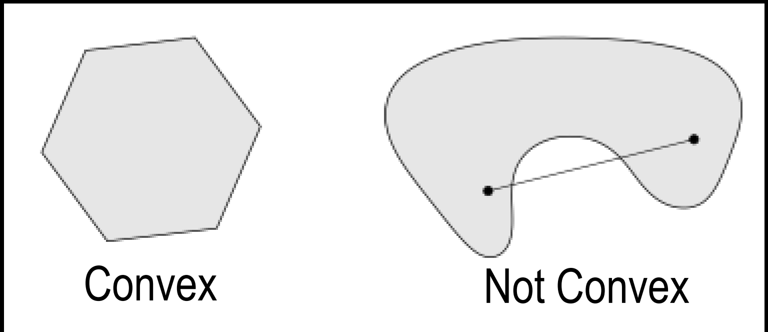
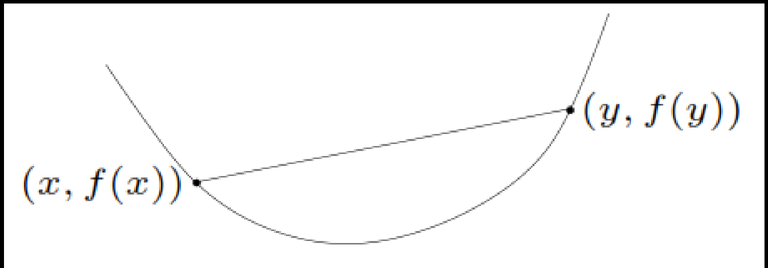
# Unconstrained Optimization

Using material from Stephen Boyd and Geoff Gordon

# Last time

- We saw how to define convex sets and functions
- We saw common convexity-preserving operations

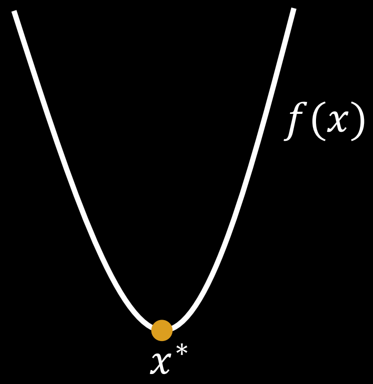



- Today we will use convexity to help us solve problems!

# Unconstrained Minimization Problem

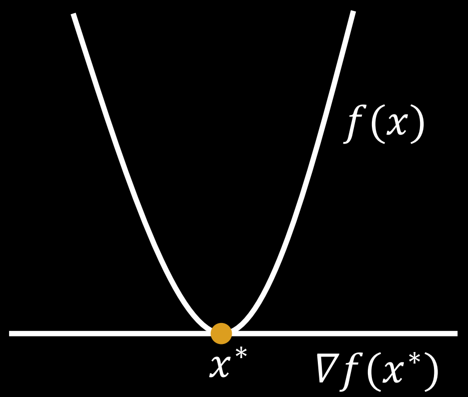
$$\underset{x}{\text{minimize}}\, f(x)$$




- Assumptions
  - *f* is convex
  - No constraints on x
- There are MANY methods for this kind of problem
  - Some are general, some exploit a specific structure of *f*
- Usually decide what to use based on
  - Differentiability of *f*
  - How quickly you can compute $\nabla f(x)$
- We will cover several important methods common in robotics

# Review: Minimizing a simple function

- For a simple function, e.g. $f(x) = x^2 - 4x$, we can use calculus directly to find the minimum
- For an optimal point $x^*$, $\nabla f(x^*) = 0$
- So,
  1. Take the derivative of $f(x)$
  2. Set it equal to 0
  3. Solve for $x$




- Example for $f(x) = x^2 - 4x$
  1. $\nabla f(x) = 2x - 4$
  2. $2x - 4 = 0$
  3. $x = 2$ is the minimum

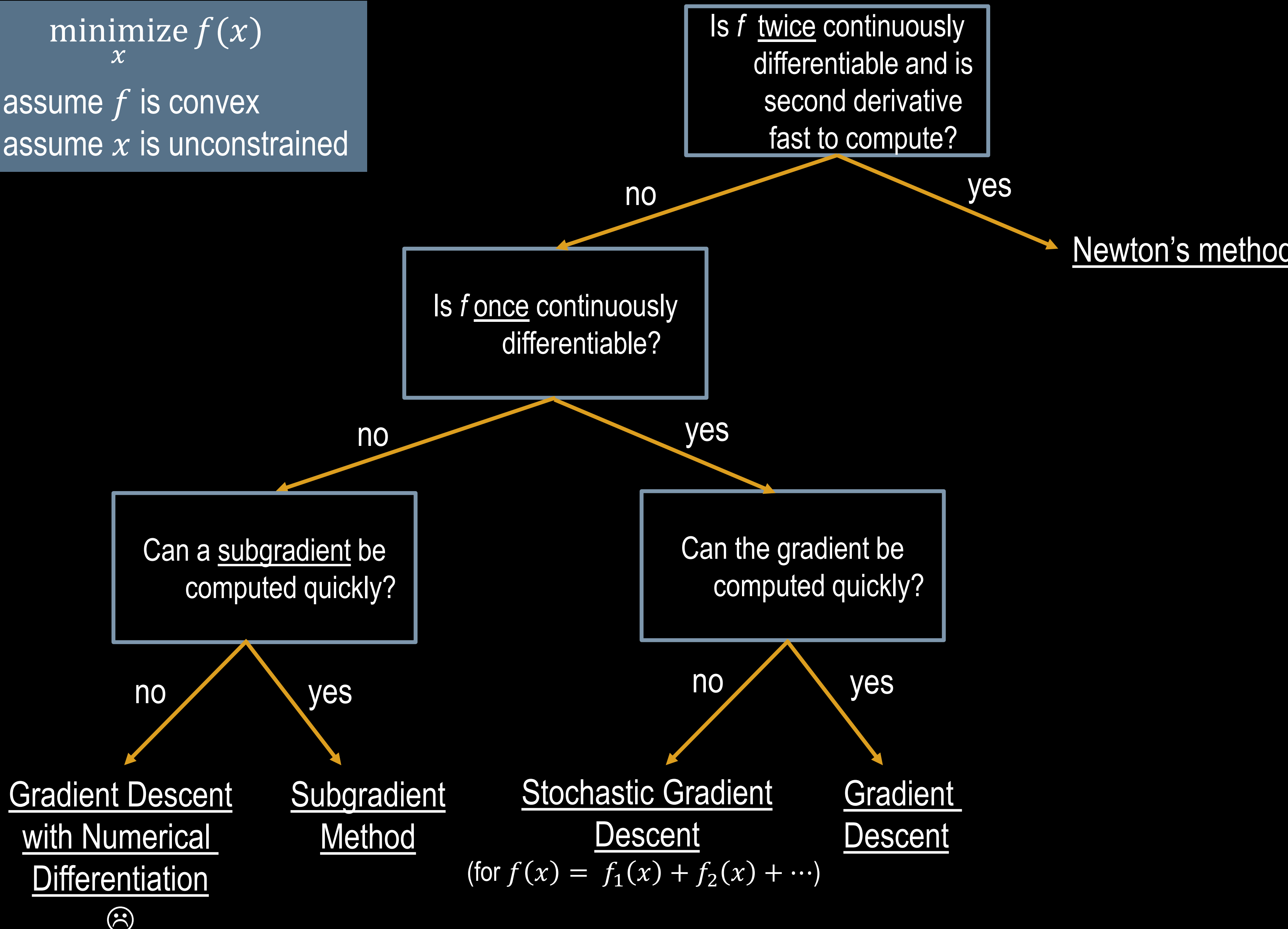
# What about a more complicated function?

