

Artificial Intelligence

CE-417, Group 1

Computer Eng. Department

Sharif University of Technology

Spring 2024

By Mohammad Hossein Rohban, Ph.D.

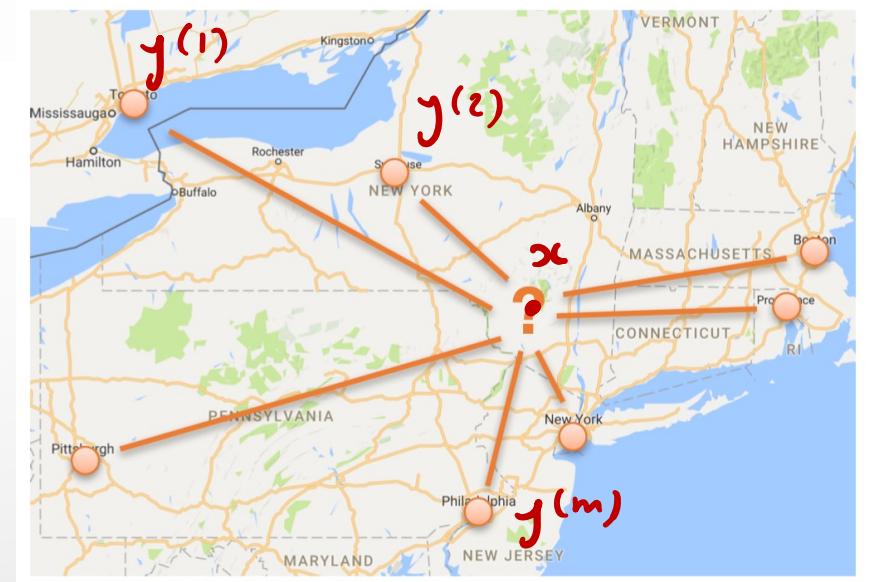
Courtesy: Most slides are adopted from **15-780** course at CMU.

Continuous Optimization

Example: Weber Point

- Given a collection of cities (assume on 2D plane) how can we find the location that minimizes the sum of distances to all cities?
- Denote the locations of the cities as $\underbrace{y^{(1)}, \dots, y^{(m)}}$
- Write as the optimization problem:

$$\underset{x}{\text{minimize}} \sum_{i=1}^m \left\| x - \underbrace{y^{(i)}}_i \right\|_2$$



Numerical Optimization

Example: Image deblurring and denoising

$$K * X$$

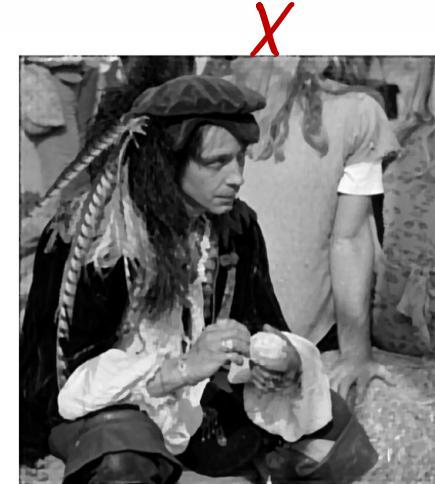


Figure from (O'Connor and Vandenberghe, 2014)

- Given corrupted image $Y \in \mathbb{R}^{m \times n}$, reconstruct the image by solving the optimization:

Convolution with a Kernel K

$$\underset{X}{\text{minimize}} \sum_{i,j} |Y_{ij} - (K * X)_{ij}| + \lambda \sum_{i,j} \sqrt{(X_{ij} - X_{i,j+1})^2 + (X_{i+1,j} - X_{ij})^2}$$

- where K^* denotes convolution with a blurring filter

$$\begin{aligned}
 & \left[\begin{array}{ccc} 1 & 2 & 1 \\ 1 & 1 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 2 \end{array} \right] \overset{\sigma}{\otimes} \left[\begin{array}{cc} 1 & 1 \end{array} \right] \overset{q}{\rightarrow} \\
 & = \left[\begin{array}{cc} 7 & 4 \\ 4 & 3 \end{array} \right]
 \end{aligned}$$

