Recitation 2

- Newton
- optimization
- least squares

Newton

$$\begin{bmatrix} x_1^2 \tanh(x_2)^{\cos(x_1)} \\ \sin(x_1)^4 / e^{x_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This is an example of a set of equations that would be basically impossible to solve by hand.

$$\begin{bmatrix} a x_1 + b x_2 \\ x_2 - 3 x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This on the other hand is linear, we have 2 equations and 2 unknowns and can just solve for with substitution.

x_k - current guess (root)

$r(x) = 0$

The point being made here is that we can't solve any r(x) = 0, but we can if it's linear, so what do we do? We take a first order Taylor series of r(x) about some current value x_k (giving us a linear approximation of r(x)), and solve that for when that is 0.

$r(x_k) \neq 0$

① $$r(x) = r(x_k) + \left(\frac{\partial r}{\partial x}\Big|_{x_k}\right)(x - x_k) = 0$$

Here we take the Taylor expansion and solve for when that equals 0, and set that new value to be x_{k+1}.

$$\frac{\partial r}{\partial x}\Big|_{x_k} (x - x_k) = -r(x_k)$$

$$x = x_k + \Delta x$$

Let's define delta x to be ^.

$$x_{k+1} = x_k - \left[\frac{\partial r}{\partial x}\Big|_{x_k}\right]^{-1} r(x_k)$$

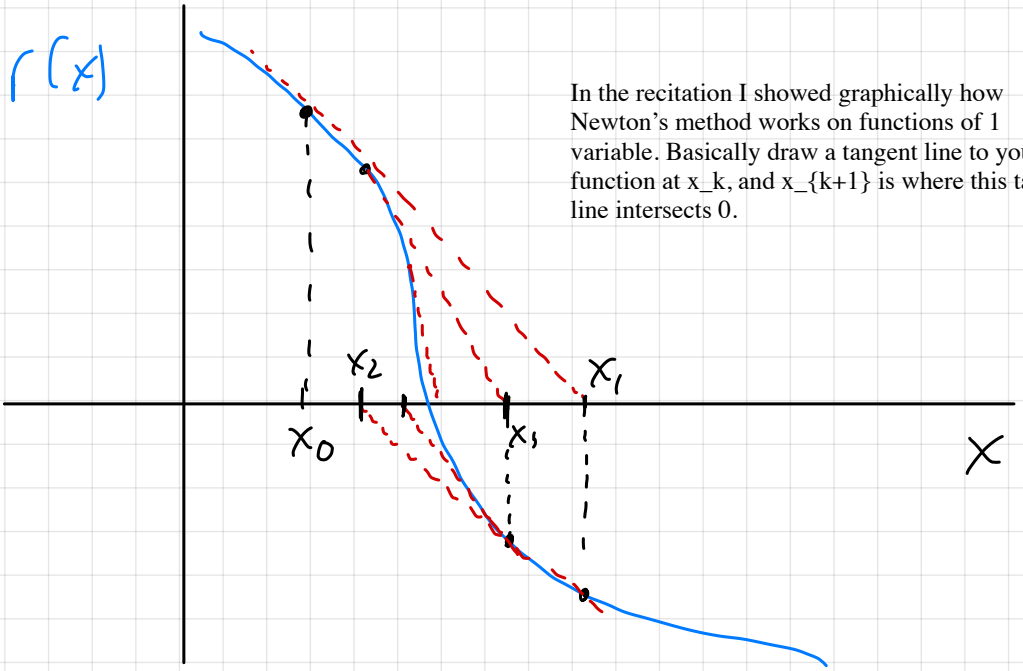
② $$r(x_k + \Delta x) = r(x_k) + \left(\frac{\partial r}{\partial x}\Big|_{x_k}\right)\Delta x = 0$$

Now I can rewrite the Taylor series using this delta notation, and just solve for the delta x (this is identical to above). This is how you will see Zac always do it.

$$\Delta x = -\left(\frac{\partial r}{\partial x}\Big|_{x_k}\right)^{-1} r(x_k)$$

$$x_{k+1} = x_k + \alpha \Delta x$$

α: step size / learning rate



In the recitation I showed graphically how Newton's method works on functions of 1 variable. Basically draw a tangent line to your function at x_k, and x_{k+1} is where this tangent line intersects 0.

quadratic convergence $e_{k+1} = e_k^2$

Newton's method has "quadratic convergence" meaning the error gets squared at every iterate. This only happens once you are in the "basin of attraction"

Optimization

$\min_x \; f(x)$

KKT: $\nabla_x f(x) = 0$

For an unconstrained optimization problem, we have an optimality condition that says the gradient of the objective function wrt x must be 0.

$r(x) = \nabla_x f(x) = 0$

This is just another rootfinding problem where the "residual function" before is just the gradient of f. We can solve this by just doing normal Newton.

$$\Delta x = -\left(\frac{\partial \nabla_x f(x)}{\partial x}\Big|_{x_k}\right)^{-1} \nabla_x f(x_k)$$

$\nabla_x^2 f(x_k)$

The Jacobian of the gradient of f(x) is the hessian of f(x). This means the Newton step is computed as the following. This is why you hear people talk about how Newton is forming a "quadratic approximation" of the cost function and minimizing that.

$$\Delta x = -\nabla_x^2 f(x_k)^{-1} \nabla_x f(x_k)$$

$$\min_{\Delta x} \; f(x_k) + (\nabla_x f(x_k))^T \Delta x_k + \frac{1}{2} \Delta x_k^T (\nabla_x^2 f(x_k)) \Delta x$$

(g = $\nabla_x f(x_k)$, H = $\nabla_x^2 f(x_k)$)

$$\Delta x = -H^{-1} g$$

Here is that "quadratic approximation" demonstrated. If we take a second order Taylor series of f(x) and minimize it, we get the equivalent Newton step.