- $f(x) = e^{0.5x+0.9} + e^{-0.5x^2-0.4} + 4x$
  1. $\nabla f(x) = 0.5e^{0.5x+0.9} - xe^{-0.5x^2-0.4} + 4$
  2. $0.5e^{0.5x+0.9} - xe^{-0.5x^2-0.4} + 4 = 0$
  3. $x = ???$

Problem: No way to solve arbitrary equations using algebra!

$$\underset{x}{\text{minimize}}\ f(x)$$

- assume $f$ is convex
- assume $x$ is unconstrained

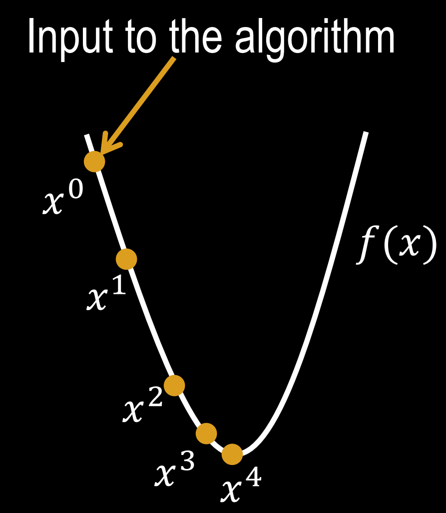



*Many more methods not covered!

# Descent Methods

# Unconstrained Minimization Methods

- Let *p** be the optimal value of *f(x)*
- Let *x** be a value of *x* that produces *p**
  - $p^* = f(x^*)$
- These methods produce a sequence of points:

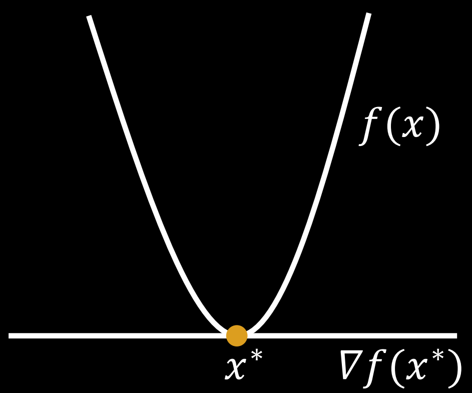



$$x^{(k)} \in \mathbf{dom}\, f,\ k = 0, 1, \ldots$$

$$f(x^{(k)}) \to p^\star$$

- Can interpret as iteratively finding an x* that solves optimality condition:

$$\nabla f(x^\star) = 0$$



# Descent methods

- We will cover two types of descent methods:
  - Gradient descent
  - Newton's method
    - Advantage: affine invariant
- Descent methods generate points with this property:

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

Other notation:

$$x := x + t\Delta x$$

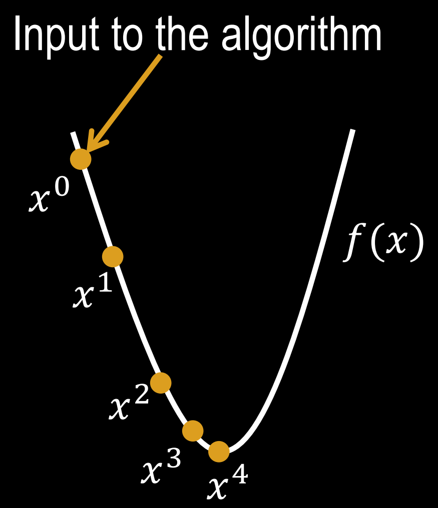
- $\Delta x$ is the step, or search direction
- $t$ is the step size, or step length

# General descent algorithm

Many ways to do these

**given** a starting point $x \in \mathbf{dom}\, f$.
**repeat**
1. *Determine a descent direction* $\Delta x$.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

**until** stopping criterion is satisfied.



# Gradient Descent

- Most common optimization algorithm
- Easy to implement, but may be slow to converge
- Descent direction:

$$\Delta x = -\nabla f(x)$$

- Termination condition:

$$\|\nabla f(x)\|_2 \leq \epsilon$$

e.g. $\epsilon = 0.001$

# Gradient Descent: Step size

- Ideally, we would use *exact line search* to determine step size *t*:

$$t = \operatorname{argmin}_{t>0} f(x + t\Delta x)$$

- But this is slow to compute in general, so often use backtracking line search:

**given** a descent direction $\Delta x$ for $f$ at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
**while** $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x, \quad t := \beta t.$

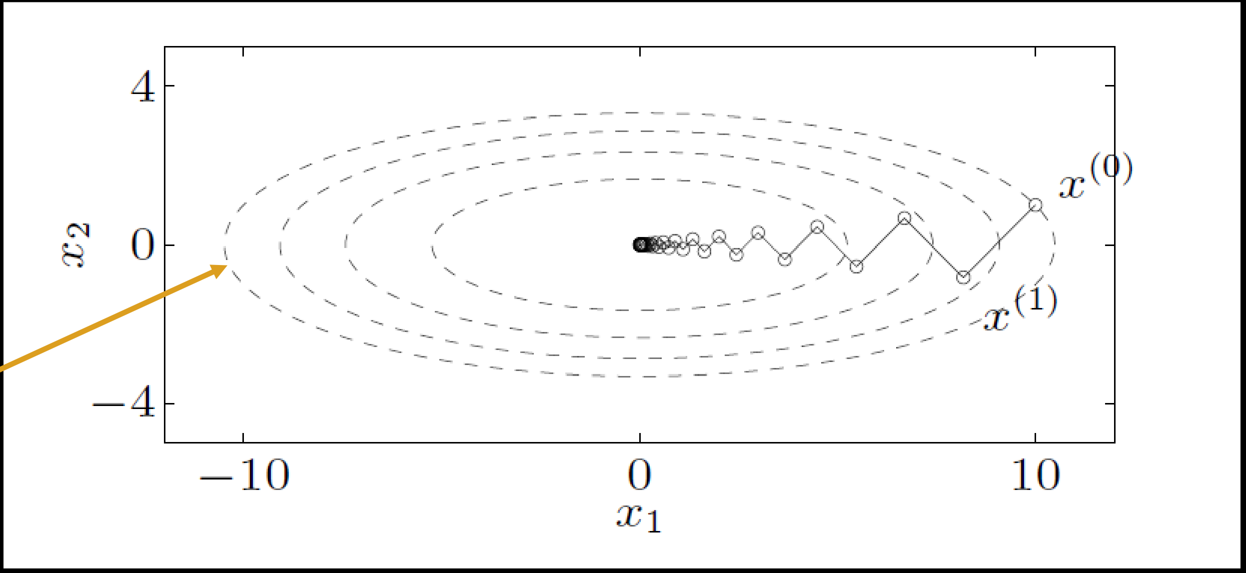
- I.e. decrease magnitude of step until you meet the stopping condition

# Gradient Descent Example

- Find the minimum of this function:

$$f(x_1, x_2) = 0.5(x_1^2 + 10x_2^2)$$

- Starting at $x^{(0)} = (10,1)$, using exact line search

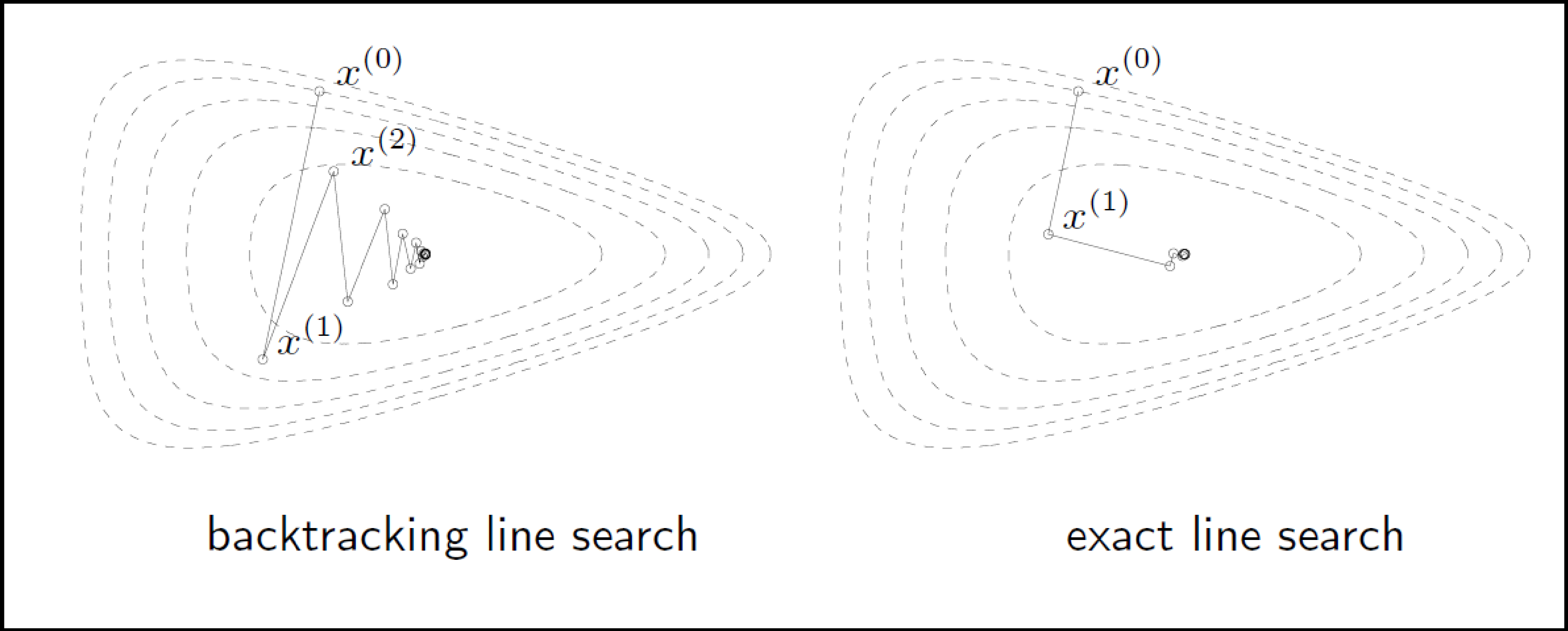



isocontours:
levels of constant f(x) value

# Gradient Descent Example

- Find the minimum of this function:

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



# Problems with Gradient Descent

- Sensitive to the condition number of the Hessian
  - High condition number means very slow convergence
- Sensitive to the coordinates you use (not affine invariant)
  - Apply a linear transform to x and you may get different results!

- Newton's method overcomes these problems by using the Hessian of the function
  - For a price (the Hessian can be expensive to compute)

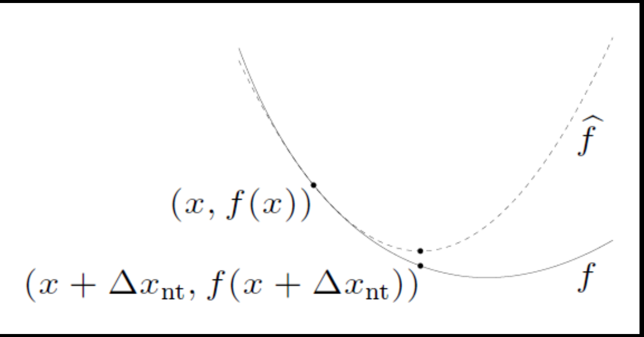
# Newton's method: Descent direction

- Determine a descent direction:

$$\Delta x_{\mathrm{nt}} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

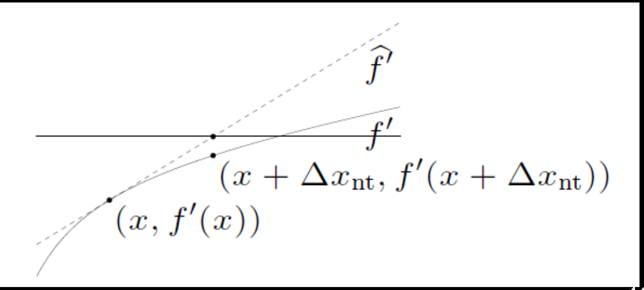
- Why?
  - Let's approximate $f(x)$ with a quadratic function (remember Taylor series):

$$\widehat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$$




  - $x + \Delta x_{nt}$ solves the linearized optimality condition:

$$\nabla f(x+v) \approx \nabla \widehat{f}(x+v) = \nabla f(x) + \nabla^2 f(x) v = 0$$



# Newton's method: Stopping criterion

- The Newton decrement $\lambda(x)$ leads to the stopping criterion:

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}$$

- $\lambda(x)$ is an estimate of the distance between f(x) and p*
- $\lambda(x)^2$ is the directional derivative in the direction of the Newton step:

$$\nabla f(x)^T \Delta x_{\mathrm{nt}} = -\lambda(x)^2$$

- If the directional derivative is very close to 0, f(x) is not changing much in this direction
  - I.e. you're very close to the optimum
  - So, when $\frac{\lambda(x)^2}{2}$ is below some small tolerance $\epsilon$, stop

# Newton's method

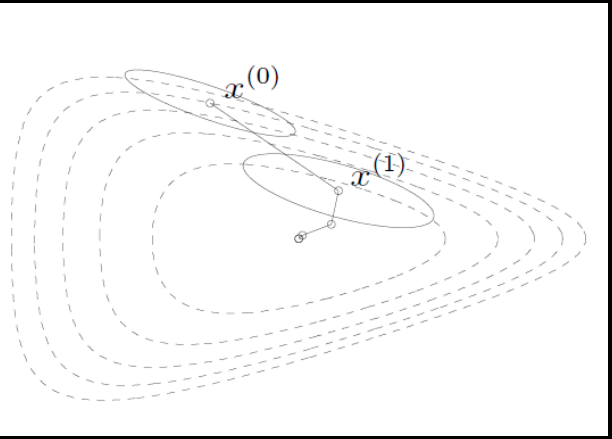
**given** a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
**repeat**

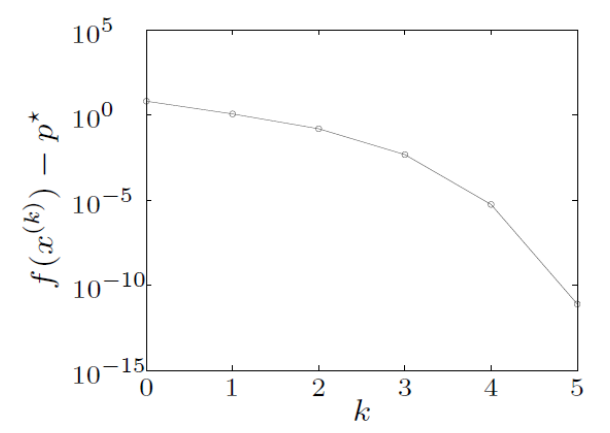
1. *Compute the Newton step and decrement.*
   $\Delta x_{\mathrm{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x).$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \le \epsilon$.
3. *Line search.* Choose step size $t$ by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

# Newton's method example

- Find the optimum of this function:

$$f(x_1, x_2) = e^{x_1+3x_2-0.1} + e^{x_1-3x_2-0.1} + e^{-x_1-0.1}$$



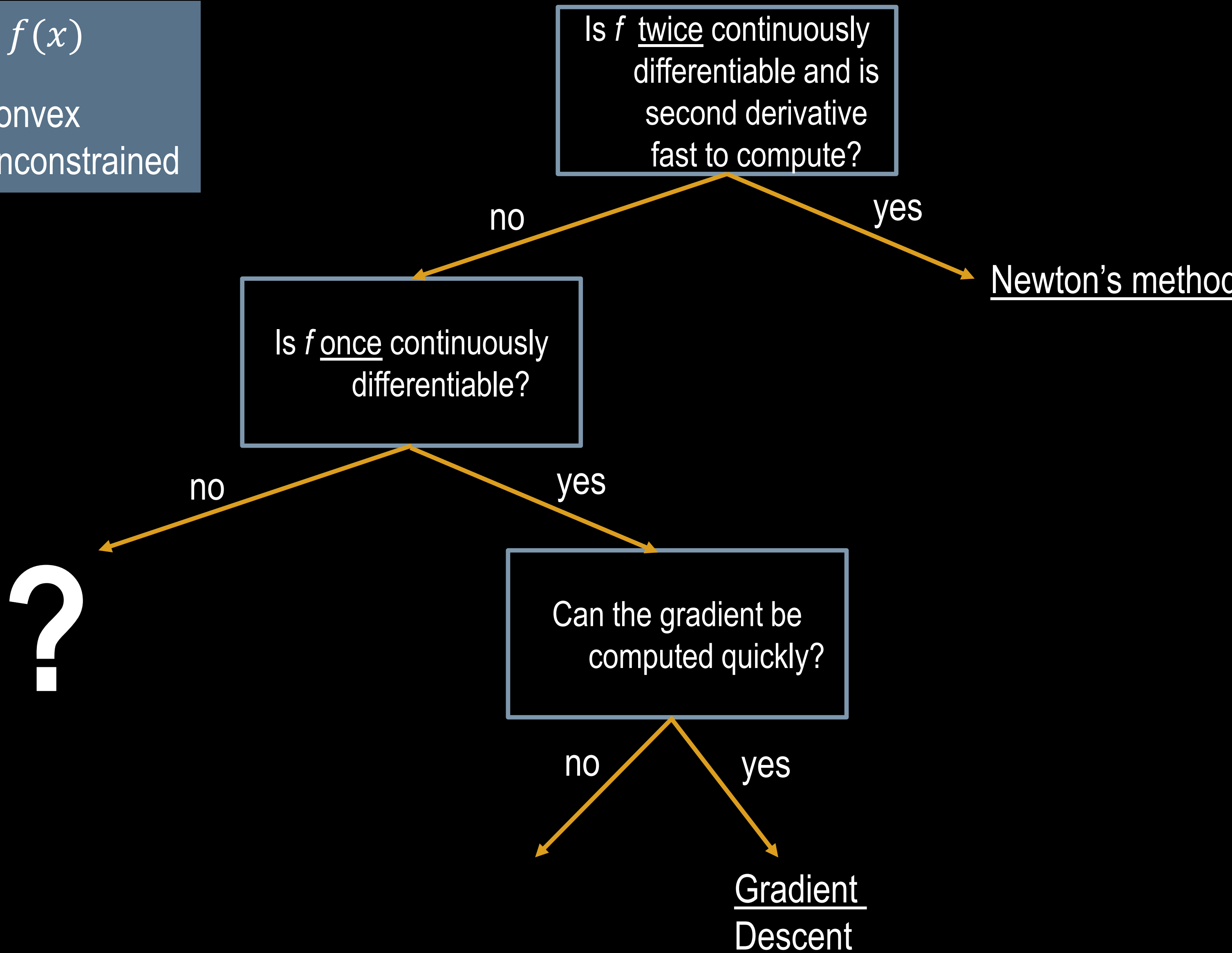

- Backtracking line search parameters:

$$\alpha = 0.1,\ \beta = 0.7$$

# Break

$$\underset{x}{\text{minimize}}\ f(x)$$

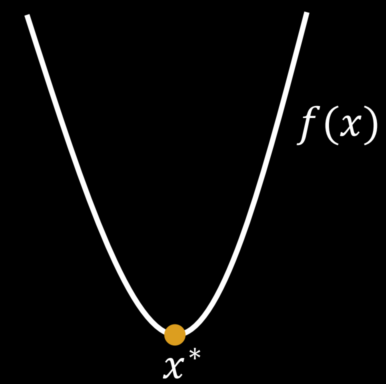
- assume $f$ is convex
- assume $x$ is unconstrained



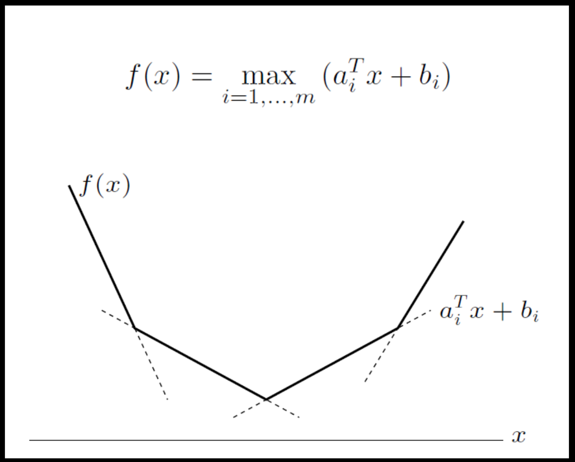

*Many more methods not covered!

# What if $f$ is not differentiable?

$$\underset{x}{\text{minimize}}\, f(x)$$




- So far, we've only considered differentiable $f$
- This is really restrictive! Can't do
  - $||x||$
  - piecewise-linear $f$
  - $\max(f_1(x), f_2(x))$
  - ….
- What if we could do something like a descent without the true gradient of $f$ ?

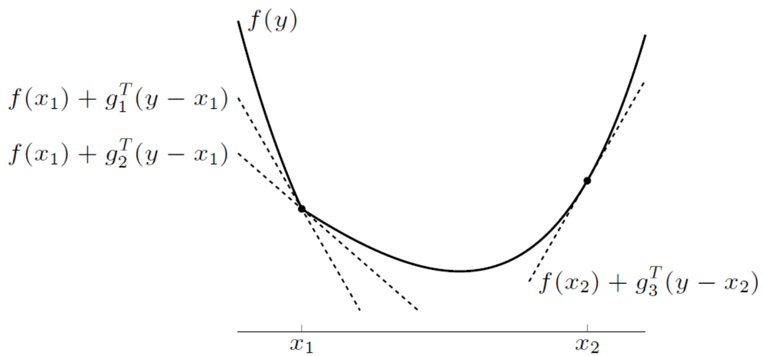


# Subgradients

- Instead of a gradient, use subgradient of the function

$g$ is a **subgradient** of a convex function $f$ at $x \in \mathrm{dom}\, f$ if

$$f(y) \geq f(x) + g^T(y - x) \quad \forall y \in \mathrm{dom}\, f$$




$g_1$, $g_2$ are subgradients at $x_1$; $g_3$ is a subgradient at $x_2$

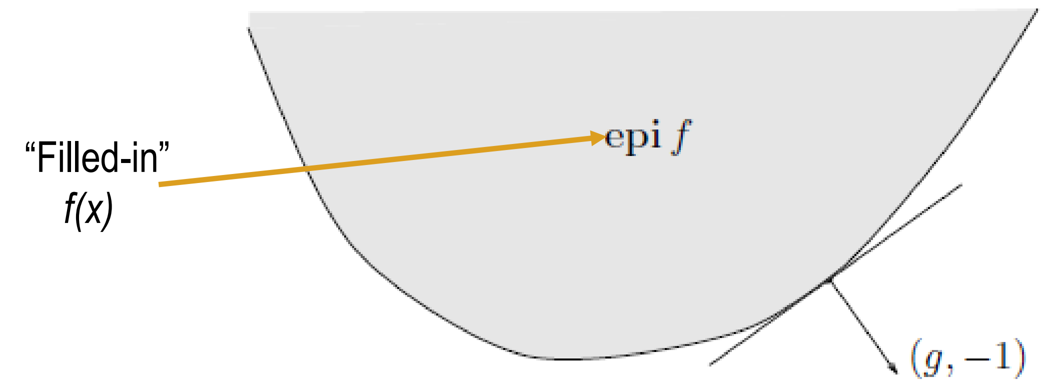
- Subdifferential:

the **subdifferential** $\partial f(x)$ of $f$ at $x$ is the set of all subgradients:

$$\partial f(x) = \{g \mid g^T(y - x) \leq f(y) - f(x),\ \forall y \in \mathrm{dom}\, f\}$$

# Subgradients

- Intuitively, think of $g$ as defining a supporting hyperplane for "filled-in" $f(x)$




A vector $g \in \mathbf{R}^n$ is a subgradient of $f$ at $x$ if and only if $(g, -1)$ defines a supporting hyperplane to $\mathbf{epi}\, f$ at $(x, f(x))$.