Example: robot trajectory planning

- Many robotic planning tasks are more complex than shortest path, e.g. have robot dynamics, require “smooth” controls
- Common to formulate planning problem as an optimization task
- Robot state x_t and inputs u_t :

$$\begin{aligned} & \underset{x_{1:T}, u_{1:T-1}}{\text{minimize}} \quad \sum_{i=1}^T \|u_t\|_2^2 \\ & \text{subject to} \quad x_{t+1} = f_{\text{dynamics}}(x_t, u_t) \\ & \qquad \qquad \qquad x_t \in \text{FreeSpace}, \forall t \\ & \qquad \qquad \qquad x_1 = x_{\text{init}}, x_T = x_{\text{goal}} \end{aligned}$$

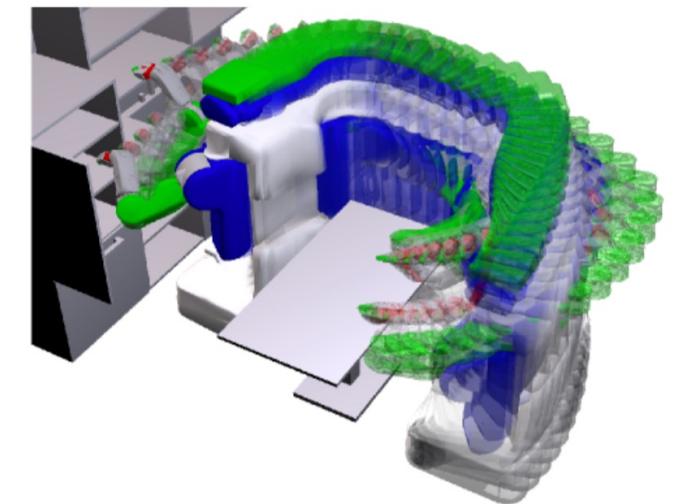


Figure from (Schulman et al., 2014)

Example: Machine Learning

- As we will see in much more detail shortly, virtually all (supervised) machine learning algorithms boil down to solving an optimization problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^m \ell(h_{\theta}(x^{(i)}), y^{(i)})$$

where

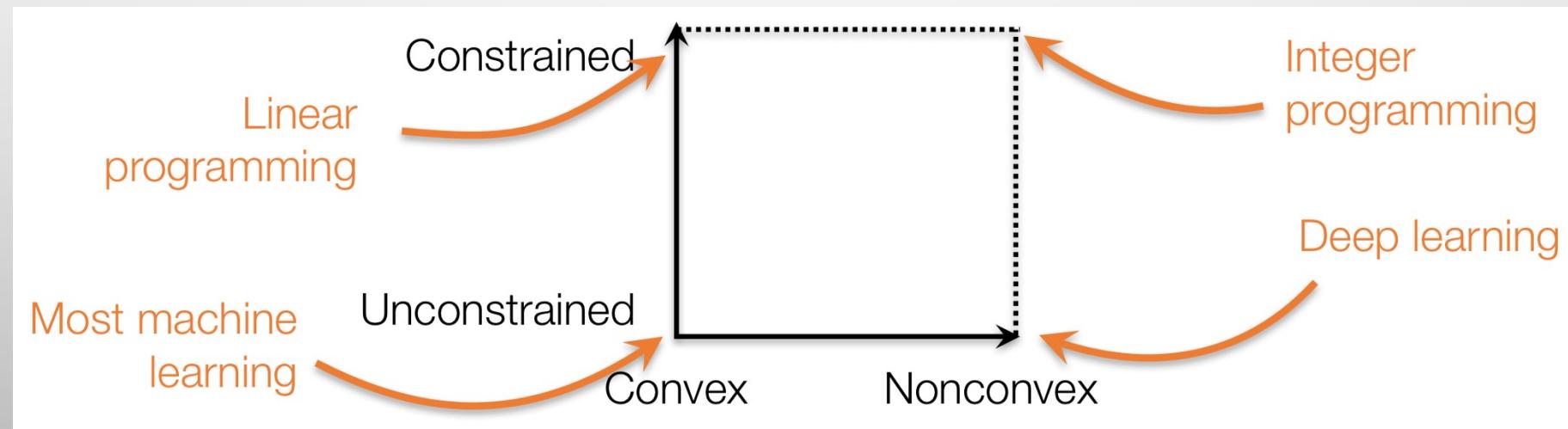
- $x^{(i)} \in \mathcal{X}$ are inputs
- $y^{(i)} \in \mathcal{Y}$ are outputs
- ℓ is a loss function
- h_{θ} is a hypothesis function parameterized by θ

