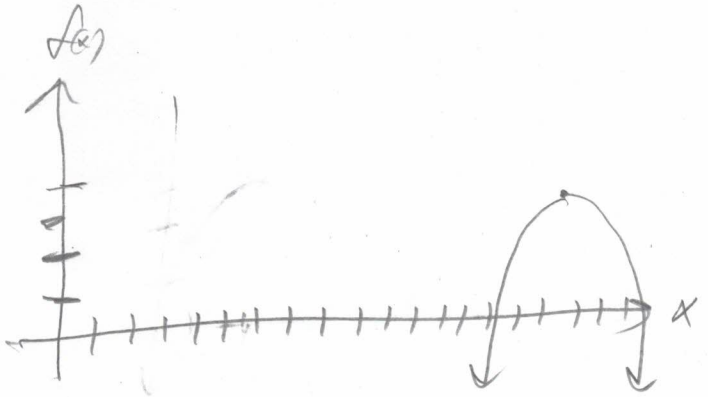
MATH 340/640 LEC 23. (FINAL!!)

Recall: let $f(x) = 3 - (x-17)^2$



$f_* = \max_{x \in \mathbb{R}} \{f(x)\} = 3$

$x_* = \operatorname{argmax}_{x \in \mathbb{R}} \{f(x)\} := \{x : f(x) = f_*\} = \{x : f(x) = 3\} = \{17\}$

How to find x_*? Set $f'(x) = 0$ and solve. Include if $f''(x_*) < 0$.

$f'(x) = -2(x-17) = 0 \Rightarrow x - 17 = 0 \Rightarrow x_* = 17$, $f''(x) = -2$ ✓

Note argmax is unaffected by strictly increasing functions of x. E.g. if

e.g. $g(x) = a f(x)$ s.t. $a > 0 \Rightarrow \operatorname{argmax}\{f(x)\} = \operatorname{argmax}\{g(x)\}$

e.g. Also $g(x) = \ln(f(x))$

If $f(x)$ is a PDF, or $p(x)$ is a PMF for rv X

$$\text{Mode}(X) := \operatorname*{argmax}_{x \in S_X}^{\text{cont}} \{f(x)\} = \operatorname*{argmax}_{x \in S_X}^{\text{dd}} \{p(x)\}$$

$$\operatorname*{argmax}_{x \in S_X} \{h(x)\} = \operatorname*{argmax}_{x \in S_X} \{\ln(h(x))\}$$

Finding modes is important in estimation scenarios (MATH 341)

Also, it is a general useful problem called "optimization". Big field!

e.g. $X \sim N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$

$\propto e^{-\frac{1}{2\sigma^2}x^2 + \frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2}}$

$\propto e^{-\frac{1}{2\sigma^2}x^2 + \frac{\mu}{\sigma^2}x} = h(x) \Rightarrow \ln(h(x)) = \frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2$

$\text{Mode}[X] = \underset{x}{\text{argmax}} \left\{ \underbrace{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2}_{g(x)} \right\}$

Since X is continuous...

$\Rightarrow g'(x) = \frac{\mu}{\sigma^2} - \frac{1}{\sigma^2}x \overset{\text{set}}{=} 0 \Rightarrow x_0 = \mu$

Note: we skipped Poisson process, Skellam distr

End of Course

Let $\vec{X}$ be a vector rv with entries $\vec{\mu}$, Σ matrix

$\text{Mode}(\vec{X}) = \underset{\vec{x} \in S_{\vec{X}}}{\text{argmax}} \left\{ \ln\left(k_{\vec{X}}(\vec{x})\right) \right\}$ HARDER!

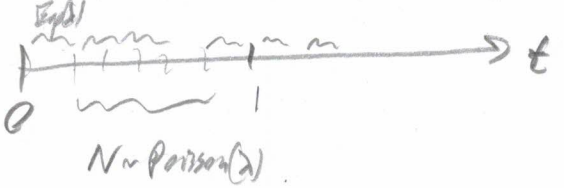
MATH 292 gives you tools to assess this. Under some conditions but in the real world, those conditions don't hold.

