Multi-Dimensional Optimization

Life gets much more complex for higher ($N > 1$) dimensional optimization.

In general, we start from a given point, pick a search direction and do a 1-D search in that direction.

Method of steepest descent:
If we want to get to a minimum, it makes sense to go down hill

Search direction $\equiv -\underset{\sim}{\nabla} F$

Thus we solve the one-D problem

$$\min_{\alpha} F\left(\underset{\sim}{x}^{(k)} - \alpha \underset{\sim}{\nabla} F(\underset{\sim}{x}^{(k)})\right)$$

to get

$$\underset{\sim}{x}^{(k+1)} = \underset{\sim}{x}^{(k)} - \alpha_{opt} \underset{\sim}{\nabla} F(\underset{\sim}{x}^{(k)})$$

where α_{opt} is the desired minimum

This method tends to be rather slow. The gradient often does not point towards the minimum! Let's work an example:

Let $F(\underset{\sim}{x}) = x_1^2 + 10x_2^2 + 100x_3^2$

This has the global minimum of $\underset{\sim}{x}^* = 0 \quad (x_1^* = x_2^* = x_3^* = 0)$

Now $\underset{\sim}{\nabla} F \equiv (2x_1, 20x_2, 200x_3)^T$

Thus at each iteration we want to solve:

$$\min_{\alpha} F(\underset{\sim}{x} - \alpha \nabla F) = \min_{\alpha} \left\{ (x_1 - 2\alpha x_1)^2 + 10(x_2 - 20\alpha x_2)^2 + 100(x_3 - 200\alpha x_3)^2 \right\}$$

We can actually get a linear equation for α for this particular problem.

$$\alpha = \frac{x_1^2 + 10^2 x_2^2 + 10^4 x_3^2}{2(x_1^2 + 10^3 x_2^2 + 10^6 x_3^2)}$$

After <u>180</u> iterations we get:

$$x^{(0)} = (1, 1, 1) \leftarrow \text{starting point}$$

$$x^{(180)} = (1.07 \times 10^{-4}, \; 0, \; 3.01 \times 10^{-6})$$

Why? The problem has different curvature in different directions.

You can think of the algorithm as wandering back and forth across a narrow river valley and only slowly rolling down to the sea.

We can do better (sometimes) with a multi-dim. Newton's method.

Let's keep an extra term in the Taylor Series:

$$F(\underset{\sim}{x}) = F(\underset{\sim}{x}_0) + \underset{\sim}{\nabla} F(\underset{\sim}{x}_0) \cdot (\underset{\sim}{x} - x_0)$$

$$+ \frac{1}{2} \underset{\sim}{\nabla} \underset{\sim}{\nabla} F(\underset{\sim}{x}_0) \cdot (\underset{\sim}{x} - \underset{\sim}{x}_0)(\underset{\sim}{x} - \underset{\sim}{x}_0)$$

$$+ \ldots$$

The matrix $\underset{\sim}{\nabla}\underset{\sim}{\nabla} F$ is the Hessian Matrix

$$\underset{\sim}{\nabla}\underset{\sim}{\nabla} F \equiv \frac{\partial^2 F}{\partial x_i \partial x_j}$$

Remember, we seek the critical points of F. Thus:

$$\underset{\sim}{\nabla} F = 0 = \underset{\sim}{\nabla} F(x_0) + \underset{\sim}{\nabla}\underset{\sim}{\nabla} F(x_0)\,(\underset{\sim}{x} - \underset{\sim}{x}_0) + \ldots$$

If we truncate after this term, we get an equation for the next guess at the critical point:

$$\underset{\sim}{x}^{(k+1)} = \underset{\sim}{x}^{(k)} - \left(\underset{\sim}{\nabla}\underset{\sim}{\nabla} F(\underset{\sim}{x}^{k})\right)^{-1} \underset{\sim}{\nabla} F(\underset{\sim}{x}^{(k)})$$

These are Newton's equations for this problem!

The method will converge quadratically near the critical point, but it can fail to converge.

Suppose we want to find a <u>minimum</u>. We want an algorithm which does the following:

1) Computes $F(\underset{\sim}{x}^{(k)})$, $\underset{\sim}{\nabla} F(x^{(k)})$

2) Computes some descent direction $\underset{\sim}{p}$ such that:

$$F(\underset{\sim}{x}^{(k)} + \varepsilon \underset{\sim}{p}) < F(x^{(k)})$$

for small ε

We can do this via steepest descent:

$$\underset{\sim}{p} = -\underset{\sim}{\nabla} F(x^{(k)})$$

or via Newton's method:

$$\underset{\sim}{p} = -\left[\underset{\sim}{\nabla}\underset{\sim}{\nabla} F\right]^{-1} \underset{\sim}{\nabla} F$$

Note that Newton's method does not <u>necessarily</u> point to a minimum!

3) Line search along vector $\underset{\sim}{p}$:

find some α s.t.