The benefit of optimization

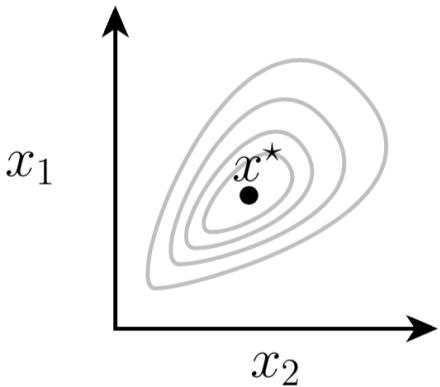
- One of the key benefits of looking at problems in AI as optimization problems: we separate out the *definition* of the problem from the *method for solving it*.
- For many classes of problems, there are off-the-shelf solvers that will let you solve even large, complex problems, once you have put them in the right form.

Classes of optimization problems

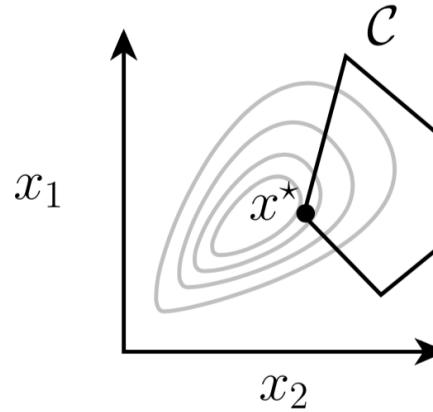
- Many different names for types of optimization problems: linear programming, quadratic programming, nonlinear programming, semidefinite programming, integer programming, geometric programming, mixed linear binary integer programming (the list goes on and on, can all get a bit confusing)
- We're instead going to focus on two dimensions: convex vs. nonconvex and constrained vs. unconstrained



Constrained vs. unconstrained



minimize _{x} $f(x)$



minimize _{x} $f(x)$
subject to $x \in \mathcal{C}$

- In **unconstrained optimization**, every point $x \in \mathbb{R}^n$ is feasible, so singular focus is on minimizing $f(x)$
- In contrast, for **constrained optimization**, it may be difficult to even *find* a point $x \in \mathcal{C}$
- Often leads to kind of different methods for optimization

How hard is real-valued optimization?

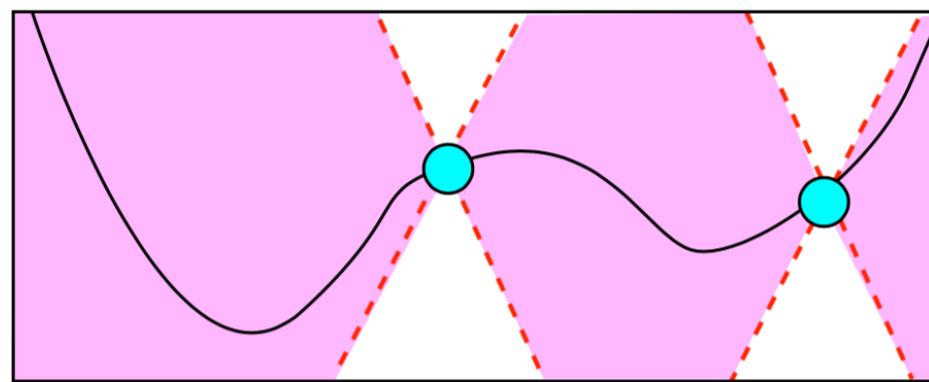
- How long does it take to find an ε -optimal minimizer of a real-valued function?

$$\min_{x \in \mathbb{R}^n} f(x).$$

General function: impossible!