In fact, usually it is *impossible* to assess, so instead it is approximated as best as possible.

Especially since if $\dim(\vec{X})$ is large. In modern statistical models this can be in the 1000's. Chat GPT has $\dim(\vec{X}) = 100$ trillion!

$T_k \sim \text{Erlang}(k, \lambda)$, $N \sim \text{Poisson}(\lambda)$

$P(T_k > 1) = P(N \leq k-1) = Q(k, \lambda)$



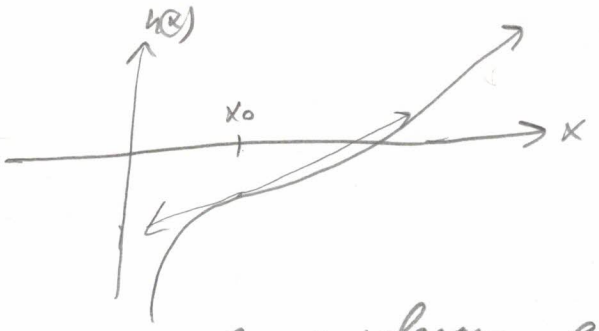
regularized upper incomplete Gamma function

the Newton-Raphson algorithm

Let's start with an approximation algorithm in one dimension

Let $g(x) = \ln(k(x))$ which is assumed differentiable

let $h(x) := g'(x)$. We want to solve for $h(x) = 0$.



Step 0: Define ε, a tolerance e.g. $\varepsilon = .01$ (where the solution will not differ from the approx sol. by ε.)

Step 1: Guess the solution at $x = x_0$

Step 2: Draw tangent line at $h(x_0)$, call it $l(x_0)$

Solve for x where $l(x_0) = 0$ (i.e. find its x intercept)

and call its solution x_1. $\Rightarrow y - y_0 = m(x - x_0) \Rightarrow y = mx - mx_0 + y_0 \overset{\text{set}}{=} 0$

$\Rightarrow mx = mx_0 - y_0 \Rightarrow x_1 = x_0 - \frac{y_0}{m} = x_0 - \frac{h(x_0)}{h'(x_0)}$

Step 3: Repeat Step 2, until $|x_i - x_{i-1}| < \varepsilon$ (where $x_0 = x_1$) — "update rule"

e.g. $X \sim \text{Beta}(\underset{\alpha}{17}, \underset{\beta}{37})$. $\text{Mode}(X) = \frac{\alpha - 1}{\alpha + \beta - 2} = \frac{17-1}{17+37-2} = .30769$

Try Newton Raphson. $f(x) = \frac{1}{B(17,37)} x^{17-1}(1-x)^{37-1} \Rightarrow k(x) = x^{16}(1-x)^{36}$, $g(x) = \ln(k(x)) = 16\ln(x) + 36\ln(1-x)$

$\Rightarrow h(x) = \frac{16}{x} - \frac{36}{1-x}$

$\Rightarrow h'(x) = -\frac{16}{x^2} - \frac{36}{(1-x)^2}$

Step 1: good guess: $x_0 = 0.5$

Step 2₁: $h'(x_0) = -\frac{16}{0.5^2} - \frac{36}{(0.5)^2} = -208$

$h(x_0) = \frac{16}{.5} - \frac{36}{.5} = -19$

$\Rightarrow x_1 = 0.5 - \frac{-19}{-208} = 0.4519$ $\quad |x_1 - x_0| = .048 \not< \varepsilon$ continue

