Ch 10.4: optimization and steepest descent (gradient descent) and Newton's method for minimization

Tuesday, September 23, 2025 2:19 PM

See Newton vs grad. desc. Saddle pt. demo

Today, we'll look at multivariate optimization in its simplest setting

↳ i.e. no constraints, everything is differentiable

* $\min_{\vec{x}\in\mathbb{R}^n} g(\vec{x})$, $g:\mathbb{R}^n\to\mathbb{R}$

going beyond this requires a full optimization course. Details on Lagrange multipliers...

$g:\mathbb{R}^n\to\mathbb{R}^m$, $m>1$, isn't well-defined. Studied in "multi-objective" optimization

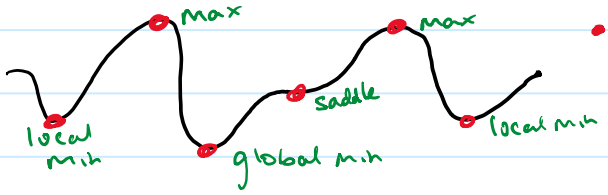
It's related to root finding:

all solutions (i.e. $\vec{x}^* \in \arg\min_{\vec{x}} g(\vec{x})$) are roots of the gradient

$\nabla g(\vec{x}^*) = 0$ } definition of a "stationary point"

∇g is our F from previous sections, $\nabla g:\mathbb{R}^n\to\mathbb{R}^n$

it's more complicated than root-finding because not all stationary points are (global) minimizers



• all stationary

Solving

We'll set $\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha\cdot\vec{v}$ for some direction $\vec{v}$.

How to pick $\vec{v}$? Pick the steepest direction.

i.e. the "fall line" if you are skiing.

i.e. the direction $\hat{v}$ $\left(=\frac{\vec{v}}{\|\vec{v}\|}\right)$ of greatest decrease in g if you were to take a tiny step:

$$\lim_{h\to 0} \frac{g(\vec{x}^{(k)} + h\hat{v}) - g(\vec{x}^{(k)})}{h} =: D_{\hat{v}} g(\vec{x}^{(k)})$$ definition of DIRECTIONAL DERIVATIVE

$$= \hat{v}^T \cdot \nabla g(\vec{x}^{(k)})$$ via chain rule

* What about maximization problems? Convert $\max_{\vec{x}} g(\vec{x})$ into $-\min_{\vec{x}} -g(\vec{x})$ "WLOG"

Tuesday, September 30, 2025 4:10 PM

want $\hat{v}^T \cdot \nabla g(\vec{x}^{(k)})$ to be as negative as possible, (and $\hat{v}$ a unit vector), so $\hat{v} = -\frac{\nabla g(\vec{x}^{(k)})}{\|\nabla g(\vec{x}^{(k)})\|_2}$

(why? $|\hat{v}^T \cdot \nabla g(\vec{x}^{(k)})| \le \underbrace{\|\hat{v}\|_2}_{=1} \cdot \|\nabla g(\vec{x}^{(k)})\|_2$ via Cauchy-Schwarz

so $\hat{v}^T \cdot \nabla g(\vec{x}^{(k)}) \ge -\|\nabla g(\vec{x}^{(k)})\|_2$ is best possible,

and choosing $\hat{v} = -\frac{\nabla g(\vec{x}^{(k)})}{\|\nabla g(\vec{x}^{(k)})\|_2}$ achieves this.

other variations of "steepest descent"

Incidentally, if we change what we mean by a "unit vector", i.e., say $\|\hat{v}\|_1 = 1$ not $\|\hat{v}\|_2 = 1$, then we'll get a different answer. For $\|\hat{v}\|_1 = 1$, we get

$\hat{v} = -\text{sign}((\nabla g)_j) \cdot \vec{e}_j$ where j is the entry of $\nabla g(\vec{x}^{(k)})$ w/ largest magnitude.

Or, for $\|\hat{v}\|_\infty = 1$, we get $\hat{v} = -\text{sign}(\nabla g(\vec{x}^{(k)}))$.