Quadratic forms

$$J(x) = \frac{1}{2} x^T Q x + q^T x$$

$$\nabla_x J(x) = Qx + q$$

$$\nabla_x^2 J(x) = Q$$

matrixcalculus.com

We love this form of quadratic functions because we can take the derivatives of it really easily. We should always try and reduce things to this form when possible. matrixcalculus.com is a good website for playing around with these derivatives.

Least squares

Least squares problems for "skinny" or overdetermined linear systems. This means there are more equations than there are unknowns, so we can't solve for Ax=b. Instead, we will solve for the Ax \approx b, and find the x that gets us as close as possible.

$$\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} x \end{bmatrix} \approx \begin{bmatrix} b \end{bmatrix}$$

Matrix manipulation

$$(ABC)^T = C^T B^T A^T$$

$$x^T A^T b = b^T A x$$

$$(Ax - b)^T = x^T A^T - b^T$$

Here are 3 matrix manipulation tricks we will need to use in the following least squares examples.

$$\min_x \; \frac{1}{2} \|Ax - b\|_2^2 = \frac{1}{2} (Ax-b)^T (Ax-b)$$

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

$$= \frac{1}{2} (x^T A^T A x - b^T A x - x^T A^T b + b^T b) \quad \text{(NO } x\text{)}$$

($x^T A^T b$ is a transpose of $b^T A x$)

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

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

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

($Q = A^T A$, $q = A^T b$)

We rewrite the cost function and use our matrix manipulation tricks until we get our favorite quadratic form.

$$\min_x \; \frac{1}{2} \|Ax - b\|_2^2 = \min_x \; J(x) = \frac{1}{2} x^T (A^T A) x - (A^T b)^T x$$

$$\nabla_x J(x) = A^T A x - A^T b = 0$$

Setting the gradient of this cost function to 0 gives us the "normal equations", now we can solve.

$$A^T A x = A^T b$$

$$x = (A^T A)^{-1} A^T b$$

pseudo inverse for skinny A

(2)

$$\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}$$

Now let's look at "fat" or underdetermined linear systems where we have more variables than we do equations. This means we are able to satisfy Ax=b, but we want to find the x with the smallest L2 norm that satisfies Ax=b.

$$\min_x \quad \frac{1}{2}\|x\|_2^2$$

x: primal variable

λ: dual variable

$$\text{s.t.} \quad Ax - b = 0$$

This is an equality constrained optimization problem with a convex cost function and linear constraints.

$$L(x, \lambda) = \frac{1}{2} x^T x + \lambda^T (Ax - b)$$

Form the Lagrangian and KKT conditions for optimality.

KKT:

$$\nabla_x L(x, \lambda) = x + A^T \lambda = 0 \quad \text{Stationarity}$$

$$Ax - b = 0 \quad \text{primal feasibility}$$

(1)

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

Since these equations are linear in x and lambda, we can just solve a linear system for them.

② $x = -A^T\lambda \implies Ax = b$

$-AA^T\lambda = b$

$\lambda = -(AA^T)^{-1}b$

Alternatively, we can just substitute and get the following:

$$x = A^T(AA^T)^{-1}b$$

$\min_x \; f(x)$ $\quad x$: primal

s.t. $\quad C(x) = 0$ $\quad \lambda$: dual

Here let's write down a generic equality-constrained optimization problem where we don't assume anything about these functions.

$$L(x,\lambda) = f(x) + \lambda^T C(x)$$

$$\nabla_x L = \nabla_x f(x) + \left(\frac{\partial C}{\partial x}\right)^T \lambda = 0$$

$$C(x) = 0$$

KKT conditions

Same KKT conditions as before (but more general since we wrote grad f and dc/dx).

$$r\left(\begin{bmatrix} x \\ \lambda \end{bmatrix}\right) = \begin{bmatrix} \nabla_x L(x,\lambda) \\ C(x) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We can't just solve this anymore (since it's not linear in x and lambda) so we turn to our favorite method for rootfinding (Newton's method).

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \nabla_x^2 L(x,\lambda) & \left(\frac{\partial C}{\partial x}\right)^T \\ \frac{\partial C}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} -\nabla_x L(x_k,\lambda_k) \\ -C(x_k) \end{bmatrix}$$

"full" Newton

Just by following our original Newton's method, here is the Newton step for this.

$$\nabla_x^2 L(x,\lambda) = \nabla_x^2 f(x) + \frac{\partial}{\partial x}\left(\left(\frac{\partial c}{\partial x}\right)^T \lambda\right)$$

Constraint curvature

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \nabla_x^2 f(x) & \left(\frac{\partial c}{\partial x}\right)^T \\ \frac{\partial c}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} -\nabla_x L(x_k, \lambda_k) \\ -c(x_k) \end{bmatrix}$$

Gauss-Newton Step

The hessian of the Lagrangian has this term in red that we call the "constraint curvature term". There are two things to be concerned about with this term:

1. It can be very expensive to compute, we are differentiating a Jacobian wrt a vector, so we get a 3rd order tensor.

2. Usually in control we choose f(x) so that it's nice and convex, making the hessian of f(x) easy to compute and positive definite. This constraint curvature term can introduce negative eigenvalues into the hessian of the Lagrangian which makes Newton's method trickier to implement (since we would have to regularize to ensure a descent direction)

Because of this, oftentimes in control we simply omit the constraint curvature term, and treat the hessian of the Lagrangian as if it were just the hessian of the cost function. This is called "Gauss-Newton", and it can take a little longer to converge.