Line Search (choosing stepsizes intelligently for GD)

Recall for GD that $-\nabla f(x)$ points towards the minimizer:

$$f(x_*) \geq f(x) + \langle \nabla f(x), x^* - x \rangle$$

$$\Rightarrow \quad 0 \geq \langle \nabla f(x), x^* - x \rangle \qquad (\text{b/c } f(x_*) \leq f(x))$$

$$\Rightarrow \quad \langle -\nabla f(x), x^* - x \rangle \geq 0$$

Similarly true for subgradient steps.

So far we've been choosing stepsizes α_t based on specific knowledge of the function (e.g. if f is β-smooth, use $\alpha_t = \alpha = \frac{1}{\beta}$ because this will converge)

Line search chooses step-sizes in a smarter, adaptive (but more expensive) way that can decrease the total cost of optimization

Idea: minimize along the ray of possible step sizes

$$\alpha_t = \underset{s \geq 0}{\operatorname{argmin}} \; f(x_t - s \nabla f(x_t))$$

This is exact line-search and α_t is called the Curry steplength

This could be a cheap optimization problem that has a closed form. If not, we can use inexact line searches.

The basic idea of inexact line search is that locally f is approximated well by a linear function:

$$f(x) = \underbrace{f(x_t) - \langle \nabla f(x_t), x - x_t \rangle}_{\text{maximum possible improvement}} + O(\|x - x_t\|_2^2)$$

so if x is close to x_t then I can achieve

$$(*) \quad f(x) < f(x_t) - c\langle \nabla f(x_t), x - x_t \rangle$$

for a given $c < 1$. That is, we can achieve some portion of the decrease predicted by the linear estimate of f.

Given x_t and $\nabla f(x_t)$, inexact line searches try to find a large α that satisfies $(*)$, the <u>sufficient decrease</u> condition

We will use backtracking line search

Given: x_t, d_t, and $c \in (0,1)$, $\beta \in (0,1)$

↑ search direction ($-\nabla f(x_t)$)
descent direction

$\alpha \leftarrow 1$

while $f(x_t + \alpha d_t) > f(x_t) + c\alpha \langle \nabla f(x_t), d_t \rangle$

$\alpha \leftarrow \beta \cdot \alpha$

return α

Note that the cost of backtracking depends on:

- the cost of evaluating f
- how much decrease you ask for (b/c c determines how often you will loop before achieving sufficient decrease)
- the aggressiveness of your backtracking, β

Note that we want to balance aggressiveness and decrease asked for with the nonlinearity of the function so that we make maximal progress per call to BTLS

(Boyd & Vandenberghe) General prescriptions:

$$c \in [.01, .3]$$

$$\beta \in [.5, .8]$$

Gradient Descent w/ Exact Line Search
(on μ-strongly convex and β-smooth convex functions)
converges linearly:

$$f(x_T) - f(x_*) \leq \left(1 - \frac{1}{K}\right)^{T-1} (f(x_1) - f(x^*))$$

$$= e^{\left[\log\left(1 - \frac{1}{K}\right)(T-1)\right]}$$

where $K = \frac{\beta}{\mu}$ is the condition number of the convex function

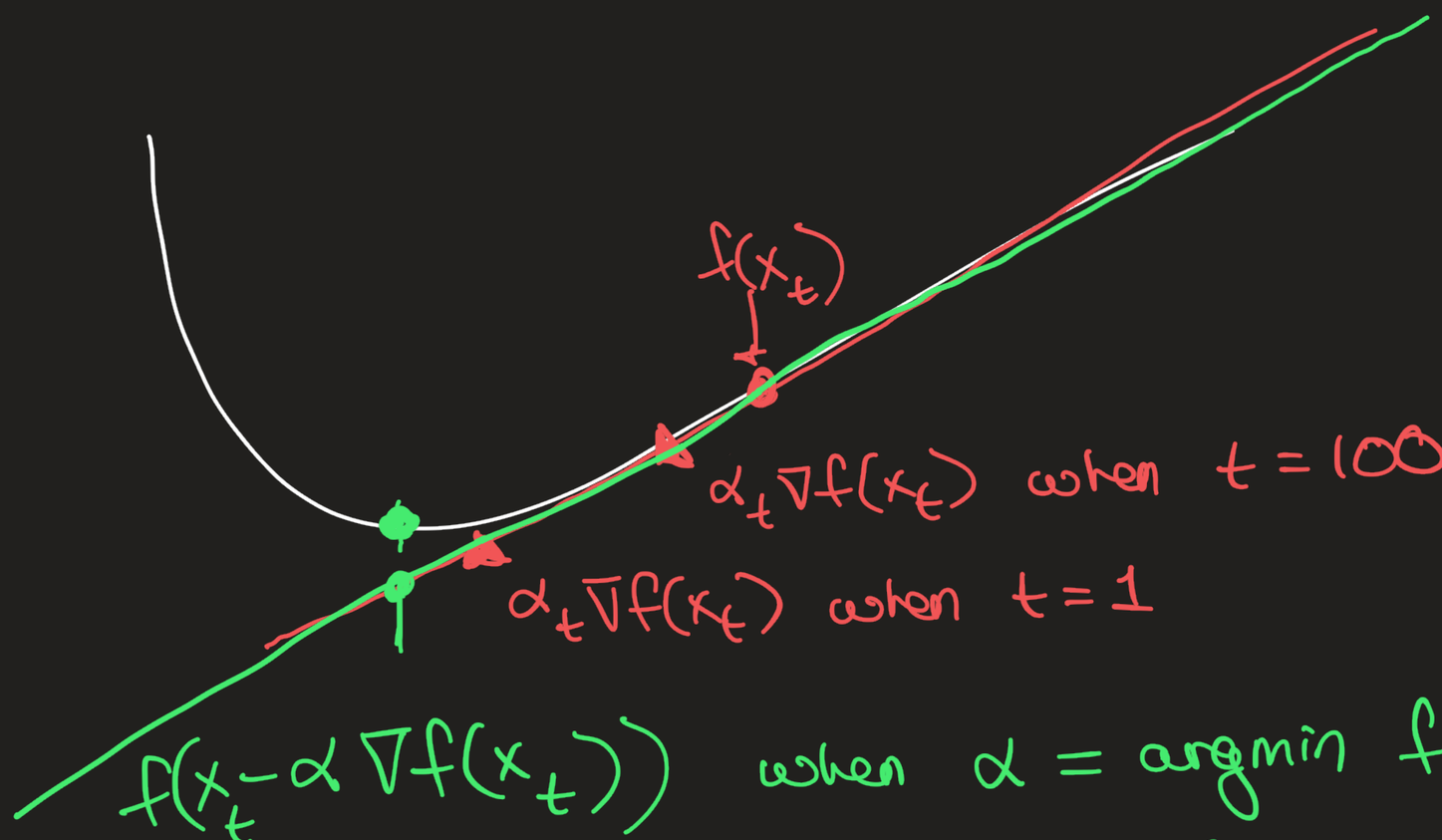
β = an upper bound on the eigenvalues of the Hessian of f

μ = a lower bound "

Compare that to gradient descent

$$f(x_T) - f(x_*) \leq O\left(\frac{\log(T)}{T}\right)$$

Gradient Descent + Line search = "Steepest Descent Method"



$$\alpha_t = \frac{1}{\mu t}$$

Newton's Method

We saw that line search works well when $K = \frac{\beta}{\mu}$ is small, which says f looks like a quadratic.

Idea: line search helps us adapt to the curvature of our function when K is small

We can explicitly model f's curvature:

$$f(x) \approx \underbrace{f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{1}{2}(x - x_t)^T \nabla^2 f(x_t)(x - x_t)}_{= m(x)}$$

and move in the direction that minimizes this quadratic model.

Note that f is convex $\Rightarrow$ $\nabla^2 f(x_t)$ is PSD

$\Rightarrow m(x)$ is convex

so

$$x_{t+1} = \underset{x}{\operatorname{argmin}}\ m(x)$$

occurs where $\nabla m(x_{t+1}) = 0$

$$\Rightarrow \quad 0 = \nabla f(x_t) + \nabla^2 f(x_t)(x_{t+1} - x_t)$$

$$\Rightarrow \quad x_{t+1} = x_t - \underbrace{\nabla^2 f(x_t)^{-1}}\ \nabla f(x_t)$$

this term accounts for our estimate of the curvature of f at x_t

So this suggest we can use $\nabla^2 f(x_t)^{-1} \nabla f(x_t)$ as our search direction

As before we can linesearch to choose how far to move in that direction

Damped/guarded Newton's method

Given: x_1

For $t = 1, \dots, T-1$

$d_t = -\nabla^2 f(x_t)^{-1} \nabla f(x_t)$

Choose α_t by using backtracking LS

$x_{t+1} = x_t + \alpha_t d_t$

output: x_T

What is the benefit of incorporating curvature by using $\nabla^2 f(x_t)^{-1} \nabla f(x_t)$ as our search direction?

There are two phases of the algorithm:

- damped phase: suboptimality goes down by a constant factor at every iteration
- purely Newton phase: at each iteration, the number of digits of accuracy doubles

$$f(x_t) - f(x_*) \leq \varepsilon \quad \text{in} \quad O\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)$$

steps

Drawbacks:

- need to form $\nabla^2 f(x_t) \in \mathbb{R}^{d \times d}$ and solve a linear system

In general this costs $O(d^2)$ memory
$O(d^3)$ operations

so if d is large (100,000) this is not feasible.

- sometimes problem structure allows us to reduce these costs
- sometimes we can use Quasi-Newton methods that cheaply approximate $\nabla^2 f(x_t)^{-1}$
- inspires faster algorithms than gradient descent & sgd that try to keep track of curvature information.