Introduction to Optimization

Prof. Dr. H. H. Takada

Quantitative Research – Itaú Asset Management
Institute of Mathematics and Statistics – University of São Paulo

The need for optimization

Machine learning often requires fitting a model to data. Often this means finding the parameters $\boldsymbol{\theta}$ of the model that 'best' fit the data.

Regression

For example, for regression based on training data $(\mathbf{x}^n, y^n)$ we might have a model $y(x|\boldsymbol{\theta})$ and wish to set $\boldsymbol{\theta}$ by minimizing

$$E(\theta) = \sum_n \left(y^n - y(\mathbf{x}^n|\theta)\right)^2$$

Complexity

In all but very simple cases, it is extremely difficult to find an algorithm that will guarantee to find the optimal $\boldsymbol{\theta}$.

A simple case: Linear regression

For example for a linear predictor

$$y(\mathbf{x}|\theta) \equiv \mathbf{x}^{\mathsf{T}}\boldsymbol{\theta}$$

$$E(\boldsymbol{\theta}) = \sum_n \left(y^n - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^n\right)^2$$

The optimum is given when the gradient wrt $\boldsymbol{\theta}$ is zero:

$$\frac{\partial E}{\partial \theta_i} = 2\sum_n \left(y^n - \boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}^n\right) x_i^n = 0$$

$$\underbrace{\sum_n y^n x_i^n}_{b_i} = \sum_j \underbrace{\sum_n x_i^n x_j^n}_{X_{ij}} \theta_j$$

Hence, in matrix form, this is

$$\mathbf{b} = \mathbf{X}\boldsymbol{\theta}, \rightarrow \boldsymbol{\theta} = \mathbf{X}^{-1}\mathbf{b}$$

which is a simple linear system that can be solved in $O\left((\dim \boldsymbol{\theta})^3\right)$ time.

Quadratic functions

A class of simple functions to optimize is, for symmetric $\mathbf{A}$:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} - \mathbf{x}^\mathsf{T}\mathbf{b}$$

This has a unique minimum if and only if $\mathbf{A}$ is positive definite. In this case, at the optimimum

$$\nabla f = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \rightarrow \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

For $\mathbf{x} + \boldsymbol{\delta}$, the new value of the function is

$$f(\mathbf{x} + \boldsymbol{\delta}) = f(\mathbf{x}) + \underbrace{\boldsymbol{\delta}^\mathsf{T}\nabla f}_{=0} + \frac{1}{2}\underbrace{\boldsymbol{\delta}^\mathsf{T}\mathbf{A}\boldsymbol{\delta}}_{\geq 0}$$

Hence $\mathbf{A}^{-1}\mathbf{b}$ is a minimum.

Gradient Descent

We wish to find $\mathbf{x}$ that minimizes $f(\mathbf{x})$. For general f there is no closed-form solution to this problem and we typically apply iterative methods.
For $\mathbf{x}_{k+1} \approx \mathbf{x}_k$,

$$f(\mathbf{x}_{k+1}) \approx f(\mathbf{x}_k) + (\mathbf{x}_{k+1} - \mathbf{x}_k)^{\mathsf{T}} \nabla f(\mathbf{x}_k)$$

setting

$$\mathbf{x}_{k+1} - \mathbf{x}_k = -\epsilon \nabla f(\mathbf{x}_k)$$

gives

$$f(\mathbf{x}_{k+1}) \approx f(\mathbf{x}_k) - \epsilon \left|\nabla f(\mathbf{x}_k)\right|^2$$

Hence, for a small ϵ, the algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \epsilon \nabla f(\mathbf{x}_k)$$

decreases f. We iterate until convergence.

Comments on gradient descent

Good

This is a very simple algorithm that is easy to implement.

Bad

Only improves the solution at each stage by a small amount. If ϵ is not small enough, the function may not decrease in value. In practice one needs to find a suitably small ϵ to guarantee convergence.

Ugly

The method is coordinate system dependent. Let $\mathbf{x} = \mathbf{M}\mathbf{y}$ and define $\hat{f}(\mathbf{y}) = f(\mathbf{x})$.
We now perform gradient descent on $\hat{f}(\mathbf{y})$:

$$[\mathbf{y}_{k+1}]_i = [\mathbf{y}_k]_i - \epsilon \sum_j \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial y_i} \rightarrow \mathbf{y}_{k+1} = \mathbf{y}_k - \epsilon \mathbf{M}^\mathsf{T} \nabla f(\mathbf{x}_k)$$

Hence

$$\mathbf{M}\mathbf{y}_{k+1} = \mathbf{M}\mathbf{y}_k - \epsilon \mathbf{M}\mathbf{M}^\mathsf{T} \nabla f(\mathbf{x}_k) \rightarrow \mathbf{x}_{k+1} = \mathbf{x}_k - \epsilon \mathbf{M}\mathbf{M}^\mathsf{T} \nabla f(\mathbf{x}_k)$$

The algorithm is coordinate system dependent (except for orthogonal transformations).

