1/26 Recitation

- derivatives
- Newton's method
- constrained optimization

$f : \mathbb{R}^N \rightarrow \mathbb{R}^M$

func. name   N inputs   M outputs

$y = g(a, b)$ $g : (\mathbb{R}^{N_a}, \mathbb{R}^{N_b}) \rightarrow \mathbb{R}^{N_y}$

This means f takes in a vector of length N, and outputs a vector of length M

$y = f(x)$, $y \in \mathbb{R}^M$

$x \in \mathbb{R}^N$

partial derivatives: ∂ \del d

$\dot{y} = \frac{dy}{dt}$, $\frac{\partial y}{\partial x}$

Here is the Jacobian, where all of the partial derivatives are collected and numbers of rows = number of outputs of f, and numbers of columns = number of inputs. There is only one interpretation of this.

matrix of scalars

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_N} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_N} \\ \vdots & & & \\ \frac{\partial y_M}{\partial x_1} & \cdots & & \frac{\partial y_M}{\partial x_N} \end{bmatrix}$$

rows = # outputs = M

cols = # inputs = N

$y = g(x)$ $x \in \mathbb{R}^N$, $y \in \mathbb{R}$

As a convenience, we define the gradient to be the partial derivatives of a scalar-output function to be a column vector, making it the transpose of the jacobian. This is a very important distinction, and only applies to scalar-output functions.

$$\frac{\partial g}{\partial x} = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \cdots & \frac{\partial g}{\partial x_N} \end{bmatrix} \in (1 \times N) \text{ row vector}$$

gradient $\nabla_x g = \left(\frac{\partial g}{\partial x}\right)^T \in \mathbb{R}^N$ vector hessian

$$\nabla_x^2 g = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_N} \\ \vdots & & & \\ \frac{\partial^2 g}{\partial x_N x_1} & & & \frac{\partial^2 g}{\partial x_N^2} \end{bmatrix} \in \mathbb{R}^{N \times N}$$

symmetric matrix

Hessian is all of the second partial derivatives, or the jacobian of the gradient of a function. It is symmetric for all the sorts of functions that we care about in optimization, and is positive semi definite for convex functions.

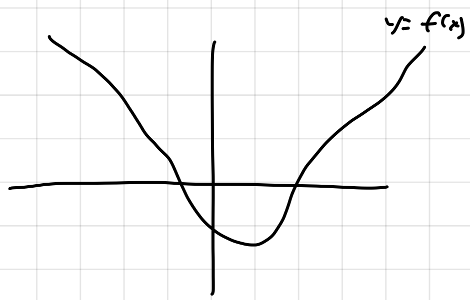
Taylor series $\quad y = f(x), \quad x \in \mathbb{R}^N, \; y \in \mathbb{R}^m$

$$\underset{(m,1)}{f(x)} \approx \underset{(m,1)}{f(\bar{x})} + \underset{(m,N)}{\left[\frac{\partial f}{\partial x}\Big|_{\bar{x}}\right]} \underset{(N,1)}{(x - \bar{x})}$$

Here we see that the jacobian convention makes sense (size wise) in a taylor series.

$$g(x) \approx g(\bar{x}) + (\nabla_x g(\bar{x}))^T (x - \bar{x})$$

Newtons Method



When it comes to solving equations, if there is a solution and the equations are linear, we can solve them. This is the one tool we have that will always work. If the equations are nonlinear, often we have to resort to numerical methods.

We formulate these problems as rootfinding problems, which just means if we want to find f(x) = y, we just solve for the roots of f(x) - y = 0.

To do this iteratively, we resort to our one tool, linear systems, and we linearize this function and solve for the root of the linearized version. It turns out that if our initial guess is within some "basin of attraction" of the solution, linearizing and solving iteratively is extremely fast effective. This is Newton's method.

