15-780 – Graduate Artificial Intelligence: Optimization

Aditi Raghunathan

Notation

→ Binary classification $\quad y \in \{-1, +1\}$

→ "linear" classifier. $\quad \theta^T x : \quad w^T x + b$ } Classifier need not pass through origin

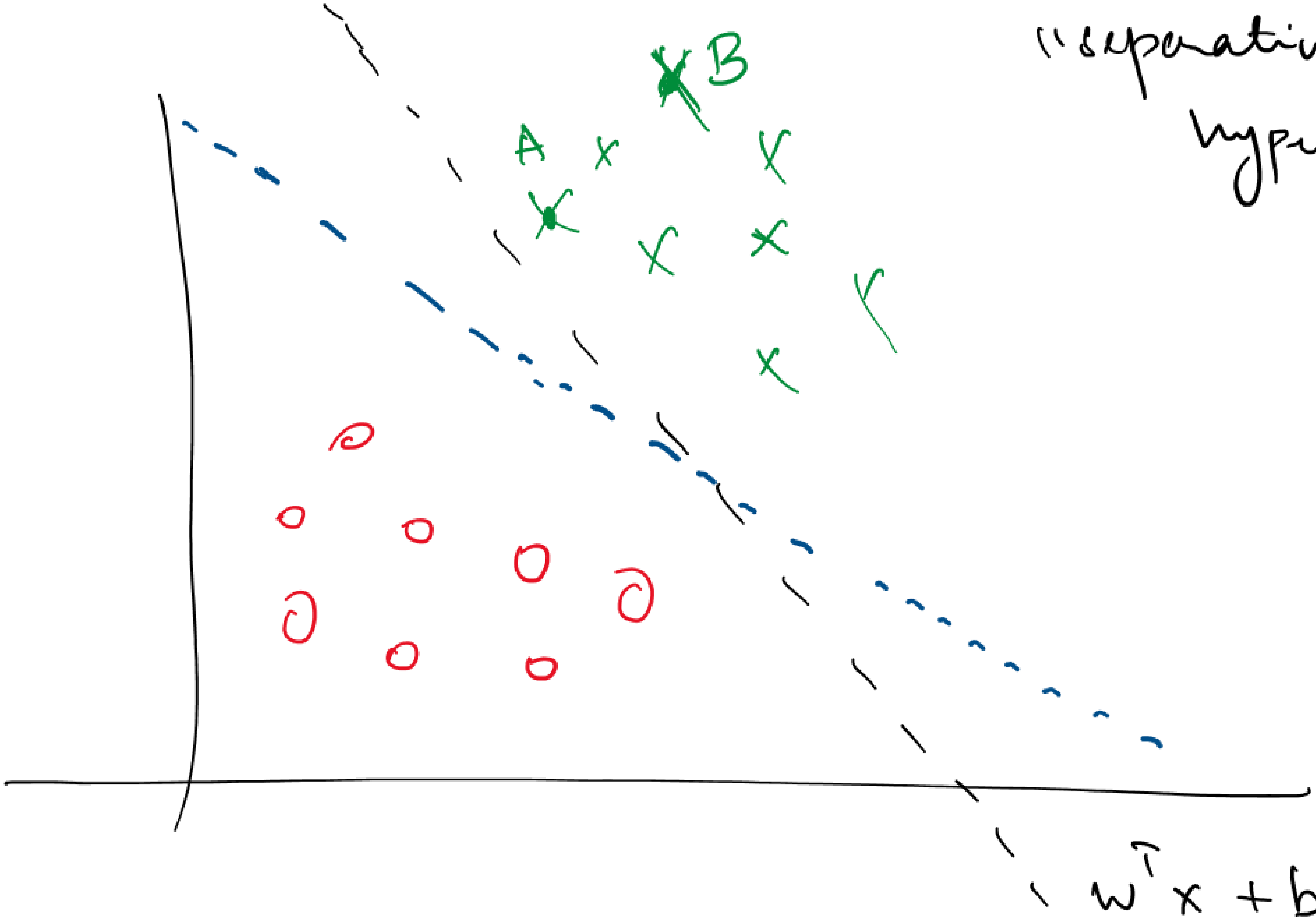
$[\tilde{x}, 1]$ ← bias