The hyperplane must be non-vertical for a valid $g$

# Subgradients

- Not all functions are subdifferentiable

- $f : \mathbf{R} \to \mathbf{R}$, $\mathrm{dom}\, f = \mathbf{R}_+$

$$f(x) = 1 \quad \text{if } x = 0, \qquad f(x) = 0 \quad \text{if } x > 0$$

- $f : \mathbf{R} \to \mathbf{R}$, $\mathrm{dom}\, f = \mathbf{R}_+$

$$f(x) = -\sqrt{x}$$

# Computing subgradients

- Method depends on the form of the function

**Differentiable functions:** $\partial f(x) = \{\nabla f(x)\}$ if $f$ is differentiable at $x$

**Nonnegative linear combination**

if $f(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x)$ with $\alpha_1, \alpha_2 \geq 0$, then

$$\partial f(x) = \alpha_1 \partial f_1(x) + \alpha_2 \partial f_2(x)$$

Important for stochastic gradient descent (coming up)

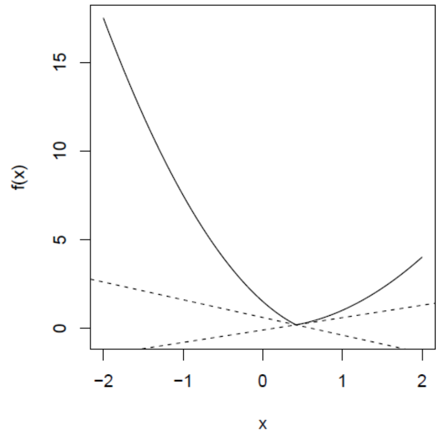
(r.h.s. is addition of sets)

**Affine transformation of variables:** if $f(x) = h(Ax + b)$, then

$$\partial f(x) = A^T \partial h(Ax + b)$$

# Example: Pointwise maximum

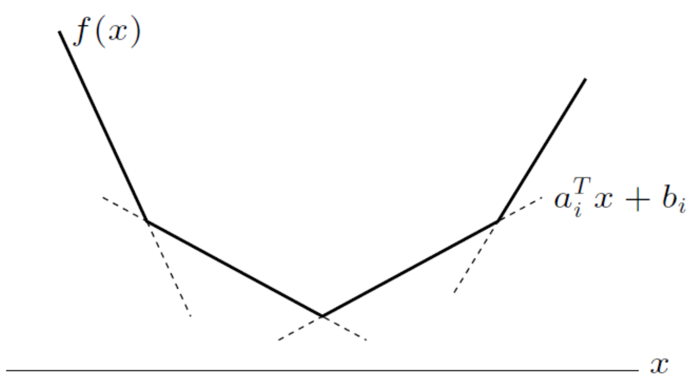
Let $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ be convex, differentiable, and consider $f(x) = \max\{f_1(x), f_2(x)\}$




- For $f_1(x) > f_2(x)$, unique subgradient $g = \nabla f_1(x)$
- For $f_2(x) > f_1(x)$, unique subgradient $g = \nabla f_2(x)$
- For $f_1(x) = f_2(x)$, subgradient $g$ is any point on the line segment between $\nabla f_1(x)$ and $\nabla f_2(x)$

# Example: Convex piecewise-linear functions

$$f(x) = \max_{i=1,\ldots,m} (a_i^T x + b_i)$$



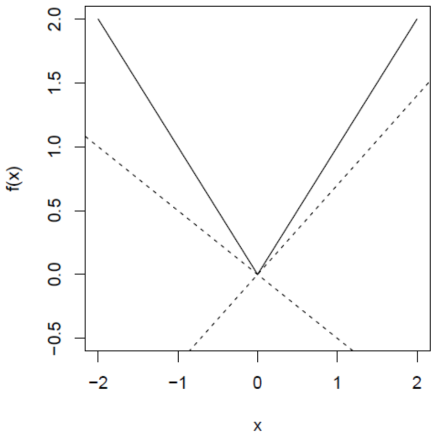

the subdifferential at $x$ is a polyhedron

$$\partial f(x) = \mathrm{conv}\,\{a_i \mid i \in I(x)\}$$

with $I(x) = \{i \mid a_i^T x + b_i = f(x)\}$ ← The "active" functions at x

# Subgradient Example

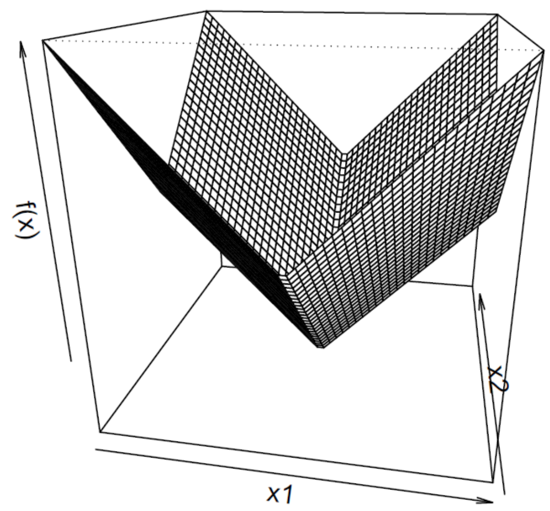
Consider $f : \mathbb{R} \to \mathbb{R}$, $f(x) = |x|$




- For $x \neq 0$, unique subgradient $g = \mathrm{sign}(x)$
- For $x = 0$, subgradient $g$ is any element of $[-1, 1]$

# Subgradient Example

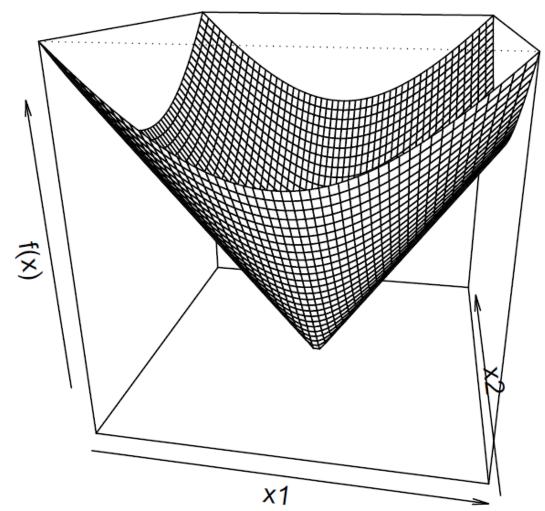
Consider $f : \mathbb{R}^n \to \mathbb{R},\ f(x) = \|x\|_1$




- For $x_i \neq 0$, unique $i$th component $g_i = \mathrm{sign}(x_i)$
- For $x_i = 0$, $i$th component $g_i$ is an element of $[-1, 1]$

# Subgradient Example

Consider $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = \|x\|$ (Euclidean norm)




- For $x \neq 0$, unique subgradient $g = x/\|x\|$
- For $x = 0$, subgradient $g$ is any element of $\{z : \|z\| \leq 1\}$

# Subgradient Method

- Similar to descent methods, but use subgradients instead of gradients and change step size

**given** a starting point $x \in \mathbf{dom}\, f$.
**repeat**
1. ~~Determine a descent direction $\Delta x$.~~
2. ~~*Line search.* Choose a step size $t > 0$.~~
3. ~~*Update.* $x := x + t\Delta x$.~~

**until** stopping criterion is satisfied.

Determine a subgradient $g^{(k)}$ of $f$ at $x^{(k)}$

(Next slide)

Update: $x^{(k+1)} = x^{(k)} - t^{(k)} g^{(k)}$

Subgradient method is not necessarily a descent method, so we keep track of best iterate $x_{\text{best}}^{(k)}$ among $x^{(1)}, \ldots x^{(k)}$ so far, i.e.,

$$f(x_{\text{best}}^{(k)}) = \min_{i=1,\ldots k} f(x^{(i)})$$

# Subgradient Method Stepsize

- Two common options:
  - Fixed step size

$$t_k = \text{constant}$$

  - Diminishing step size: choose $t_k$ to satisfy

$$\sum_{k=1}^{\infty} t_k^2 < \infty, \quad \sum_{k=1}^{\infty} t_k = \infty,$$

(square summable but not summable)