Proof of optimality via Hölder's inequality)

So... $\vec{x}^{(k+1)} = \vec{x}^{(k)} - \alpha \cdot \frac{\nabla g(\vec{x}^{(k)})}{\|\nabla g(\vec{x}^{(k)})\|_2}$ for some $\alpha > 0$ ("steepest descent")

↑ TBD. "stepsize" or "learning rate"

or, "absorb" $\frac{1}{\|\nabla g(\vec{x}^{(k)})\|_2}$ into α since we have to find α anyway

$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \alpha_k \cdot \nabla g(\vec{x}^{(k)})$ ("gradient descent")

↑ still TBD, possibly allow α_k to change every iteration

⚠ May not converge for some α

EXAMPLE: $g(\vec{x}) = \frac{1}{2}\vec{x}^T C \vec{x} + \vec{d}^T \vec{x}$, a quadratic, $C = C^T$ symmetric

$\nabla g(\vec{x}) = C\vec{x} + \vec{d}$ is affine, gradient descent is $\vec{x}^{(k+1)} = \vec{x}^{(k)} - \alpha(C\vec{x} + \vec{d})$

This will converge if $\alpha \cdot \nabla g$ is a contraction, eg. $\|I - \alpha C\|_2 < 1$ (spectral norm ("natural"/"induced" by ℓ^2 norm))

"extra," skip in lecture: If C isn't positive definite, there's no $\alpha \ge 0$ to make $I - \alpha C$ a contraction, but $\alpha = \frac{1}{\|C\|_2}$ will work if $C \ge 0$ (proof beyond scope of our class)

Tuesday, September 30, 2025 6:07 PM

Connection to fixed pt. equations

Solve $\nabla g(\vec{x}) = \vec{0}$

ie. $F(\vec{x}) = \vec{0}$, root finding

→ recast as fixed pt. problem:

$$\eta \cdot F(\vec{x}) = \vec{0}$$

$$\underbrace{\vec{x} - \eta \cdot F(\vec{x})}_{G(\vec{x})} = \vec{x}$$

$G(\vec{x}) = \vec{x}$ fixed pt.

So iterate $\vec{x}^{(k+1)} = G(\vec{x}^{(k)})$

$= \vec{x}^{(k)} - \eta \cdot F(\vec{x}^{(k)})$

$= \vec{x}^{(k)} - \eta \cdot \nabla g(\vec{x}^{(k)})$

GRADIENT DESCENT!

ref: Nocedal and Wright textbook "Numerical Optimization"

How to choose the stepsize?

No single "best" way for all scenarios. Some options:

(1) A single fixed value, based on theory, or trial-and-error,
or (if data are normalized well) $\alpha = 0.001$ for neural nets ☺

(2) A predetermined Schedule, often heuristic (eg. cosine or exp. rates for neural nets)

ex: $\alpha_k = \frac{\alpha_1}{k}$ so $\alpha_k \to 0$

often desirable if you have noisy gradients

(3) Via exact linesearch

$$\alpha_k = \underset{\alpha > 0}{\operatorname{argmin}} \underbrace{g(\vec{x}^{(k)} - \alpha \cdot \nabla g(\vec{x}^{(k)}))}_{\varphi(\alpha)}$$

(4) Via an approximate linesearch

(4a) Sample a few values $y_i = g(\vec{x}^{(k)} - \alpha_i \nabla g \vec{x}^{(k)}) = \varphi(\alpha_i)$
and build a simple "surrogate" model $\tilde{\varphi}(\alpha)$ from this data
(usually an interpolating polynomial, as we'll discuss later this semester)
and do an exact linesearch on $\tilde{\varphi}(\alpha)$

(4b) backtrack: Start w/ a large α, and decrease until some criteria are met (ex: Armijo conditions, Goldstein conditions, Wolfe conditions)

(4c) Hybrid: can get complicated.

Tuesday, September 30, 2025 6:19 PM

Differences w/ root-finding

For optimization, we have a loss function, so that gives a natural criteria for things (like a stepsize)

In root-finding, you don't have that.

For $F(\vec{x})=0$, you could use $\|F(\vec{x})\|_2^2$ as a criterion, but that means you implicitly cast it as a minimization problem!