→ $\mathcal{H} : \mathcal{X} \rightarrow \mathbb{R}$

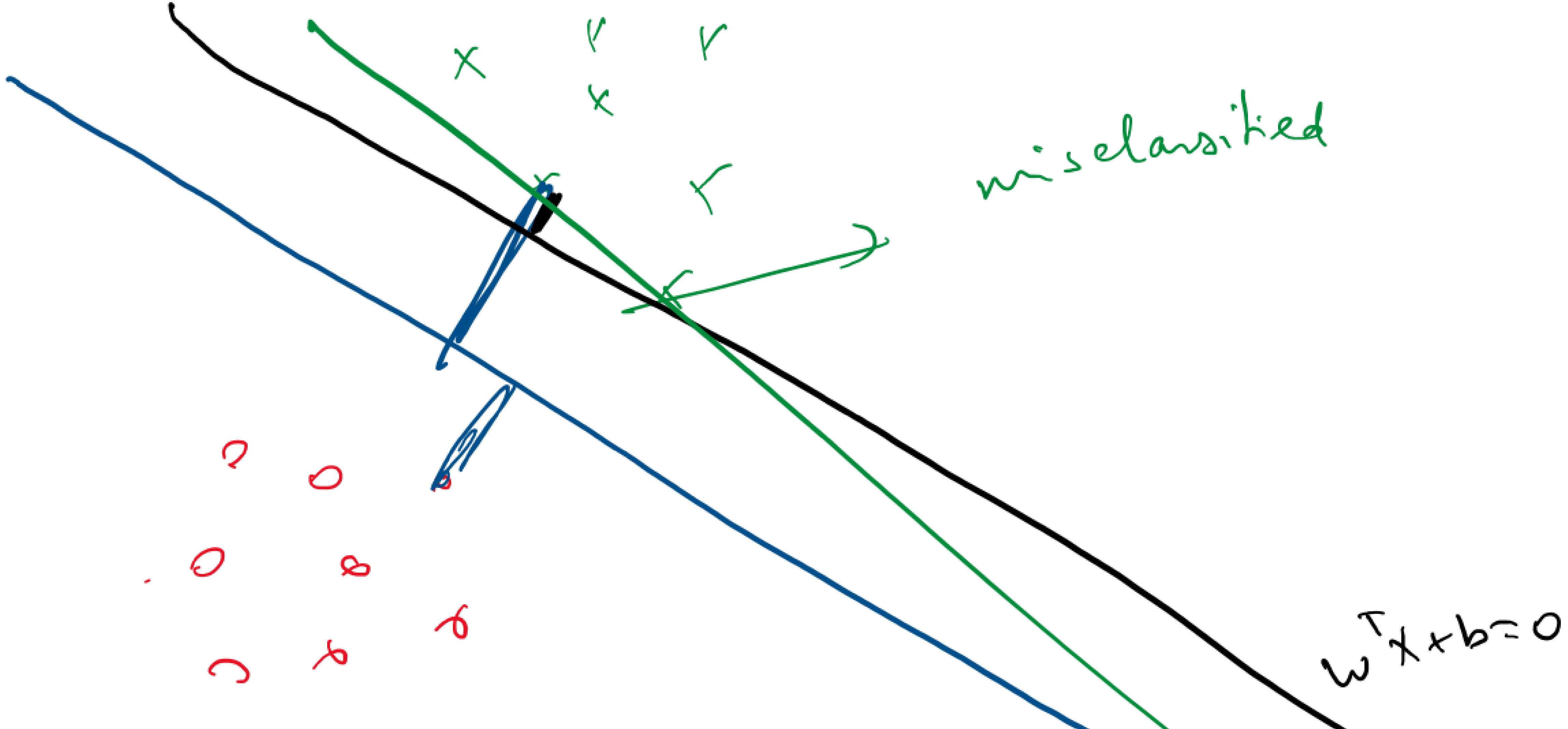
$$h(x) = w^T x + b$$

$$y = \text{sign}(h(x)) = \text{sign}(w^T x + b)$$

"separating hyperplane"



$w^T x + b = 0$



classifier that maximizes the "minimum margin" on train set

What is "margin"?