    - I.e. step sizes go to 0 but not too fast

- Important difference from other descent methods:

All step sizes are pre-specified, not computed through line search

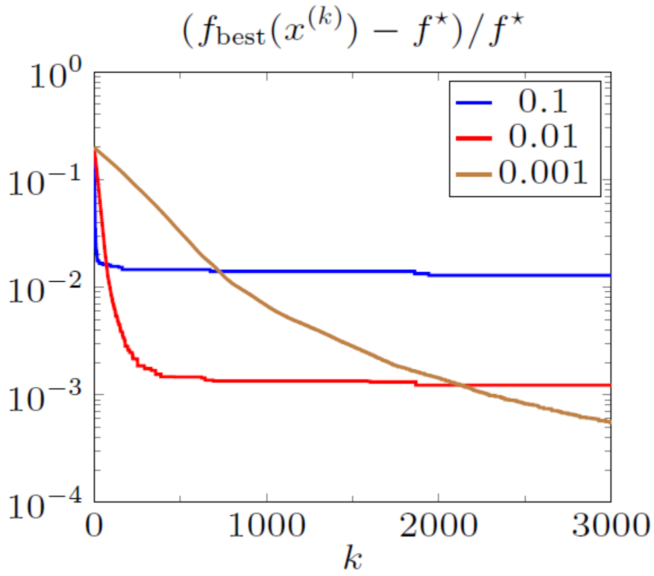
- Convergence is hard to test, since subgradient is not necessarily a descent direction

# Example: 1-norm minimization

$$\text{minimize } \|Ax - b\|_1$$

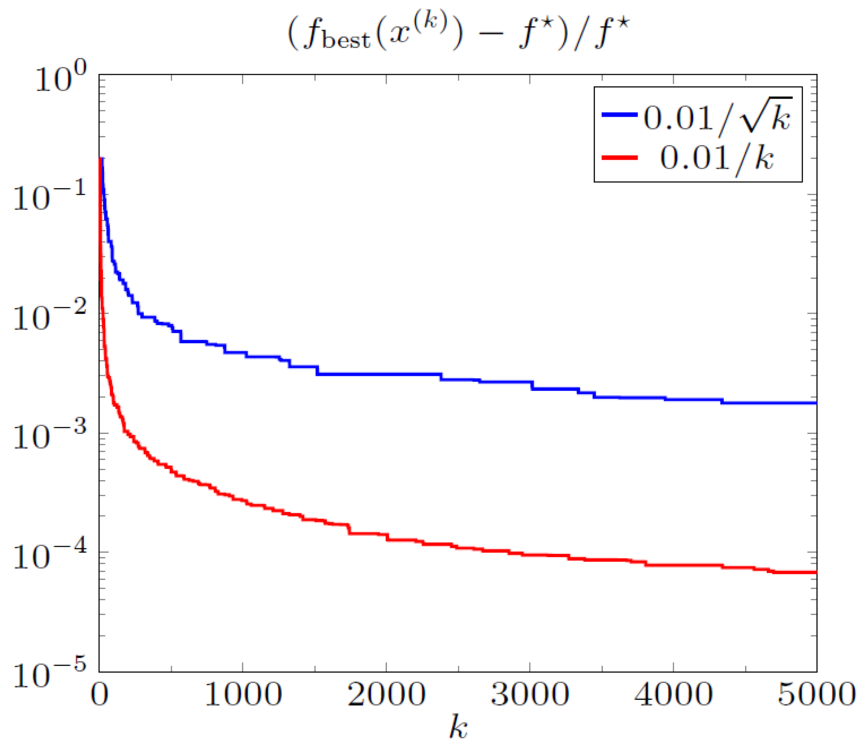
- subgradient is given by $A^T \,\mathbf{sign}(Ax - b)$
- example with $A \in \mathbf{R}^{500\times 100}$, $b \in \mathbf{R}^{500}$

**Fixed steplength** $t_k = s/\|g^{(k-1)}\|_2$ for $s = 0.1, 0.01, 0.001$



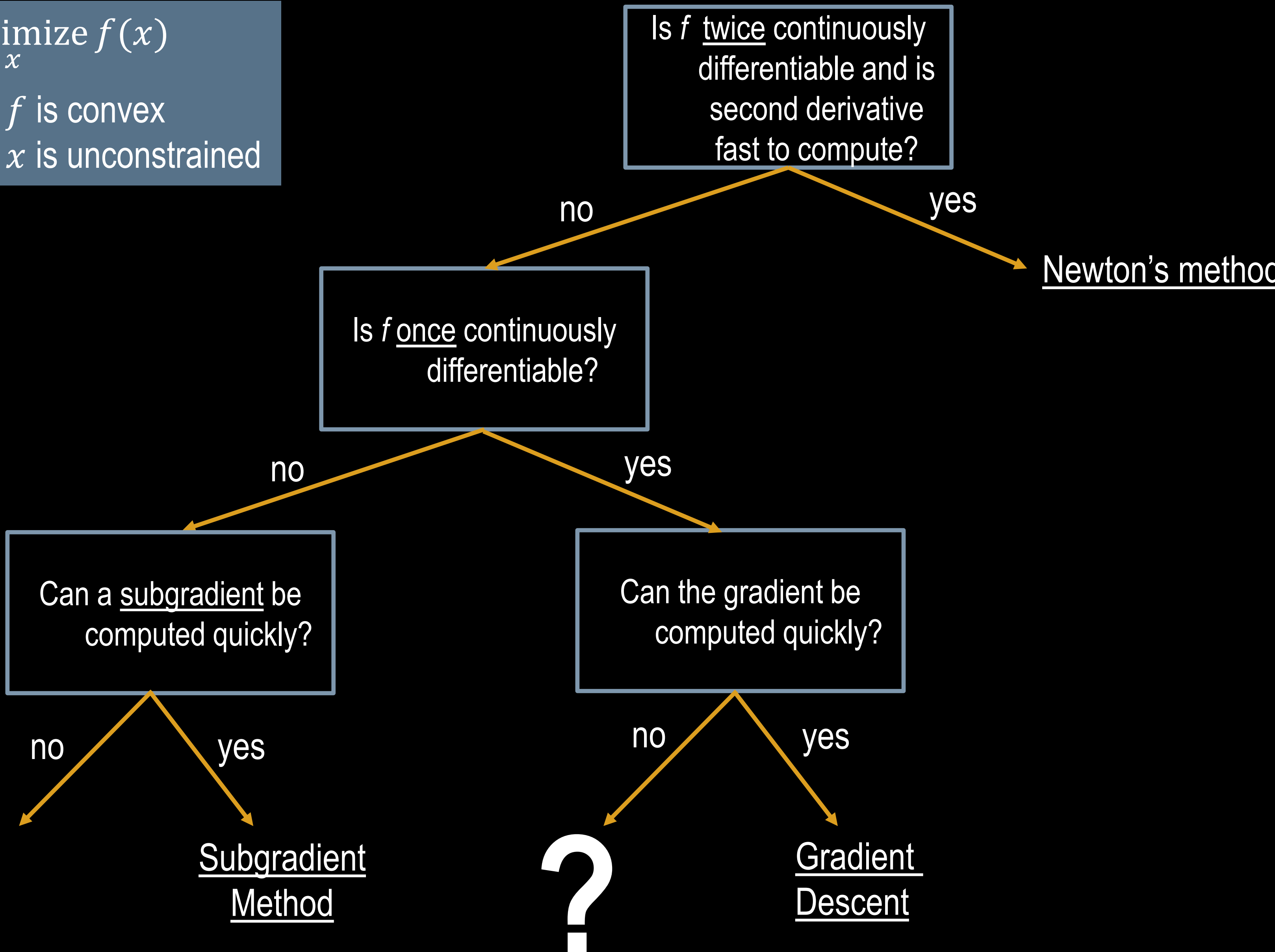
# Using diminishing step sizes

**Diminishing step size:** $t_k = 0.01/\sqrt{k}$ and $t_k = 0.01/k$



$$\underset{x}{\text{minimize}}\ f(x)$$

- assume $f$ is convex
- assume $x$ is unconstrained




*Many more methods not covered!

