Optimization

Alex Sverdlov
alex@theparticle.com

1 Introduction

The primary objective of optimization is to find the x where function $f(x)$ is minimized.

2 Direct Solution

Sometimes we're very lucky, and $f(x)$ just happens to be a known function. Such as $y = x^2 + 4x + 2$. If that's the case, we can take the derivative and set it to zero:

$$\frac{df}{dx} = 2x + 4 = 0$$

From that we get $x = -2$.

Now, this x could be a minimum or a maximum. One way to check is to find the 2nd derivative, which gets us 2. Positive number means it's a minimum (negative number indicates maximum).

3 Iterations

Outside of a direct solution, we can iterate over values of x, keeping track of minimum $f(x)$.

This raises a few issues: What if x is a real number, then iterating over integers may not yield the minimum. We can somewhat address this by iterating in tiny increments, such as 0.001.

Another issue is how far do we check. Let's say we indend to check all values of x between -10000 and $+10000$. But what if the minimum is at $x = 100000$?

This can be solved by a technique known as bracketing. Essentially before minimizing we find bounds, such as a, b in between which there's a minimum to be found.

Once the minimum (or other interesting values of $f(x)$ have been bracketed, is a linear search the best strategy? Most efficient methods utilize some form of divide and conquer, such as bisection or golden section.

4 Finding Roots

In the above direct solution, we set derivative to zero, and solved for x. This is root finding: we're looking for the x where $f(x)$ is zero. If we have access to the function we can use algebra to solve for x.

To find roots of a generic f (where we can't use algebra) we first need to bracket the root: we need to pick a and b such that $f(a) * f(b) < 0$.

Result less then zero indicates that $f(a)$ has different sign from $f(b)$: one is positive the other is negative—the function must pass through zero somewhere between a and b.

We find a and b by taking an initial guess, then extending a and b exponentially (double distance between a and b with each iteration) until the sign condition is satisfied. That will give us the bracket: a and b between which we're guaranteed to have a root.

What happens if the function doesn't have a root (never goes through zero). Then the bracketing step will fail as we run out of precision (e.g. if we represent x with a 32bit integer, then we'll fail at around -2 billion to 2 billion. Much bigger if we use doubles (though we might want to cut off the bracketing search if we go into crazy-numbers territory).

Once we have a and b we can solve for the root via bisection: we evaluate $f((a+b)/2)$, If it's zero, we found the root. If it's not zero, we replace either a or b with $(a+b)/2$, depending on the sign (if $f(a)$ and $f((a+b)/2)$ are both positive, we replace a with $(a+b)/2$, etc.)

If the root is bracketed (it is really there, between a and b), then bisection is guaranteed to succeed. The technique is very fast, as the search space halfs with each iteration.

Limitations are that we're finding a single root: Hopefully only one root has been bracketed.

5 Minimizing

To extend the above root-finding method to minimization we need to bracket the minimum. For the root, we needed a and b such that $f(a)$ has a different sign from $f(b)$. Now we need a triplet: a, b, c such that $a < b < c$ and $f(b) < f(a)$ and $f(b) < f(c)$.

The procedure to bracket the minimum is similar to bracketing roots: extend a and c out with every iteration, and checking midpoint $b = (a + c)/2$ for required inequalities.

If we know the minimum is within some locality, we can cut off the search early if the minimum isn't found.

The minimizing iteration loop is similar to bisection: Start with (a, b, c). Pick larger of the two intervals, a, b or b, c.

Pick point x within the bigger interval. If $f(x) > f(b)$ then the triplet condition is maintained. Replace the boundary point (e.g. if interval we used was a, b, then continue iteration with (x, b, c)).

If $f(x) < f(b)$ then we found a new minimum (better than our guess b). Continue iteration with (a, x, c).

5.1 Golden Section

Lets pretend that (a, b, c) is our initial guess, interval a, b is the largest, and b is almost exactly the minimum already. What should the guess for x be?

Obviously we could do $x = (a + b)/2$. But then our guess would be far away from b (which we mentioned was almost the minimum).

The golden ratio is 1.618. Which is often viewed as a ratio of two subsequent Fibonacci numbers. e.g.

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 \ldots$$

If we take any two consecutive Fibonacci numbers and find their ratio: e.g. $89/55 = 1.618$ we get the golden ratio.

Ignoring the 1 part, the golden section method picks x that is 0.382 of the interval length away from b within the larger interval.

The golden section converges quicker than splitting the larger interval in half.

6 Using Gradient

Another way to find the minimum is to use the derivative, or an estimate of a derivative.

To minimize $f(x)$, we may start with a random guess for x and then evaluate:

$$f'(x) \approx \frac{f(x + dx) - f(x)}{dx}$$

for some suitably tiny dx (perhaps 0.001) That will get us the slope (estimate) at x. We can then adjust x by some fraction of the negative slope (negative slope would push us downhill).

7 Line Minimization

Line minimization is a meta-algorithm that uses the single-variable (one-dimensional) minimizer as its building block. It allows us to minimize in a given direction in any number of dimensions.

Imagine we're trying to minimize $f(\boldsymbol{x})$, and our first guess at the minimum is: $\boldsymbol{x}$. The $\boldsymbol{x}$ in this case is a multi-dimensional point. We estimate the gradient of f at $\boldsymbol{x}$ to be $\boldsymbol{w}$ (normalized).

We can now define a function of a single variable z:

$$g(z) = f(\boldsymbol{x} - \boldsymbol{w} * z)$$

When $z = 0$ the output is $f(\boldsymbol{x})$. As z increases, the output moves away from $f(\boldsymbol{x})$ in the direction $-\boldsymbol{w}$, z units away.

We pass $g(z)$ to the single-variable minimizer, find the z that minimizes $g(z)$, then update $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \boldsymbol{w}_n * z$, and continue iterating until there's no more improvement.

8 Multi-dimensional optimization

The line minimization can be used as a core of a more general minimizer. The trick now is to figure out which direction to line-minimize in.

If we have access to the gradient, it's very obvious which direction to try to go into.

If we don't have access to the gradient, we can iterate through each dimension, optimizing each one. We can estimate the slope in each direction, etc. That unfortunately often causes us to zig-zag towards the minimum.

TODO,

For more details, google for Hessian matrix and conjugate gradient methods.

9 Stochastic gradient descent

Another idea is to randomly pick direction at every iteration. At every point, generate a random direction, and try to line-optimize in that direction. If succeed, use new point as next guess, otherwise pick another random direction.