Lecture 10.

Optimization

Material from Chapter 6 of the book Scientific Computing by Michael T. Heath.

One dimensional optimization. Given a strictly convex function on $[0,1]$ which has a minimum at the point a in the interval we want to locate a quickly. We take two points $0 < x_1 < x_2 < 1$ and examine $f(x_1)$ and $f(x_2)$. If $f(x_1) \geq f(x_2)$, the point can be found in $[x_1, 1]$ and we can drop the interval $[0, x_1]$. The other case is similar. We started with $0 < x_1 < x_2 < 1$ and now have $x_1 < x_2 < 1$, We fix the ratio of the three intervals as $1-\tau : 2\tau - 1 : 1-\tau$. The new interval is $[1-\tau, 1]$ of length τ with one the intermediate point at τ. Introduce x_3 such that $x_1 < x_2 < x_3 < 1$ and repeat the process. If we drop either end the new size of the whole interval will be τ. To preserve the same ratios, we need $2\tau - 1 = \tau(1-\tau)$ an then we can choose $x_3 = 1 - \tau(1-\tau)$. This requires $\tau = \frac{-1+\sqrt{5}}{2}$. Successive iteration locates the minimizing point geometrically fast.

A more rapid method is Newton's approximation for solving $f'(x) = 0$.

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

This is a bit unstable. But can be used if we are already reasonably close to the minimizing point and f'' is not too small. Then it is super fast! In any case optimization in one dimension, or along a line in higher dimension, is relatively easy.

Higher Dimensions. Steepest descent and variations.

The solution $x(t)$ of the ODE

$$\frac{dx}{dt} = -(\nabla f)(x(t))$$

has the property

$$\frac{df(x(t))}{dt} = -\langle (\nabla f)(x(t)), (\nabla f)(x(t)) \rangle$$

$-\nabla f(x(t))$ being the direction in which f falls off most rapidly. We may want to take

$$x_{k+1} = x_k - c(\nabla f)(x_k)$$

with the value of c is determined by performing a line (one dimensional) optimization. This is not quite efficient because it does not take into account the geometry provided by the Hessian $H(x) = \{f_{i,j}(x)\}$. For example if the function is a quadratic form $f(x, y) = x^2 + 8y^2$, the minimizing point is $(0, 0)$ and the level lines are ellipses. The direction $-\nabla f$ is the inward normal to the ellipse and it is not always pointing to the center $(0, 0)$. But $-H^{-1}\nabla f$ is. So the iteration

$$x_{k+1} = x_k - H(x_k)^{-1}(\nabla f)(x_k)$$

will do much better, but needs more computation. One can replace $H(x_k)$ by some positive definite B_k as an approximation and use

$$x_{k+1} = x_k - cB_k^{-1}(\nabla f)(x_k)$$

again optimizing over c.

One can periodically update B_k. If we have $y_k = (\nabla f)(x_k) - (\nabla f)(x_{k-1})$ and $s_k = x_k - x_{k-1}$, we have some information about the Hessian $H_{i,j}$. By Taylor approximation

$$Hs_k = y_k$$

We can update B_k by changing how it acts on s_k but leaving the rest as is. This leads to

$$B_{k+1} = B_k + \frac{y_k \otimes y_k}{\langle y_k, s_k \rangle} - \frac{B_k s_k \otimes B_k s_k}{\langle s_k, B_k s_k \rangle}$$

For any vector $v = \{v_i\}$, $v \otimes v$ is the rank 1 matrix $\{v_i v_j\}$.

Conjugate Gradient Method. The difference between steepest descent and the conjugate gradient method is that in the former the successive searches are in the directions $(\nabla f)(x_k)$ and are not well controlled. One could be searching in the same direction many times thereby slowing the process. Even when the function is quadratic the iteration will not reach the minimizer in a finite number of steps. This being a linear problem one would expect an exact solution in a finite number (n?) of steps. the conjugate gradient method achieves this.