- We need to make some assumptions about the function:
 - Assume f is **Lipschitz-continuous**: (can not change too quickly)

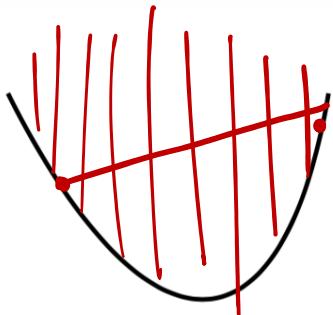
$$|f(x) - f(y)| \leq L\|x - y\|.$$



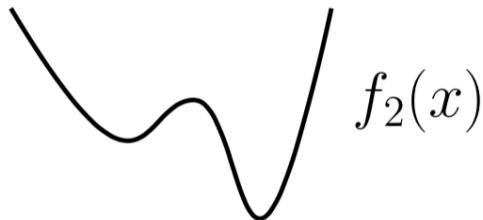
How hard is real-valued optimization? (cont.)

- After t iterations, the error of any algorithm is $\Omega\left(\frac{1}{t^{1/n}}\right)$.
 - Any grid-search is nearly optimal
- Optimization is hard, but assumptions make a big difference.
 - we went from impossible to very slow

Convex vs. nonconvex optimization



Convex function



Nonconvex function

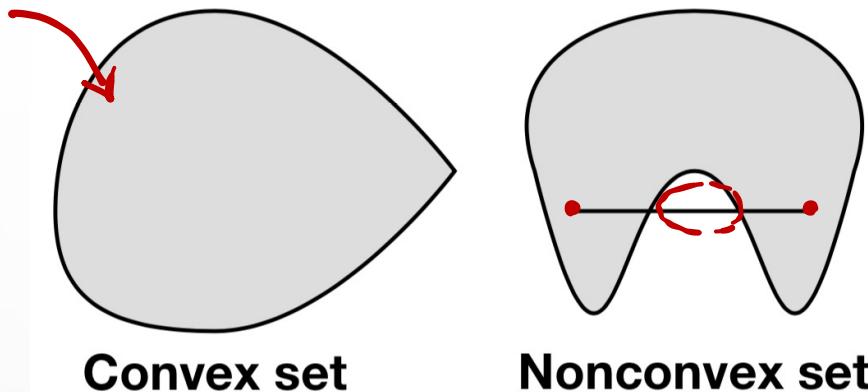
- Originally, researchers distinguished between linear (easy) and nonlinear (hard) problems
- But in 80s and 90s, it became clear that this wasn't the right distinction, key difference is between convex and nonconvex problems
- Convex problem:

$$\begin{aligned} & \underset{x}{\text{minimize}} && f(x) \\ & \text{subject to} && x \in \mathcal{C} \end{aligned}$$

Convex function
Convex Set

where f is a convex function and \mathcal{C} is a convex set

Convex sets

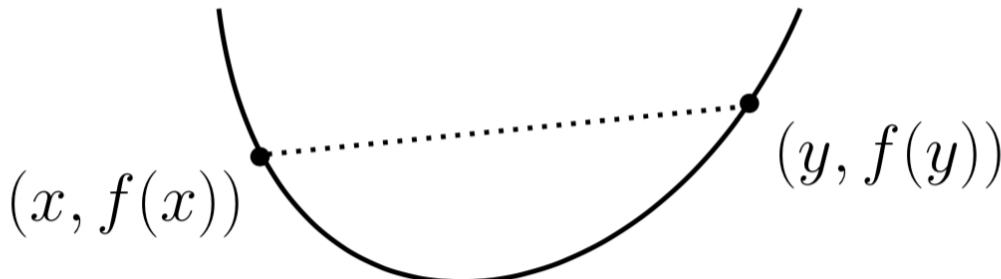


- A set \mathcal{C} is **convex** if, for any $x, y \in \mathcal{C}$ and $0 \leq \theta \leq 1$
 - $\theta x + (1 - \theta) y \in \mathcal{C}$
- Examples:
 - All points $\mathcal{C} = \mathbb{R}^n$
 - Intervals $\mathcal{C} = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$ (elementwise inequality)
 - Linear equalities $\mathcal{C} = \{x \in \mathbb{R}^n \mid Ax = b\}$ (for $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$)
 - Intersection of convex sets $\mathcal{C} = \bigcap_{i=1}^m \mathcal{C}_i$