Step 2₂: $h'(x_1) = -\frac{16}{.4519^2} - \frac{36}{(1-.4519)^2} = -130.215$

$h(x_1) = \frac{16}{.4519} - \frac{36}{(1-.4519)} = -19.8.10$

$\Rightarrow x_2 = .4519 - \frac{-30.215}{-198.10} = .2991$

$|x_2 - x_1| = .152 \not< \varepsilon$ continue

Step 2₃: $h'(x_2) = -252.1$ $\quad x_3 = .2991 - \frac{2.122}{-252.1} = .30755$

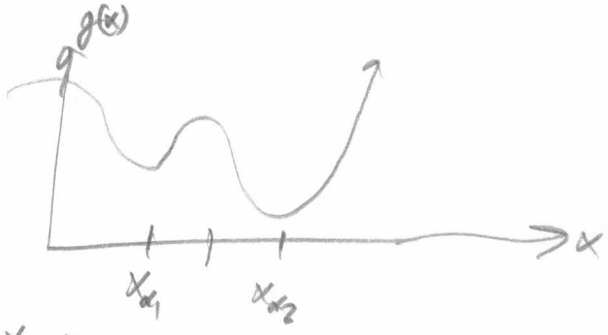
$h(x_2) = 2.122$

$(x_3 - x_2) = .008 < \varepsilon \Rightarrow$ STOP

Very close to true answer!!

Advantage: we never overshoot the minimum by stepping "too far"!

Major pitfall: if function has multiple minimums: $x_{\alpha_1}, x_{\alpha_2}$ are being later than the other, $g(x_{\alpha_2}) < g(x_{\alpha_1})$, different values of $x_0 \Rightarrow$ different answers.



$x_0 \Rightarrow$ Converges to x_{α_1} ← | → $x_0 \Rightarrow$ Converges to x_{α_2}

Other problems: slow! It can get faster if you use 2nd derivative of h as well

Also, sometimes you need to examine the function itself by hand $\Rightarrow$ HARD

Can this be extended to dimensions greater than 1? Yes.

Let $g(\vec{x}) := h(h(\vec{x})) \in \mathbb{R}^n$

Now, the "derivative" of g is derivative of each dimension called the "gradient"

$$h(\vec{x}) := \vec{\nabla} g(\vec{x}) := \begin{bmatrix} \frac{\partial g}{\partial x_1}\Big|_{\vec{x}} \\ \vdots \\ \frac{\partial g}{\partial x_n}\Big|_{\vec{x}} \end{bmatrix} \in \mathbb{R}^n$$

What is the analogue of the $h'(\vec{x})$ here? This is called the Hessian

$$H(\vec{x}) := \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2}\Big|_{\vec{x}} & \frac{\partial^2 g}{\partial x_1 \partial x_2}\Big|_{\vec{x}} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_n}\Big|_{\vec{x}} \\ \vdots & & & \\ \frac{\partial^2 g}{\partial x_n \partial x_1}\Big|_{\vec{x}} & \frac{\partial^2 g}{\partial x_n \partial x_2}\Big|_{\vec{x}} & \cdots & \frac{\partial^2 g}{\partial x_n^2}\Big|_{\vec{x}} \end{bmatrix}$$

Step 0: ε decided Step 1: x_0 decided

Step 2: Now our update step becomes

$$\vec{x}_1 = \vec{x}_0 - H(\vec{x}_0)^{-1} \vec{\nabla} h(\vec{x}_0)$$

Step 3: continue until $|\vec{x}_i - \vec{x}_{i-1}| < \varepsilon$.

All the problems in the 1-d case apply here and are a lot worse! Additionally: the Hessian is expensive to compute and sometimes is estimated from data and can be inverted!

$g(\vec{x}) := -\ln(h(\vec{x}))$. So how to find $\operatorname{argmin}(h(\vec{x}))$ we find $\operatorname{argmin}\{g(\vec{x})\}$

Is there a less expensive algorithm? How about nothing from defining ε as step magnitude and using an update like: → AKA "learning rate"

$$\vec{x}_i = \vec{x}_{i-1} - \varepsilon \vec{\nabla} h(\vec{x}_{i-1})$$

↑ descend ↑ in the direction of the gradient ⇒ gradient descent

If ε is constant throughout the iterations, we can move too slow or we can move too fast

Stochastic Gradient Descent: when we use data to evaluate $\vec{\nabla} h(\vec{x})$, we don't use all the data, we take a random subset ⇒ better performance and better speed