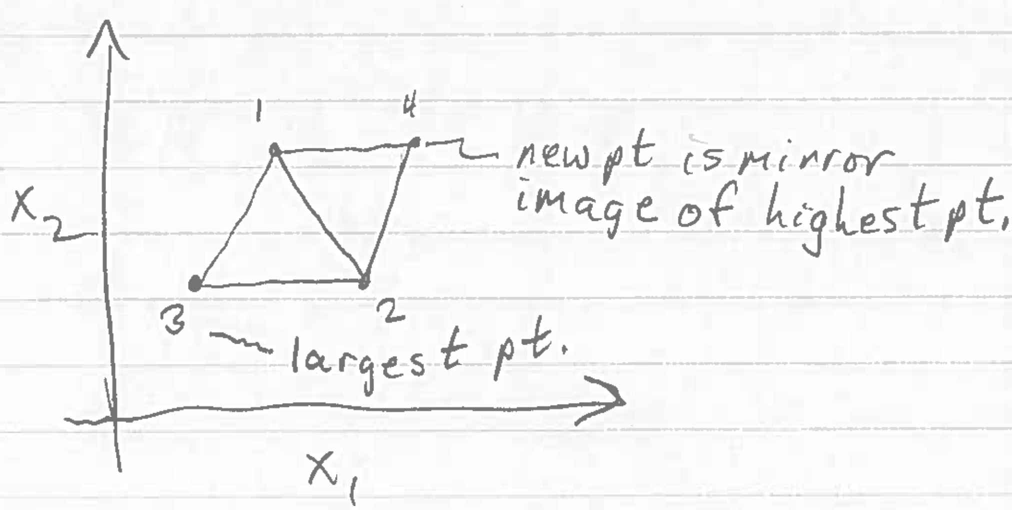
$$F(\underset{\sim}{x}^{(k)} + \alpha \underset{\sim}{p}) \text{ is minimized}$$

This is a one-D optimization problem!

Then return to step(1).

Simplex method:

A completely different algorithm is the simplex method. This algorithm uses a search based on triangles in 2-space and multi-dim. pyramids in n-space. We look at the 2-D problem:



we pick 3 pts in form of equilateral triangle. We discard the largest pt. (when looking for a minimum) and pick a new point which is its mirror image. We then wander through space until the min. is reached!

Technique works well if all elements of ∇F are comparable.

Tip: Rescale variables in an ill-cond. problem so that this is realized

Method is modified when minimum is approached (size of triangle is reduced.

Nelder-Mead implementation also changes triangle shape ⇒ makes triangle longer in the direction of the minimum.

Matlab function fminsearch uses this approach.

Constrained Optimization

So far, we have dealt w/ <u>unconstrained</u> optimization methods

Most of the time (except in regression problems) we have <u>constraints</u> on feasible solutions. How do we deal with this?

One-Dim.

In one D we may have inequality constraints (say $r > 0$). In this case we calculate the optimum in the <u>interior</u> and then compare to function on <u>bdy</u> of feasible soln region.

If F is lower on the <u>boundary</u>, then that point is the answer!

Multi-Dim.

- If N-dim., the problem is much more complex.

We can have either <u>equality constraints</u> or inequality constraints.

Look at equality constraints first

In general, we seek

$$\min_{\underset{\sim}{x}} F(\underset{\sim}{x}) \quad \text{subject to} \quad \underset{\sim}{g}(\underset{\sim}{x}) = 0$$

The <u>best</u> way of treating this is to use the m constraints $\underset{\sim}{g} = 0$ to <u>eliminate</u> m variables from $F(\underset{\sim}{x})$!

We did this with the soup can example:

$$F(x_1, x_2) = 2\pi x_1^2 + 2\pi x_1 x_2$$

$$\underset{\sim}{g}(\underset{\sim}{x}) = 2\pi x_1^2 x_2 - V = 0$$

we used this to eliminate x_2:

$$x_2 = \frac{V}{2\pi x_1^2}$$

$$F = 2\pi x_1^2 + \frac{V}{x_1}$$

So F is now an unconstrained 1-D problem!

Usually you can't get away with this. One approach is using Lagrange multipliers!

Let $F^* = F + \underset{\sim}{\lambda} \cdot \underset{\sim}{g}$

where $\underset{\sim}{\lambda}$ contains m multipliers for the m constraints $\underset{\sim}{g}(\underset{\sim}{x}) = 0$

The optimization problem was the solution to

$$\underset{\sim}{\nabla} F = 0$$

Here we have the augmented problem:

$$\underset{\sim}{\nabla}^* F^* = 0$$

$$\text{where } \underset{\sim}{\nabla}^* \equiv \begin{pmatrix} \underset{\sim}{\nabla}_{\underset{\sim}{x}} \\ \underset{\sim}{\nabla}_{\underset{\sim}{\lambda}} \end{pmatrix}$$

This is because

$$\nabla_{\tilde{\lambda}} F^* = \tilde{g}(\tilde{x}) = 0$$

or just the equality constraints!

Now for inequality constraints

we have: $\tilde{g}(\tilde{x}) \leq 0$

We can convert inequality constraints to equality constraints by adding slack variables

Let $\tilde{g} + \tilde{S} = 0$

where $S_i = x_{n+i}^2 \geq 0$

convenient choice insuring that $S_i \geq 0$

We then treat the problem using Lagrange multipliers again!

Finally, look at penalty functions

We wish to have an unconstrained optimization problem. We can do this even with constraints in an artificial manner. Suppose we have:

$$\min_{\underset{\sim}{x}} F(\underset{\sim}{x}) \quad , \quad \underset{\sim}{g}(\underset{\sim}{x}) = 0$$

We may define

$$F^*(\underset{\sim}{x}) = F(\underset{\sim}{x}) + P \, \| \underset{\sim}{g}(\underset{\sim}{x}) \|_2^2$$

where P is a positive number

We proceed by obtaining the solution for $\underset{\sim}{x}$ at moderate values of P, and then slowly increase it.

$P \to \infty$ corresponds to $\underset{\sim}{g}(\underset{\sim}{x}) \Rightarrow 0$!

This can also be used with inequality constraints through the use of slack variables.