Convex functions

- A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if, for any $x, y \in \mathbb{R}^n$ and $0 \leq \theta \leq 1$

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$



- Convex functions “curve upwards” (or at least not downwards)
- If f is convex then $-f$ is concave
- If f is both convex and concave, it is affine, must be of form:

$$f(x) = \sum_{i=1}^n a_i x_i + b$$

2nd derivative being positive iff convexity (one dimensional)

if part

From convexity, $f(ta + (1 - t)b) \leq tf(a) + (1 - t)f(b)$.

Let $t = 1/2$, $a = x - h$, and $b = x + h$.

$$\begin{aligned} f(x) &\leq \frac{1}{2}f(x - h) + \frac{1}{2}f(x + h) \\ \implies f(x + h) - 2f(x) + f(x - h) &\geq 0 \end{aligned}$$

Then

Only if part

Proof: We use the Taylor series expansion of the function around x_0 :

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x^*)}{2}(x - x_0)^2, \quad (2.73)$$

where x^* lies between x_0 and x . By hypothesis, $f''(x^*) \geq 0$, and thus the last term is nonnegative for all x .

We let $x_0 = \lambda x_1 + (1 - \lambda)x_2$ and take $x = x_1$, to obtain

$$f(x_1) \geq f(x_0) + f'(x_0)((1 - \lambda)(x_1 - x_2)). \quad (2.74)$$

Similarly, taking $x = x_2$, we obtain

$$f(x_2) \geq f(x_0) + f'(x_0)(\lambda(x_2 - x_1)). \quad (2.75)$$

Multiplying (2.74) by λ and (2.75) by $1 - \lambda$ and adding, we obtain (2.72). The proof for strict convexity proceeds along the same lines. \square

Hessian being positive semi-definite iff convexity (multi-dimensional)

- Function $f(\cdot)$ is convex iff its one-dimensional projection along any direction d , $g(t) = f(\cdot + td)$ is convex.
- Note that the 2nd derivative of g is $d^T H_f d$, where H_f is the hessian of the function f .
- $d^T H_f d$ being non-negative for any d means H_f being positive semi-definite.

Examples of convex functions

Exponential: $f(x) = \exp(ax)$, $a \in \mathbb{R}$

Negative logarithm: $f(x) = -\log x$, with domain $x > 0$

Squared Euclidean norm: $f(x) = \|x\|_2^2 \equiv x^T x \equiv \sum_{i=1}^n x_i^2$

Euclidean norm: $f(x) = \|x\|_2$

Non-negative weighted sum of convex functions

$$f(x) = \sum_{i=1}^m w_i f_i(x), \quad w_i \geq 0, f_i \text{ convex}$$

Poll: convex sets and functions

Which of the following functions or sets are convex?

1. A union of two convex sets $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$
2. The set $\{x \in \mathbb{R}^2 | x \geq 0, x_1 x_2 \geq 1\}$
3. The function $f: \mathbb{R}^2 \rightarrow \mathbb{R}, f(x) = x_1 x_2$
4. The function $f: \mathbb{R}^2 \rightarrow \mathbb{R}, f(x) = x_1^2 + x_2^2 + x_1 x_2$

Convex Optimization

- The key aspect of convex optimization problems that make them tractable is that *all local optima are global optima*.
- **Definition:** a point x is globally optimal if x is feasible and there is no feasible y such that $f(y) < f(x)$
- **Definition:** a point x is locally optimal if x is feasible and there is some $R > 0$ such that for all feasible y with $\|x - y\|_2 \leq R$, $f(x) \leq f(y)$
- **Theorem:** For a convex optimization problem all locally optimal points are globally optimal.

