &\overset{!}{=} 0 \\
\nabla f_0(\mathbf{x}) + \nabla^2 f_0(\mathbf{x})(\mathbf{t} - \mathbf{x}) &= 0 \\
\nabla^2 f_0(\mathbf{x})(\mathbf{t} - \mathbf{x}) &= -\nabla f_0(\mathbf{x}) \\
\mathbf{t} - \mathbf{x} &= -\nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x}) \\
\mathbf{t} &= \mathbf{x} - \nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x})
\end{aligned}$$

Newton's Step

- Be $f_0 : \mathbb{R}^n \to \mathbb{R}$ a twice differentiable convex function
- Newton's step uses the inverse of the Hessian matrix $\nabla^2 f_0(\mathbf{x})^{-1}$ and the gradient $\nabla f_0(\mathbf{x})$

$$\Delta^{\text{Newton}}\mathbf{x} = -\nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x})$$

Newton Decrement

We have a measure of the proximity of $\mathbf{x}$ to the optimal solution $\mathbf{x}^*$:

$$\lambda(\mathbf{x}) = \left(\nabla f_0(\mathbf{x})^T \nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x})\right)^{\frac{1}{2}}$$

- It provides a useful estimate of $f_0(\mathbf{x}) - f_0(\mathbf{x}^*)$ using the quadratic approximation $\hat{f}$:

$$f_0(\mathbf{x}) - \inf_{\alpha} \hat{f}(\alpha) = \frac{1}{2}\lambda(\mathbf{x})^2$$

- it is affine invariant (insensitive to the choice of coordinates)

Newton's method

```
procedure NEWTONS METHOD
input: f_0, tolerance ε > 0
    Get initial point x
    repeat
        Δx ← −∇²f_0(x)^{-1} ∇f_0(x)
        λ² ← ∇f_0(x)^T ∇²f_0(x)^{-1} ∇f_0(x)
        if λ²/2 ≤ ε then
            Quit
        end if
        Get Step Size μ
        x ← x + μΔx
    until convergence
    return x, f_0(x)
end procedure
```

Affine Invariance

We want to minimize $f_0(\mathbf{x})$.

Be T a positive-semidefinite matrix such that: $T\alpha = \mathbf{x}$

We can minimize $\tilde{f}(\alpha) = f_0(T\alpha) = f_0(\mathbf{x})$

The gradient of $\tilde{f}$ is:

$$\nabla \tilde{f}(\alpha) = T^\top \nabla f_0(T\alpha)$$

This means that the gradient method isn't affine invariant!

Considerations

- Works extremely well for a lot of problems
- f_0 must be twice differentiable
- The Hessian has n^2 elements.
- Compute and store the Hessian might hinder it's scalability for high dimensional problems
- Inverting the Hessian might be in some cases impractical

Newton's method - Example

For $\mathbf{x} \in \mathbb{R}$

$$\min_{\mathbf{x}} \quad (2\mathbf{x} - 4)^4$$

Algorithm:

- Let us use a fixed step size $\mu = 1$
- Initialize $\mathbf{x}^{(0)}$
- Repeat until convergence:
  - $\mathbf{x}^{(t)} \leftarrow -\frac{\nabla f_0(\mathbf{x}^{(t-1)})}{\nabla^2 f_0(\mathbf{x}(t-1))}$

Newton's method - Example

For $\mathbf{x} \in \mathbb{R}$

$$\min_{\mathbf{x}} \quad (2\mathbf{x} - 4)^4$$

Algorithm:

- $\nabla f_0(\mathbf{x}) = 8(2\mathbf{x} - 4)^3$
- $\nabla^2 f_0(\mathbf{x}) = 48(2\mathbf{x} - 4)^2$
- Step: $\Delta\mathbf{x} = -\nabla^2 f_0(\mathbf{x})^{-1} \nabla f_0(\mathbf{x})$
- $\Delta\mathbf{x} = -\frac{1}{6}(2\mathbf{x} - 4)$

Newton's method - Example

We Start at $\mathbf{x}^0 = 2.5$

- $\mathbf{x}^1 \leftarrow 2.5 - \frac{1}{6}(2 \cdot 2.5 - 4) = 2.3333$
- $\mathbf{x}^2 \leftarrow 2.33333 - \frac{1}{6}(2 \cdot 2.3333 - 4) = 2.22222$
- $\mathbf{x}^3 \leftarrow 2.22222 - \frac{1}{6}(2 \cdot 2.22222 - 4) = 2.148148$
- $\mathbf{x}^4 \leftarrow 2.148148 - \frac{1}{6}(2 \cdot 2.148148 - 4) = 2.098765$
- $\mathbf{x}^5 \leftarrow 2.098765 - \frac{1}{6}(2 \cdot 2.098765 - 4) = 2.065844$
- $\mathbf{x}^6 \leftarrow 2.065844 - \frac{1}{6}(2 \cdot 2.065844 - 4) = 2.043896$
- ...
- $\mathbf{x}^{20} \leftarrow 2.000226 - \frac{1}{6}(2 \cdot 2.0000134 - 4) = 2.00015$