# Stochastic Gradient Descent

# Stochastic Gradient Descent (SGD)

- A particularly important function form for machine learning:

$$F(x) = f_1(x) + f_2(x) + \cdots + f_n(x)$$

- Example: each $f_i(x)$ represents the error of model $x$ when estimating the $i$th data point, want to find the $x$ with minimum total error.

- Imagine you have *millions* of data points, the gradient $\nabla F(x)$ will be *very expensive* to compute for a complex $x$ (e.g. a neural network).
- Key idea: Use gradient (or subgradient) of *only one* $f_i(x)$ at each iteration

# Stochastic Gradient Descent (SGD)

- Recall update for descent methods

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

- For Stochastic Gradient Descent

$$\Delta x^{(k)} = -\nabla f_i(x^{(k)}) \qquad \text{for some } i \leq n$$

  - Could also use subgradient here if no gradient is available
- Can loop over $i = 1,2, \dots n$ then randomize order at end of each pass
  - Randomization of order can prevent cycling
- Can also do a batch of $i$s at each iteration

$$\Delta x^{(k)} = -\nabla f_i\left(x^{(k)}\right) - \nabla f_j\left(x^{(k)}\right) \dots \qquad \text{for some } i, j, \dots \leq n$$

# Stochastic Gradient Descent (SGD)

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

- To prove convergence, need

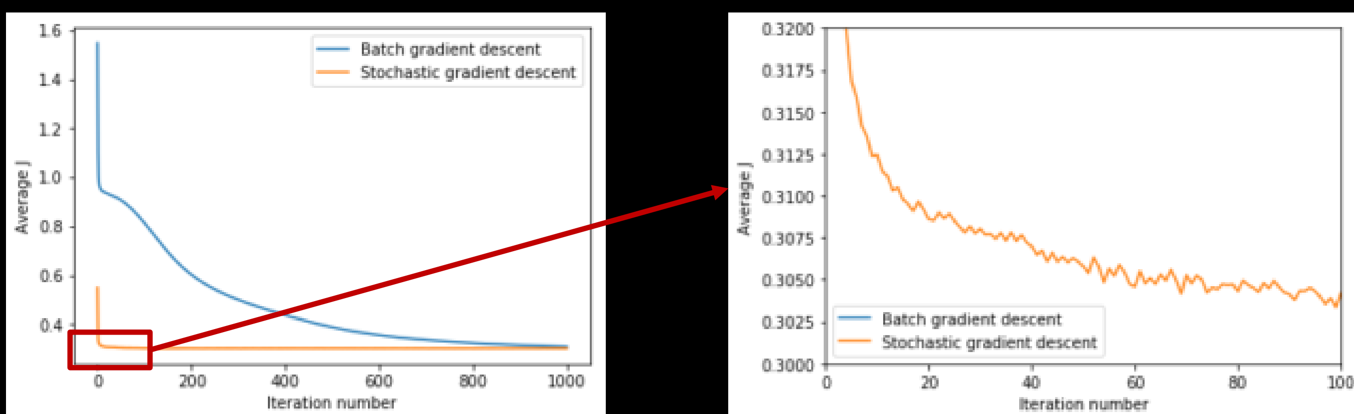
$$t^{(k)} \to 0 \text{ as } k \to \infty$$

$$\sum_{k=1}^{\infty} t^{(k)} = \infty$$

- Example: $t^{(k)} = \frac{1}{k}$
- However, in practice, many people use a fixed small $t^{(k)}$

# Stochastic Gradient "Descent"

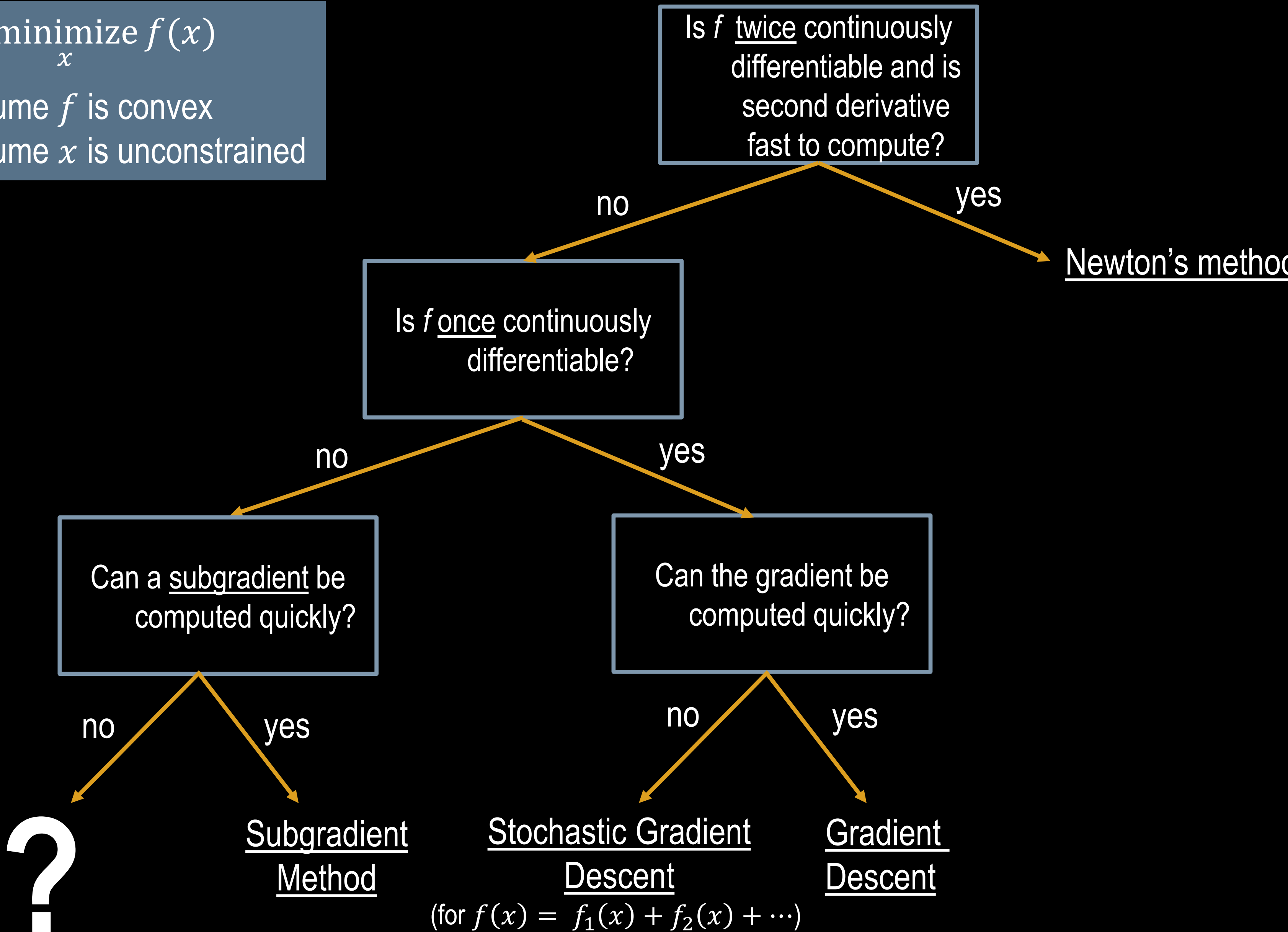
- Descent methods guarantee that $F\left(x^{(k+1)}\right) < F\left(x^{(k)}\right)$
- SGD <u>does not</u>, so not a true descent method.
  - Updates are "noisy", so $F$ value not always decreasing