Proof of global optimality

- **Proof:** Given a locally optimal x (with optimality radius R), and suppose there exists some feasible y such that $f(y) < f(x)$

Now consider the point

$$z = \theta x + (1 - \theta)y, \quad \theta = 1 - \frac{R}{2\|x - y\|_2}$$

- 1) Since $x, y \in \mathcal{C}$ (feasible set), we also have $z \in \mathcal{C}$ (by convexity of \mathcal{C})
- 2) Furthermore, since f is convex:

$$f(z) = f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) < f(x) \quad \text{and}$$
$$\|x - z\|_2 = \left\| x - \left(1 - \frac{R}{2\|x - y\|_2}\right)x + \frac{R}{2\|x - y\|_2}y \right\|_2 = \left\| \frac{R(x - y)}{2\|x - y\|_2} \right\|_2 = \frac{R}{2}$$

Thus, z is feasible, within radius R of x , and has lower objective value, a contradiction of supposed local optimality of x

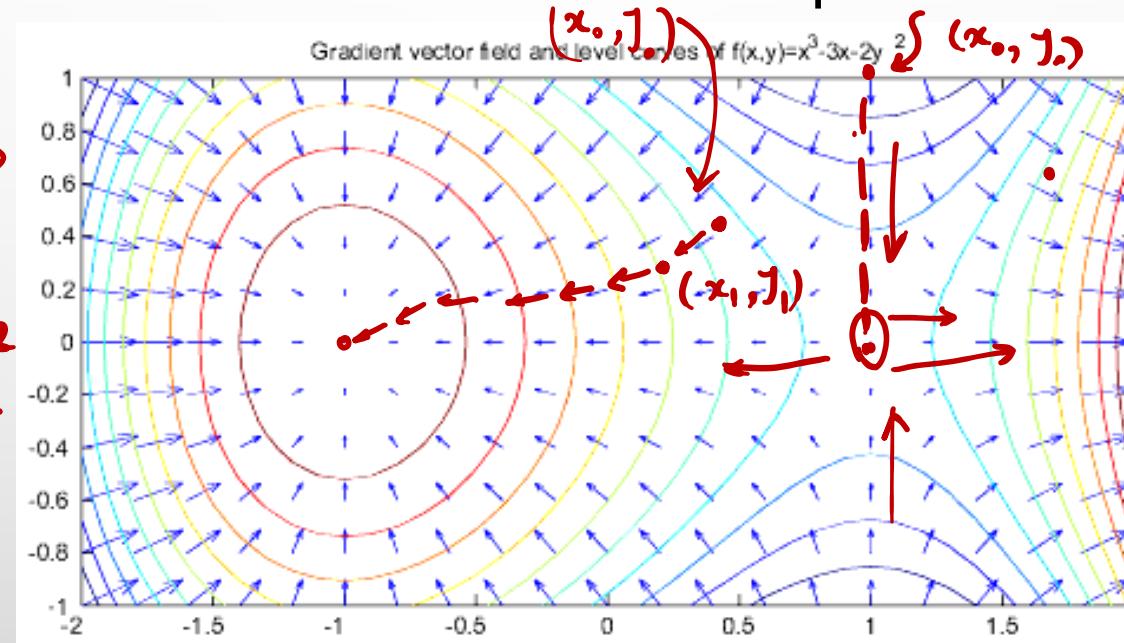
The gradient

Gradient Descent

- A key concept in solving optimization problems is the notation of the gradient of a function (multi-variate analogue of derivative)
- For $f : \mathbb{R}^n \rightarrow \mathbb{R}$, gradient is defined as vector of partial derivatives

S

Saddle
Point



$$f(x,y) = x^3 - 3x - 2y^2$$

$$\max_{x,y} f(x,y)$$

- Points in “steepest direction” of increase in function f .

Gradient descent

- Gradient motivates a simple algorithm for minimizing $f(x)$: take small steps in the direction of the negative gradient

$$f(x + \Delta x) \approx f(x) + \frac{\nabla f(x) \cdot \Delta x}{\| \nabla f(x) \|^2} + \dots o(\alpha)$$

Algorithm: Gradient Descent

Given:

Function f , initial point x_0 , step size $\alpha > 0$

Initialize:

$$\underline{x \leftarrow x_0}$$

Repeat until convergence:

$$\underline{x \leftarrow x - \alpha \nabla_x f(x)}$$

How does this guarantee to decrease f ?