More on this ("nonlinear least squares") next lecture

Newton's Method for optimization

At iteration $\vec{x}^{(k)}$, we'll form a model m_k of $g(\vec{x})$ based on its 2nd order Taylor Series:

$$m_k(\vec{x}) = \underbrace{g(\vec{x}^{(k)})}_{\text{scalar}} + \underbrace{\nabla g(\vec{x}^{(k)})^T}_{\text{vector}} \cdot \underbrace{(\vec{x}-\vec{x}^{(k)})}_{\text{vector}} + \tfrac{1}{2}(\vec{x}-\vec{x}^{(k)}) \cdot \underbrace{\nabla^2 g(\vec{x}^{(k)})}_{\text{call this } H_k,\ \text{the Hessian}} \cdot (\vec{x}-\vec{x}^{(k)})$$

($\nabla^2 g(\vec{x}^{(k)})$: $n \times n$ matrix)

$$\vec{x}^{(k+1)} = \underset{\vec{x}}{\operatorname{argmin}}\ m_k(\vec{x})$$

it's a quadratic, and we can solve this* in closed form (it's a variant of the "normal equations") by setting $\nabla m_k(\vec{x})=0$ and solving for $\vec{x}$

Aside: "argmin" is location of min value

ex: $2 = \min_x (x-3)^2+2$

$3 = \operatorname{argmin}_x (x-3)^2+2$

*well, true if H_k is positive semidefinite!

$\nabla m_k(\vec{x}) = \nabla g(\vec{x}^{(k)}) + H_k \cdot (\vec{x}-\vec{x}^{(k)})$

so setting this equal to zero yields... $\vec{x} = \vec{x}^{(k)} - H_k^{-1} \nabla g(\vec{x}^{(k)})$

So NEWTON'S METHOD for optimization is

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \underbrace{H_k^{-1} \cdot \nabla g(\vec{x}^{(k)})}_{\text{solve via } LDL^T \text{ or, if } H_k \succeq 0 \text{ (eg. } g \text{ is convex) use Cholesky. Matlab's backslash does it well}} \quad \text{where } H_k = \nabla^2 g(x_k) \in \mathbb{R}^{n \times n}$$

(H_k: always symmetric)

practical methods add a stepsize $\eta \le 1$ and do a backtracking linesearch starting at $\eta = 1$.

and just like we have QUASI-NEWTON for root-finding, we have QUASI-NEWTON for optimization, replacing H_k with something easier to compute and invert. Most well-known versions are BFGS (and it's variant L-BFGS for when $n \gg 100$) and SR1

Tuesday, September 30, 2025 8:20 PM

(probably skip this in lecture)

One more example

$$g(\vec{x}) = \frac{1}{2}\|A\vec{x}-\vec{b}\|_2^2, \quad \text{(linear) least squares}$$

$$= \frac{1}{2}\vec{x}^T A^T A\vec{x} - \vec{b}^T A\vec{x} + \frac{1}{2}\vec{b}^T\vec{b}$$

A^TA: C, Gram matrix

a special case of $g(\vec{x}) = \frac{1}{2}\vec{x}^T C\vec{x} + \vec{d}^T\vec{x}$ w/ $C = A^TA$, $d = -A^T\vec{b}$

Recall from APPM 3310, $C \geq 0$ always and $C > 0$ if A has linearly independent columns.

$C \geq 0$ means $g(\vec{x})$ is a convex function and setting $\nabla g(\vec{x}) = \vec{0}$ will return the global minimizer

i.e. solve normal equations

since $\nabla g(\vec{x}) = A^TA\vec{x} - A^T\vec{b}$

so solve $A^TA\vec{x} = A^T\vec{b}$

(via LDL^T or Cholesky)

like LU for symmetric matrices

Gradient descent for this problem is

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \alpha \cdot \underbrace{A^T(A\vec{x}^{(k)} - \vec{b})}_{\nabla g(\vec{x}^{(k)})}, \quad \text{will converge if } \alpha < \frac{2}{\|A^TA\|_2}$$

Solution is unique iff $A^TA > 0$

Sometimes this is advantageous over LDL^T or Cholesky

eg. if A is very sparse

(though we'll later see even better methods like conjugate gradient ...)