

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 simpler setting

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

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

↳ i.e. no constraints,
everything is differentiable

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

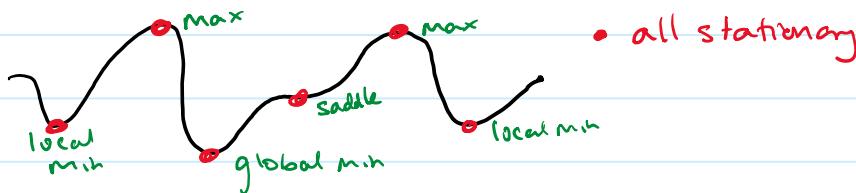
It's related to root finding:

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

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

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

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



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 $\hat{\vec{v}} (= \frac{\vec{v}}{\|\vec{v}\|})$ of greatest decrease in g if you were to take a tiny step:

$$\lim_{h \rightarrow 0} \frac{g(\vec{x}^{(k)} + h \hat{\vec{v}}) - g(\vec{x}^{(k)})}{h} =: D_{\hat{\vec{v}}} g(\vec{x}^{(k)})$$

definition of DIRECTIONAL DERIVATIVE
via chain rule

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

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

want $\hat{v}^T \nabla g(\vec{x}^{(k)})$ to be as negative as possible,

(and \hat{v} a unit vector), so $\hat{v} = -\frac{\nabla g(\vec{x}^{(k)})}{\|\nabla g(\vec{x}^{(k)})\|_2}$

$$\|\nabla g(\vec{x}^{(k)})\|_2$$

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

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

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

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

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

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

Proof of optimality via Hölder's inequality)

other variations of "steepest descent"

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

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

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - \alpha_k \cdot \nabla g(\vec{x}^{(k)}) \quad \begin{matrix} \text{"gradient descent"} \\ \uparrow \text{TBD, possibly allow } \alpha_k \text{ to change every iteration} \end{matrix}$$

⚠ 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, e.g. $\|I - \alpha C\|_2 < 1$

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

spectral norm
("natural"/"induced"
by ℓ^2 norm)

Connection to fixed pt. equations

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

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

→ recast as fixed pt. problem:

so convergence
proofs are all
about contractions
just like for
fixed pt. iteration

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

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

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

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

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

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

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 (e.g. cosine or exp. rates for neural nets)

$$\text{ex: } \alpha_k = \frac{\alpha_1}{k} \quad \text{so } \underbrace{\alpha_k \rightarrow 0}_{\text{often desirable if you have noisy gradients}}$$

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 $y_i = g(\vec{x}^{(k)} - \alpha_i \cdot \nabla g(\vec{x}^{(k)})) = \varphi(\alpha_i)$

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.

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 of $g(\vec{x})$ based on its 2nd order Taylor Series:

$$m_k(\vec{x}) = \underbrace{g(\vec{x}^{(k)})}_{\text{Scalar}} + \underbrace{\nabla g(\vec{x}^{(k)})^\top}_{\text{Vector}} \cdot (\underbrace{\vec{x} - \vec{x}^{(k)}}_{\text{Vector}}) + \frac{1}{2} (\underbrace{\vec{x} - \vec{x}^{(k)}}_{\text{Vector}})^\top \underbrace{\nabla^2 g(\vec{x}^{(k)})}_{\substack{n \times n \text{ matrix} \\ \text{call this } H_k, \\ \text{the Hessian}}} \cdot (\underbrace{\vec{x} - \vec{x}^{(k)}}_{\text{Vector}})$$

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

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}

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

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

*well, true if H_k is positive semidefinite!

Aside: "argmin" is location of min value

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

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

so NEWTON'S METHOD for optimization is

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} - H_k^{-1} \cdot \nabla g(\vec{x}^{(k)}) \quad \text{where } H_k = \nabla^2 g(\vec{x}_k) \in \mathbb{R}^{n \times n}$$

\rightarrow solve via LDL^\top or, always symmetric

if $H_k \geq 0$ (e.g. g is convex)

use Cholesky. Matlab's backslash does it well

and do a backtracking linesearch starting at $\gamma = 1$.

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 its variant L-BFGS for when $n \gg 100$) and SR1

Newton's Method for optimization is Newton's method for root-finding

i.e. just apply Newton (root-finding)
to $\nabla g(\vec{x}) = \vec{0}$

$$\text{so } \vec{x} \leftarrow \vec{x} - \underbrace{\nabla g(\vec{x})^{-1}}_{\text{Jacobian of gradient}} \cdot \nabla g(\vec{x})$$

-- is the same as $\nabla^2 g(\vec{x})^{-1}$ is the Hessian

$$\vec{x} \leftarrow \vec{x} - \nabla^2 g(\vec{x})^{-1} \cdot \nabla g(\vec{x})$$

One more example

(probably skip this in lecture)

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

$$= \frac{1}{2} \vec{x}^T \underbrace{A^T A}_{C, \text{ Gram matrix}} \vec{x} - \vec{b}^T A \vec{x} + \frac{1}{2} \vec{b}^T \vec{b}$$

a special case of
 $g(\vec{x}) = \frac{1}{2} \vec{x}^T C \vec{x} + \vec{d}^T \vec{x}$

$$\text{w/ } C = A^T A \\ d = -A^T \vec{b}$$

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

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

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

(via LDL^T or Cholesky)

like LU for symmetric matrices

Gradient descent for this problem is

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

solution is unique

iff $A^T A > 0$

Sometimes this is advantageous over LDL^T or Cholesky
e.g. if A is very sparse

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