- “Convergence” can be defined in a number of ways

Gradient descent works

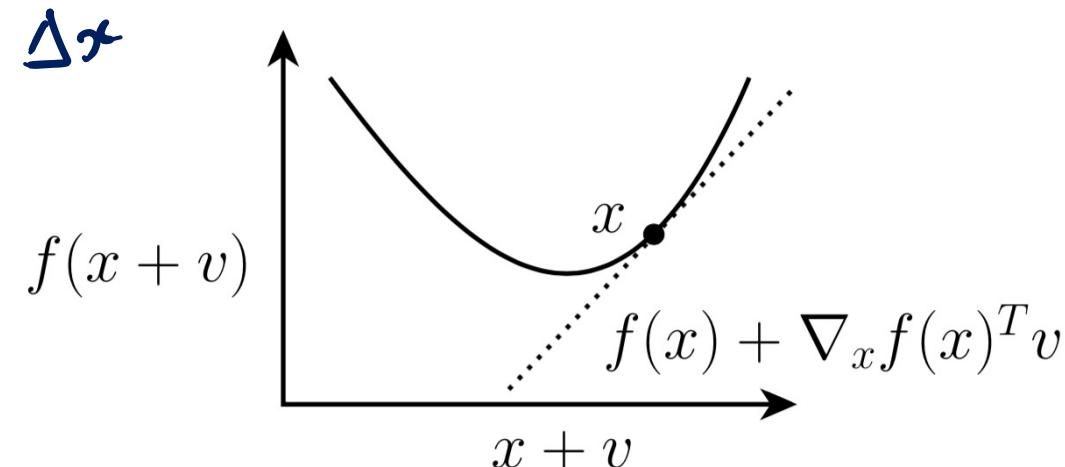
- **Theorem:** For differentiable f and small enough α , at any point x that is not a (local) minimum

$$f(x - \alpha \nabla_x f(x)) < f(x)$$

i.e., gradient descent algorithm will decrease the objective

- **Proof:** Any differentiable function f can be written in terms of its *Taylor expansion*:

$$f(x + v) = f(x) + \nabla_x f(x)^T v + O(\|v\|_2^2)$$



Gradient descent works (cont.)

Δx

- Choosing $v = -\alpha \nabla_x f(x)$, we have

$$f(x - \alpha \nabla_x f(x)) = f(x) - \alpha \nabla_x f(x)^T \nabla_x f(x) + O(\|\alpha \nabla_x f(x)\|_2^2)$$

$$\begin{aligned} &\leq f(x) - \alpha \|\nabla_x f(x)\|_2^2 + C \|\alpha \nabla_x f(x)\|_2^2 \quad \alpha \leq C' \\ &= f(x) - (\alpha - \alpha^2 C) \|\nabla_x f(x)\|_2^2 \\ &< f(x) \quad (\text{for } \alpha < 1/C \text{ and } \|\nabla_x f(x)\|_2^2 > 0) \end{aligned}$$

$$\alpha - \alpha^2 C > 0$$

$$\alpha(1 - \alpha C) > 0$$

- (Watch out: a bit of subtlety of this line, only holds for small $\alpha \nabla_x f(x)$)