Practical Example: Household Location

Suppose we have the following data about different households:

- Number of workers in the household (a_1)
- Household composition (a_2)
- Weekly household spending (a_3)
- Gross normal weekly household income (a_4)
- **Region** (y): North $y = 1$ or south $y = 0$

We want to creat a model of the location of the household

Practical Example: Household Spending

If we have data about m households, we can represent it as:

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,2} & \dots & a_{1,n} \\ 1 & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,2} & \dots & a_{m,n} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

We can model the household location is a linear combination of the household features with parameters $\mathbf{x}$:

$$\hat{y}_i = \sigma(\mathbf{x}^T \mathbf{a_i}) = \sigma(\mathbf{x}_0 1 + \mathbf{x}_1 a_{i,1} + \mathbf{x}_2 a_{i,2} + \mathbf{x}_3 a_{i,3} + \mathbf{x}_4 a_{i,4})$$

where: $\sigma(x) = \frac{1}{1+e^{-x}}$

Example II - The Logistic Regression

The logistic regression learning problem is

$$\text{minimize} \quad \sum_{i=1}^{m} y_i \log \sigma(\mathbf{x}^T \mathbf{a_i}) + (1 - y_i) \log(1 - \sigma(\mathbf{x}^T \mathbf{a_i}))$$

$$A_{m,n} = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 1 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{m,1} & a_{m,2} & a_{m,3} & a_{m,4} \end{pmatrix} \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

The Logistic Regression

First we need to compute the gradient of our objective function:

$$\text{minimize} \quad \sum_{i=1}^{m} y_i \log \sigma(\mathbf{x}^T \mathbf{a_i}) + (1 - y_i) \log(1 - \sigma(\mathbf{x}^T \mathbf{a_i}))$$

$$\begin{aligned}
\frac{\partial f_0}{\partial \mathbf{x}_k} &= \sum_{i=1}^{m} y_i \frac{1}{\sigma(\mathbf{x}^T \mathbf{a_i})} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(1 - \sigma(\mathbf{x}^T \mathbf{a_i})\right) a_{ik} \\
&\quad - (1 - y_i) \frac{1}{1 - \sigma(\mathbf{x}^T \mathbf{a_i})} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(1 - \sigma(\mathbf{x}^T \mathbf{a_i})\right) a_{ik} \\
&= \sum_{i=1}^{m} y_i a_{ik} \left(1 - \sigma(\mathbf{x}^T \mathbf{a_i})\right) - (1 - y_i) a_{ik} \sigma(\mathbf{x}^T \mathbf{a_i}) \\
&= \sum_{i=1}^{m} a_{ik} \left(y_i - \sigma(\mathbf{x}^T \mathbf{a_i})\right)
\end{aligned}$$

The Logistic Regression

$$\frac{\partial f_0}{\partial \mathbf{x}_k} = \sum_{i=1}^{m} a_{ik} \left(y_i - \sigma(\mathbf{x}^T \mathbf{a_i}) \right)$$

Now we need to compute the Hessian matrix:

$$\begin{aligned}\frac{\partial^2 f_0}{\partial \mathbf{x}_k \partial \mathbf{x}_j} &= \sum_{i=1}^{m} -a_{ik} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(1 - \sigma(\mathbf{x}^T \mathbf{a_i})\right) a_{ij} \\ &= \sum_{i=1}^{m} a_{ik} a_{ij} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(\sigma(\mathbf{x}^T \mathbf{a_i}) - 1\right)\end{aligned}$$

The Hessian H is an $n \times n$ matrix such that:

$$H_{k,j} = \sum_{i=1}^{m} a_{ik} a_{ij} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(\sigma(\mathbf{x}^T \mathbf{a_i}) - 1\right)$$

The Logistic Regression

So we have our gradient $\nabla f_0 \in \mathbb{R}^n$ such that

$$\nabla_{\mathbf{x}_k} f_0 = \sum_{i=1}^{m} a_{ik} \left(y_i - \sigma(\mathbf{x}^T \mathbf{a_i}) \right)$$

And the Hessian $H \in \mathbb{R}^{n \times n}$:

$$H_{k,j} = \sum_{i=1}^{m} a_{ik} a_{ij} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(\sigma(\mathbf{x}^T \mathbf{a_i}) - 1 \right)$$

the newton update rule is:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \mu H^{-1} \nabla f_0$$

Newton's Method for Logistic Regression - Considerations

The newton update rule is:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \mu H^{-1} \nabla f_0$$

Biggest problem:

How to efficiently compute H^{-1} for:

$$H_{k,j} = \sum_{i=1}^{m} a_{ik} a_{ij} \sigma(\mathbf{x}^T \mathbf{a_i}) \left(\sigma(\mathbf{x}^T \mathbf{a_i}) - 1 \right)$$

Considerations:

- H is symmetric: $H_{k,j} = H_{j,k}$