Line Search

One way to potentially improve on gradient descent is choose a particular direction $\mathbf{p}_k$ and search along there. We then find the minimum of the one dimensional problem

$$F(\lambda) = f(\mathbf{x}_k + \lambda \mathbf{p}_k)$$

Finding the optimal λ^* can be achieved using a standard one-dimensional optimization method. Once found we set

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda^* \mathbf{p}_k$$

and then choose another search direction $\mathbf{p}_{k+1}$ and iterate.

Good

By reducing the problem to a sequence of one dimensional optimization problems, at each stage the potential change in f is greater than would be typically achievable by gradient descent.

Search directions

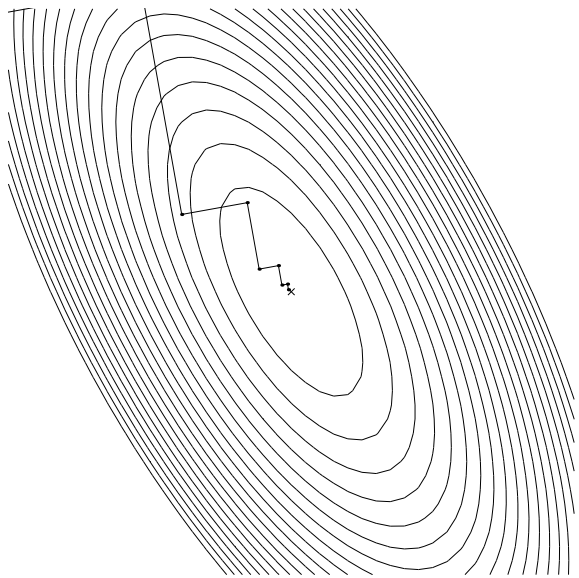
How do we choose a good search direction?

Choosing the search directions

It would seem reasonable to choose a search direction that points 'maximally downhill' from the current point $\mathbf{x}_k$. That is, to set

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$$

However, at least for quadratic functions, this is not optimal and leads to potentially zig-zag behaviour:



This zig-zag behaviour occurs for non axis-aligned surfaces.

Philosophy

Use Quadratic functions to gain insight

Much of the theory of optimisation is based on the following: Even though the problem is not quadratic, let's pretend the problem is quadratic and see what we would do in that case. This will inspire the solution for the general case.

Quadratic functions

In the quadratic function case, we get zig-zag behaviour if $\mathbf{A}$ is not diagonal. If we could find a coordinate transformation $\mathbf{x} = \mathbf{P}\hat{\mathbf{x}}$

$$\hat{f}(\hat{\mathbf{x}}) = \frac{1}{2}\hat{\mathbf{x}}^{\mathsf{T}}\mathbf{P}^{\mathsf{T}}\mathbf{A}\mathbf{P}\hat{\mathbf{x}} - \mathbf{b}^{\mathsf{T}}\mathbf{P}\hat{\mathbf{x}}$$

with

$$\mathbf{P}^{\mathsf{T}}\mathbf{A}\mathbf{P}$$

being diagonal, then we can perform line-search independently along each axis of $\hat{\mathbf{x}}$ and find the solution efficiently. This is achieved by the conjugate gradient algorithm.

Conjugate vectors

Two non-zero vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ are conjugate with respect to the $n \times n$ symmetric and positive definite matrix $\mathbf{A}$ if

$$\mathbf{p}_i^{\mathsf{T}} \mathbf{A} \mathbf{p}_j = 0$$

Searching along these conjugate directions effectively 'diagonalizes' the problem. To keep the argument simple, set $\mathbf{x}_1 = \mathbf{0}$ and in general

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

where the $\mathbf{p}_k$, $k = 1, \ldots, n$ form a conjugate set. Then for

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^{\mathsf{T}} \mathbf{A} \mathbf{x} - \mathbf{x}^{\mathsf{T}} \mathbf{b}$$

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) = f(\mathbf{x}_k) + \alpha_k \mathbf{p}_k^{\mathsf{T}} \mathbf{A} \mathbf{x}_k + f(\alpha_k \mathbf{p}_k)$$

Since $\mathbf{x}_k$ is a linear combination of $\mathbf{p}_1, \ldots, \mathbf{p}_{k-1}$ then

$$\mathbf{p}_k^{\mathsf{T}} \mathbf{A} \mathbf{x}_k = 0$$

and we can find the optimal α_k by simply minimizing $f(\alpha_k \mathbf{p}_k)$, independently of the previous search directions.