$$\alpha < \min(C', 1/C)$$

- We are guaranteed to have $\|\nabla_x f(x)\|_2^2 > 0$ except at optima

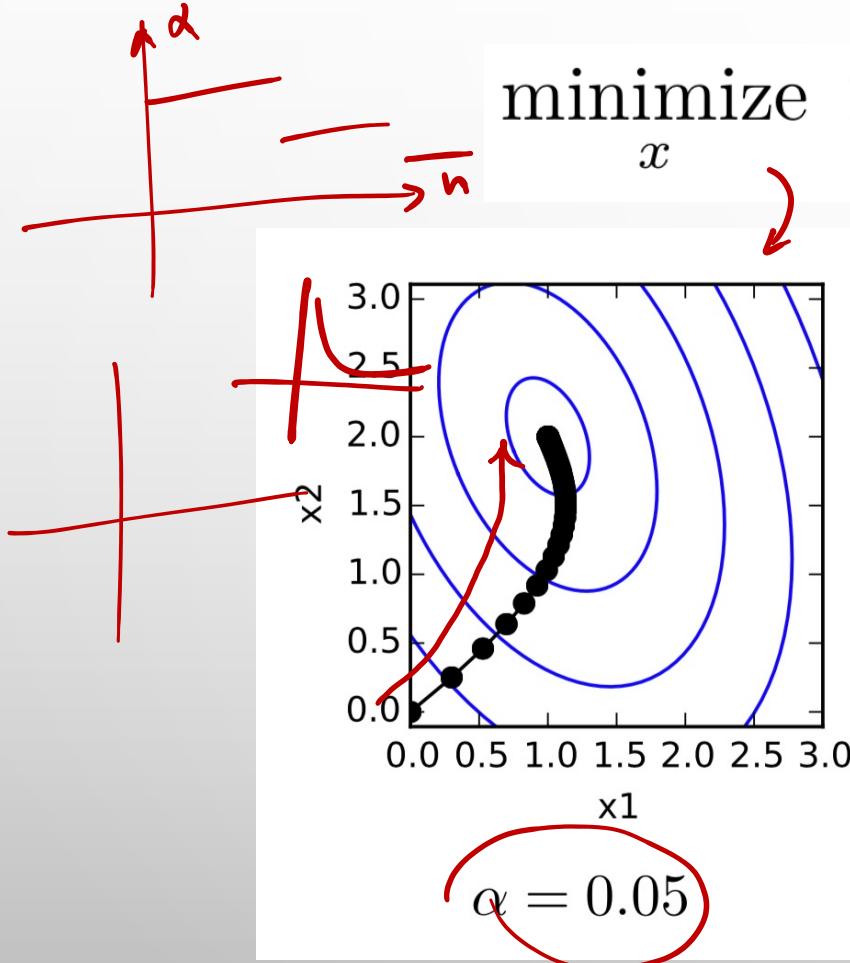
- Works for both convex and non-convex functions, but with convex functions guaranteed to find global optimum



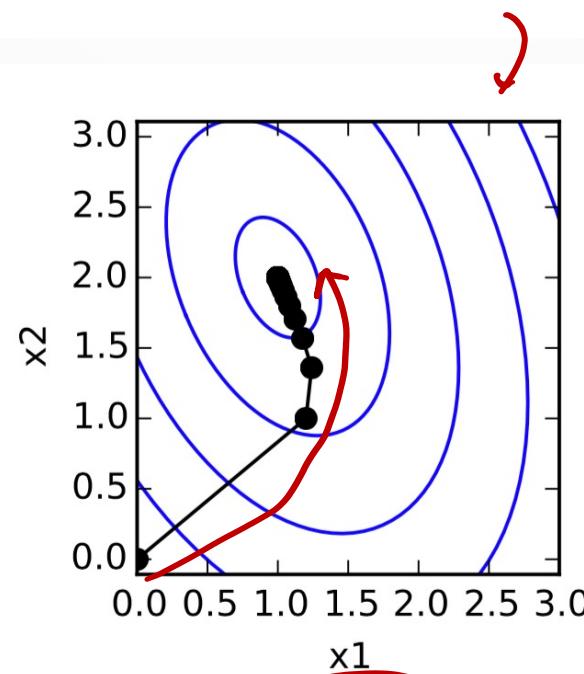
Gradient descent in practice

α Schedule α_n

- Choice of α matters a lot in practice:



$$\text{minimize}_x 2x_1^2 + x_2^2 + x_1 x_2 - 6x_1 - 5x_2$$



$$g(\alpha) \in O(\alpha^2)$$

$$\Leftrightarrow \forall \underline{\alpha} \leq \alpha \leq \bar{\alpha} : g(\alpha) \leq C\alpha^2$$