We want to minimize $F(x) = \frac{1}{2}\sum a_{i,j}x_i x_j$. The minimum is at 0. We start out at an arbitrary initial value x_1 and define successively with

$$\begin{aligned} r_1 &= -Ax_1 \\ e_1 &= r_1 \end{aligned}$$

and for $i \geq 1$,

$$\begin{aligned} \alpha_i &= \frac{\|r_i\|^2}{\langle Ae_i, e_i\rangle} \\ x_{i+1} &= x_i + \alpha_i e_i \\ r_{i+1} &= r_i - \alpha_i A e_i \\ \beta_i &= \frac{\|r_{i+1}\|^2}{\|r_i\|^2} \\ e_{i+1} &= r_{i+1} + \beta_i e_i \end{aligned}$$

In the quadratic case with $(\nabla F)(x) = Ax$, it can be shown that the process will end in at most n steps with $r_n = x_n = 0$. If $F(x)$ is not quadratic then Ax gets replaced $(\nabla F)(x)$ and the process is restarted after n steps.

Nonlinear Regression.

We start with the problem fixing a regression of the form $y = f(t, \mathbf{x}) + r$ where t is the observed variable y is the variable to be predicted, f is the model, r is the noise and $\{x\}$ is a collection $\{x_i\}$ of d parameters to be determined form the training set of n values $\{y_i, t_i\}$. Computationally we want to minimize

$$F(x_1, x_2, \ldots, x_d) = \frac{1}{2}\sum_{i=1}^{n}[y_i - f(t_i, \mathbf{x})]^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2$$

over $\mathbf{x}$.

$$\nabla F = -\sum_{i=1}^{n} r_i (\nabla f)(t_i, \mathbf{x})$$

and

$$H(\mathbf{x}) = \sum_{i=1}^{n}[-r_i (Hf)(t_i, \mathbf{x}) + (\nabla f)(t_i, \mathbf{x}) \otimes (\nabla f)(t_i, \mathbf{x})]$$

making the iteration process

$$\mathbf{x}_{k+1} = \mathbf{x}_k - H^{-1}(\mathbf{x}_k)(\nabla F)(\mathbf{x}_k)$$

This makes for a difficult computation because n different Hessians have to be computed. One can expect $\{r_i\}$ to be small or there are cancellations to make $\sum_{i=1}^n r_i(Hf)(t_i, \mathbf{x})$ negligible in comparison to the second term.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\sum_{i=1}^n (\nabla f)(t_i, \mathbf{x}) \otimes (\nabla f)(t_i, \mathbf{x})]^{-1} \nabla F = \mathbf{x}_k - [\overline{H}]^{-1} \nabla F$$

If you think of $J = J_{i,\ell}(\mathbf{x})$ as the matrix $\frac{\partial f}{\partial x_\ell}(t_i.\mathbf{x})$ then $\overline{H} = J^*J$ is an approximation for H. We rewrite it as

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

and s_k solves

$$J^*Js_k = -J^*r(\mathbf{x}_k)$$

This is the solution to the linear regression or least square solution of the model

$$J(\mathbf{x}_k)s_k = -r(\mathbf{x}_k)$$

If the problem is degenerate then add $\mu_k I$.

Constrained Optimization. If we have to optimize subject to constrains then we use Lagrange multipliers. The Hessian is not positive definite any more. We are looking for a saddle point. Find the saddle points of

$$F(x_1, x_2, \ldots, x_d) + \sum_{i=1}^k \lambda_i g_i(x_1, \ldots, x_d)$$

to minimize $F(x_1, x_2, \ldots, x_d)$ subject to k conditions $g_i(x_1, \ldots, x_d) = 0$ for $i = 1, 2, \ldots, k$.

Linear programming.

Consists of optimizing a linear function over a set define by a bunch of linear equations and inequalities. Maximize

$$f(x) = \sum_i l_i x_i$$

subject to constraints of the form $Ax = b$ and $x \geq 0$. x has n components $Ax = b$ is m equations in the n unknowns. $m < n$. The set defined by these constraints is a convex set, a simplex. The vertices are the extreme points and the sup is attained at one of them. To find a vertex we can select $n - m$ of the variables $\{x_i\}$ and set them to zero. The resulting point may not be admissible. Then discard and choose a new set, till an admissible vertex is found. Then replace one variable at a time moving through admissible vertices till the maximum is reached.