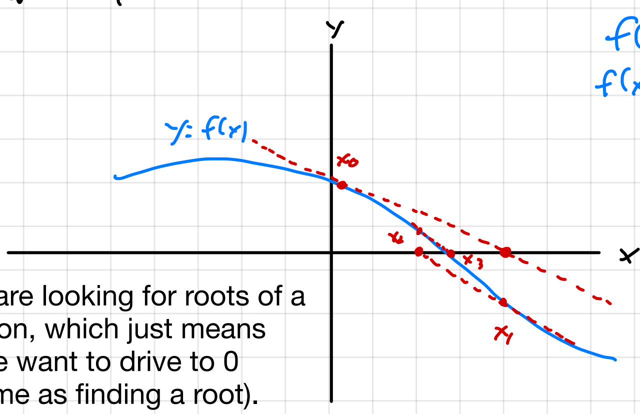
$$5a + 6b = 7$$
$$6a - 4b = 11$$

if a solution exists I can find it

$$e^a + \cosh(b) = \sin(a)$$
$$a^2 + \sin(b)\, a^e = -1$$

not easy...

$$f(x) = y$$
$$f(x) - y = 0 \leftarrow$$ find "roots"



Oftentimes, we are looking for roots of a "residual" function, which just means it's a function we want to drive to 0 (which is the same as finding a root).

root finding

$$f(x) = 0 \qquad y \approx f(x_0) + \left(\frac{\partial f}{\partial x}\Big|_{x_0}\right)(x - x_0) = 0$$

solve for root of our linearized version

Solve for root of our linearized version

$$y \approx f(x_0) + \left(\frac{\partial f}{\partial x}\bigg|_{x_0}\right)(x - x_0) = 0$$

$$f(x_0) = -\left(\frac{\partial f}{\partial x}\bigg|_{x_0}\right) x - x_0$$

$$\left(-\frac{\partial f}{\partial x}\bigg|_{x_0}\right)^{-1} f(x_0) = x - x_0$$

$$x = x_0 - \left(\frac{\partial f}{\partial x}\bigg|_{x_0}\right)^{-1} f(x_0)$$

$$\boxed{x_{k+1} = x_k - \left(\frac{\partial f}{\partial x}\bigg|_{x_k}\right)^{-1} f(x_k)}$$

We approximate our residual function with a first order taylor series (making our approximate function linear), then we solve for the root of this linearized function. This is an iteration of Newton's method.

$$x = x_k + \Delta x$$

I can define delta x to be this

$$\cancel{x_k} + \Delta x - \cancel{x_k} = \Delta x$$

$$f(x_k) + \left(\frac{\partial f}{\partial x}\bigg|_{x_k}\right)(x - x_k)$$

$$f(x_k) + \left(\frac{\partial f}{\partial x}\bigg|_{x_k}\right)\Delta x = 0$$

Which let's me write out the Newton step as the following equivalent expression:

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

$$x_{k+1} = x_k + \alpha \overbrace{\Delta x}^{\text{"Newton step"}}$$

$$= x_k + \alpha \left(-\frac{\partial f}{\partial x}\bigg|_{x_k}\right)^{-1} f(x_k)$$

I can apply a step length (alpha) to this step. If my linearization is accurate, alpha=1 is ok, if it is inaccurate, we probably have to use a smaller alpha. We can use a linesearch to find an appropriate alpha to use (more on this later).

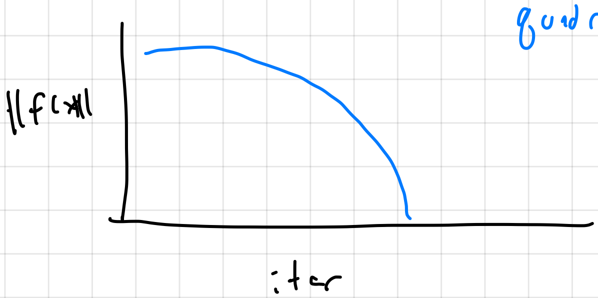
input : $f(x)$, x_0,

Newton's method has quadratic convergence, this means once we are in the basin of attraction, the error will square itself each iteration. (1e-2 -> 1e-4 -> 1e-8 -> 1e-16). This is really, really fast (in terms of iteration count) convergence for anything.