Conjugate Gradients

The question now is how to find the conjugate directions. As we saw, we only require that $\mathbf{p}_k$ is conjugate to $\mathbf{p}_1, \ldots, \mathbf{p}_{k-1}$. Defining $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$, and

$$\beta_k = \mathbf{g}_{k+1}^{\mathsf{T}} \mathbf{g}_{k+1} / (\mathbf{g}_k^{\mathsf{T}} \mathbf{g}_k)$$

one can show that for the quadratic function at least,

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k$$

is conjugate to all previous $\mathbf{p}_i$, $i = 1, \ldots, k$.
A more common alternative is the Polak-Ribière formula.

$$\beta_k = \frac{\mathbf{g}_{k+1}^{\mathsf{T}} (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^{\mathsf{T}} \mathbf{g}_k}$$

For quadratic functions, with $\dim \mathbf{x} = n$, conjugate gradients is guaranteed to find the optimum in n iterations, each iteration taking $O\left(n^2\right)$ operations. In a more general non-quadratic problem, no such guarantee exists.

Conjugate Gradients

This gives rise to the conjugate gradients for minimizing a function $f(\mathbf{x})$

1: $k = 1$
2: Choose $\mathbf{x}_1$.
3: $\mathbf{p}_1 = -\mathbf{g}_1$
4: **while** $\mathbf{g}_k \neq \mathbf{0}$ **do**
5: $\quad \alpha_k = \underset{\alpha_k}{\operatorname{argmin}}\ f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$ ▷ Line Search
6: $\quad \mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7: $\quad \beta_k := \mathbf{g}_{k+1}^\mathsf{T} \mathbf{g}_{k+1} / (\mathbf{g}_k^\mathsf{T} \mathbf{g}_k)$
8: $\quad \mathbf{p}_{k+1} := -\mathbf{g}_{k+1} + \beta_k \mathbf{p}_k$
9: $\quad k = k + 1$
10: **end while**

Newton's method

Consider a function $f(\mathbf{x})$ that we wish to find the minimum of. A Taylor expansion up to second order gives

$$f(\mathbf{x}+\boldsymbol{\Delta}) = f(\mathbf{x}) + \boldsymbol{\Delta}^{\mathsf{T}}\nabla f + \frac{1}{2}\boldsymbol{\Delta}^{\mathsf{T}}\mathbf{H}_f\boldsymbol{\Delta} + O(|\boldsymbol{\Delta}|^3)$$

The matrix $\mathbf{H}_f$ is the Hessian.
Differentiating the right hand side with respect to $\boldsymbol{\Delta}$ (or, equivalently, completing the square), we find that the right hand side has its lowest value when

$$\nabla f = -\mathbf{H}_f\boldsymbol{\Delta} \Rightarrow \boldsymbol{\Delta} = -\mathbf{H}_f^{-1}\nabla f$$

Hence, an optimization routine to minimize f is given by the Newton update

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \epsilon\mathbf{H}_f^{-1}\nabla f$$

For quadratic functions, Newton's method converges in one step (for $\epsilon = 1$). More generally one uses a value $\epsilon < 1$ to avoid overshooting effects.

Comments on Newton's method

Good

A benefit of Newton method over gradient descent is that the decrease in the objective function is invariant under a linear change of co-ordinates, $\mathbf{x} = \mathbf{M}\mathbf{y}$. Defining $\hat{f}(\mathbf{y}) \equiv f(\mathbf{x})$, the change in $\mathbf{y}$ under a Newton update is

$$-\mathbf{H}_{\hat{f}}^{-1}\nabla_{\mathbf{y}}\hat{f}$$

where $\nabla_{\mathbf{y}}\hat{f} = \mathbf{M}^{\mathsf{T}}\nabla_{\mathbf{x}}f$ $\mathbf{H}_{\hat{f}} = \mathbf{M}^{\mathsf{T}}\mathbf{H}_f\mathbf{M}$. In terms of the $\mathbf{x}$ coordinate system the change is

$$\mathbf{M}\Delta\mathbf{y} = -\mathbf{M}\mathbf{H}_{\hat{f}}^{-1}\nabla_{\mathbf{y}}\hat{f} = -\mathbf{M}\left(\mathbf{M}^{\mathsf{T}}\mathbf{H}_f\mathbf{M}\right)^{-1}\mathbf{M}^{\mathsf{T}}\nabla_{\mathbf{x}}f = -\mathbf{H}_f^{-1}\nabla_{\mathbf{x}}f = \Delta\mathbf{x}$$

so that the change is independent of the coordinate system (up to linear transformations of the coordinates).

Bad

Storing the Hessian and solving the linear system $\mathbf{H}_f^{-1}\nabla f$ is very expensive.

Quasi-Newton

In Quasi-Newton methods such as Broyden-Fletcher-Goldfarb-Shanno, an approximate inverse Hessian is formed iteratively.