Results of training a Neural Network on the MNIST dataset (handwritten character recognition)
(http://adventuresinmachinelearning.com/stochastic-gradient-descent/)

$$\underset{x}{\text{minimize}}\ f(x)$$

- assume $f$ is convex
- assume $x$ is unconstrained



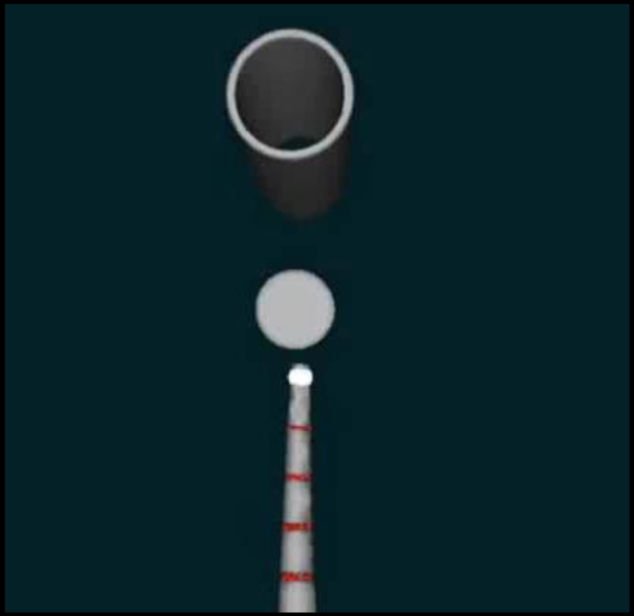

*Many more methods not covered!

# Numerical Differentiation

# What about functions that you don't know analytically?

- So far *f(x)* is always represented analytically
- What if *f(x)* is this:

$x$ is actuator forces/torques



$f(x)$ outputs positions of soft bodies

"Optimization-based inverse model of Soft Robots with Contact Handling"
Eulalie Coevoet, Adrien Escande, Christian Duriez

# Numerical Differentiation

- Need a way to differentiate when the function is not represented analytically
- Assume we can evaluate the function at any $x$
  - E.g. by running some code like a simulation
- Recall standard derivative definition for $f: \mathbf{R} \to \mathbf{R}$

$$Df(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Key idea: Evaluate function at two points per dimension and estimate the derivative

# Numerical differentiation for univariate functions

1. Pick a small $h$
2. Use a Finite Difference method. Two common ones:
   a) Newton's Difference Quotient

$$Df(x) \approx \frac{f(x+h) - f(x)}{h}$$

   b) Symmetric Difference Quotient

$$Df(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

- There are other numerical methods which can give better estimates but use more function evaluations

# Numerical differentiation for multidimensional functions

- For $f: \mathbf{R}^n \to \mathbf{R}^m$ we do the same thing to compute the Jacobian
- Recall:

$$Df(x)_{ij} = \frac{\partial f_i(x)}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- Let $\delta(j,h) = [0, \ldots, \underset{\text{index } j}{h}, \ldots 0]^T$

$$Df(x)_{ij} \approx \frac{f(x + \delta(j,h))_i - f(x)_i}{h}$$

- Similar process for Symmetric Difference Quotient
- Thus we can use numerical differentiation to compute the gradient for gradient descent

# Limitations

- Choosing $h$ well is difficult in general (it is function-dependent)
  - Many use a fixed $h$ for simplicity
- Numerical methods can be very sensitive to the choice of $h$
- There can be errors due to machine precision and floating point arithmetic

# Going Further: Automatic Differentiation

- Automatic Differentiation is far more sophisticated than numerical differentiation

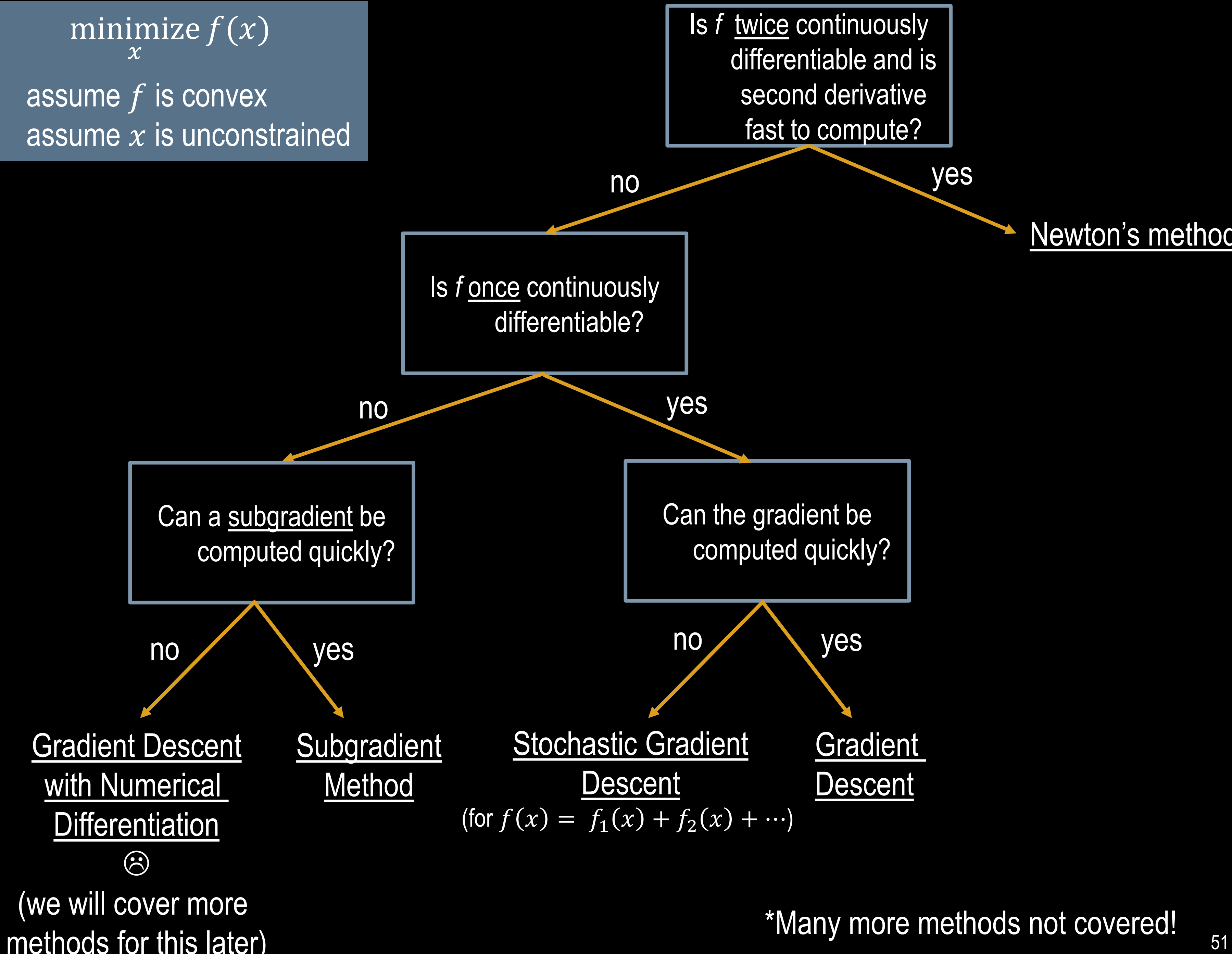
- We won't go into it, but an overview is here:

  https://en.wikipedia.org/wiki/Automatic_differentiation

- Very popular in the machine learning world!

$$\underset{x}{\text{minimize}}\ f(x)$$

- assume $f$ is convex
- assume $x$ is unconstrained




*Many more methods not covered!

# Homework

- Reading from Optimization Book
  - Optimization Problems (Ch. 4.1-4.1.2, 4.3-4.3.1 (skip examples), 4.4-4.4.1 (only read first example in 4.4.1)
  - Duality (Ch. 5.1-5.1.5, 5.2-5.2.3, 5.5)
  - Barrier Method (Ch. 11.1-11.3)