use newton

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

solved when $\|f(x_k)\| <$ tolerance (1e-3, 1e-6, 1e-15)



quadratic convergence

$$e_{k+1} \approx e_k^2$$

$$\min_x \quad f(x)$$

We can easily use Newton's rootfinding method for optimization. First we can take an unconstrained optimization problem, write out the optimality condition (gradient = 0), and then rootfind on this condition. This means our residual function is the gradient of f.

$$\nabla_x f(x) = 0$$ ← optimality condition

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

If we write out our Newton step, we see that we are taking the jacobian of our gradient, which is the hessian of the original function. This is why you always see people in optimization talk about how Newton's method is really approximating the objection function as a quadratic, and solving for the minimum of this quadratic. This is a really imporant idea and it's worth spending some time and convincing yourself of this.

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

$$= x_k - \alpha \left(\nabla_x^2 f(x_k) \right)^{-1} g(x_k)$$

Newton for unconstrained opti

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

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

Notice how if we approximate our objective function with it's second order taylor expansion, then minimize this approximation, we get the exact same Newton step.

$$\nabla_x f(x_k) + (\nabla_x^2 f(x_k)) \Delta x = 0$$

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

Root finding = linearize + solve

$$\min_x \quad f(x)$$

$$\text{s.t.} \quad c(x) = 0$$

Newtons method for rootfinding can also be used for constrained optimization in the exact same way. Write out the optimality conditions as a root finding problem -> use Newton to solve this rootfinding problem.

Our optimality conditions use lagrange multipliers to enforce the constraints, so we are solving for both the variable x (primal variable) and the lagrange multipliers lambda (dual variable).

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

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

$$c(x) = 0$$

optimality (KKT)

$$z = \begin{bmatrix} x \\ \lambda \end{bmatrix} \quad r(z) = \begin{bmatrix} \nabla_x f(x) + \left(\frac{\partial c}{\partial x}\right)^T \lambda \\ c(x) \end{bmatrix} = 0$$

Stacking these conditions into a residual function like this lets us use Newton normally.

$$z = \begin{bmatrix} x \\ \lambda \end{bmatrix} \qquad r(z) = \begin{bmatrix} \nabla_x f(x) + \left(\frac{\partial c}{\partial x}\right)^T \lambda \\ c(x) \end{bmatrix} = 0$$

When we are doing Newtons steps on a constrained optimization problem, we see that the hessian of the lagrangian shows up. We can form this (especially with modern autodiff tools), but it's expensive and can sometimes ruin the nice positive definiteness of our cost hessian. This leads us to a choice for Newton steps when it comes to constrained optimization:

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

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = -\begin{bmatrix} \boxed{\nabla_x^2 L(x_k, \lambda_k)} & \left(\frac{\partial c}{\partial x}\Big|_{x_k}\right)^T \\ \frac{\partial c}{\partial x}\Big|_{x_k} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_x L(x_k, \lambda_k) \\ c(x_k) \end{bmatrix}$$

Full Newton: Do Newton as normal (form the hessian of the lagrangian)

Gauss Newton: Do Newton mostly as normal, but in the jacobian of our residual (the matrix we form), ignore the constraint curviture and just assume hess(lag) = hess(cost).

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

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

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

$$\begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = -\begin{bmatrix} H & \left(\frac{\partial c}{\partial x}\Big|_{x_k}\right)^T \\ \frac{\partial c}{\partial x}\Big|_{x_k} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_x L(x_k, \lambda_k) \\ c(x_k) \end{bmatrix}$$

for full Newton: $H = \nabla_x^2 f(x_k) + \frac{\partial}{\partial x}\left(\left(\frac{\partial c}{\partial x}\Big|_{x_k}\right)^T \lambda_k\right) = \nabla_x^2 L$

Gauss-Newton: $H = \nabla_x^2 f(x_k)$

Constraint Curvature: $\frac{\partial}{\partial x}\underbrace{\left(\underbrace{\left(\frac{\partial c}{\partial x}\Big|_{x_k}\right)^T \lambda_k}_{(N,1)}\right)}_{(N,N)}$

$\lambda \in \mathbb{R}^m$

$c(x) \in \mathbb{R}^m$

$x \in \mathbb{R}^N$