B is more confident than A

$$\rightarrow \hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

$$\tilde{w}, \tilde{b} = 100 \times w, \ 100 \times b$$

margin increases by 100

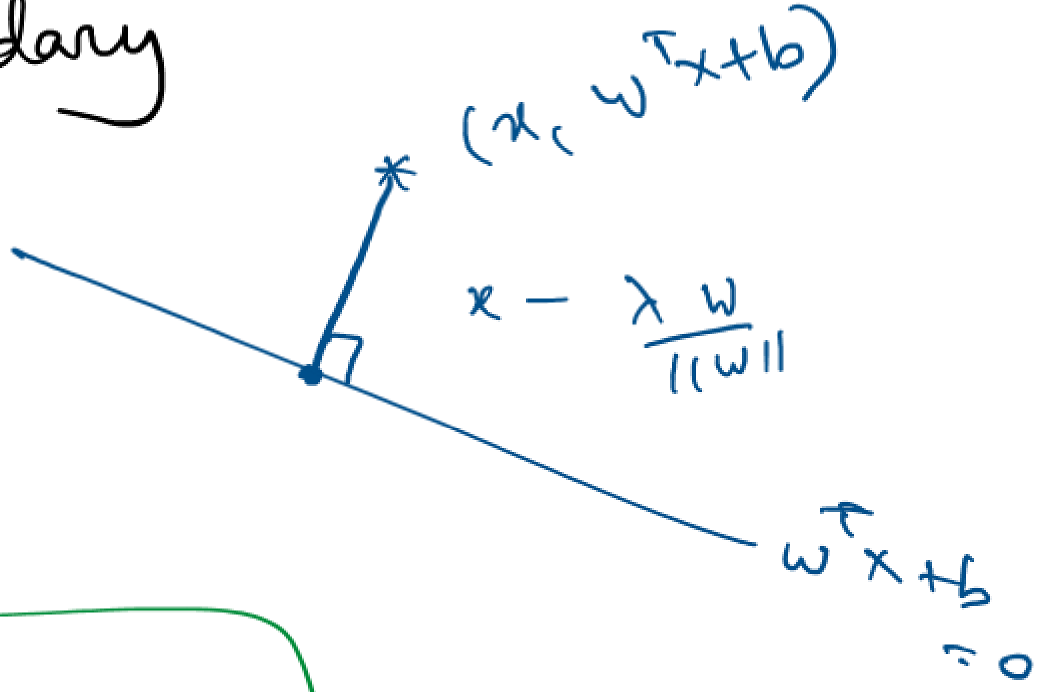
functional margin

geometric margin

Geometric margin

Distance to decision boundary

$$\gamma^{(i)} = y^{(i)}\left(\frac{w^T x^{(i)} + b}{\|w\|}\right)$$



$$\max \ \gamma \quad \text{s.t.}$$

$$y^{(i)}\left(\frac{w^T (x^{(i)}) + b}{\|w\|}\right) \geq \gamma$$

$$\hat{\gamma} = \gamma \cdot \|w\|$$

$\|w\| = 1$: $\max \gamma$

s.t. $y^{(i)}\left(w^T(x^{(i)}+b)\right) \geq \gamma$

$\|w\| = 1$ → hard to optimize!

$\max \dfrac{\hat{\gamma}}{\|w\|}$ → functional margin

in dr ⇒ not a nice fn to optimize

s.t. $y^{(i)}\left(w^T(x^{(i)}+b)\right) \geq \hat{\gamma}$

$\hat{\gamma} = 1$ (functional margin can be arbitrarily scaled)

$$\max \frac{1}{\|w\|} \equiv \min \frac{1}{2}\|w\|_2^2$$

$$\boxed{\begin{aligned} &\min \quad \frac{1}{2}\|w\|_2^2 \\ &\text{s.t.} \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 \end{aligned}}$$

"Non-seperable case"

slack variable $\quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$

$\xi_i \geq 0$

$$\min \boxed{\frac{1}{2}\|w\|_2^2 + C\sum_i \xi_i} \quad \text{convex}$$

s.t. $\quad y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i$

$\xi_i \geq 0$

regularizer

$$\frac{1}{2}\|w\|_2^2 + \text{Trainloss}(y^{(i)}, x^{(i)})$$

hinge-loss

$$\max(1 - y^{(i)}(w^T x + b^{(i)}), 0)$$

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

Opt: Train Loss + λ Regularizer

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

Optimization definitions

We'll write optimization problems like this:

θ : min trainLoss + λreg

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

which should be interpreted to mean: we want to find the value of x that achieves the smallest possible value of $f(x)$, out of all points in $\mathcal{C}$

Optimization definitions

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

Important terms:

- $x \in \mathbb{R}^n$ – optimization variable (vector with n real-valued entries)
- $f: \mathbb{R}^n \to \mathbb{R}$ – optimization objective
- $\mathcal{C} \subseteq \mathbb{R}^n$ – constraint set
- $x^\star \equiv \underset{x \in \mathcal{C}}{\operatorname{argmin}} f(x)$ – optimal solution
- $f^\star \equiv f(x^\star) \equiv \min_{x \in \mathcal{C}} f(x)$ – optimal objective

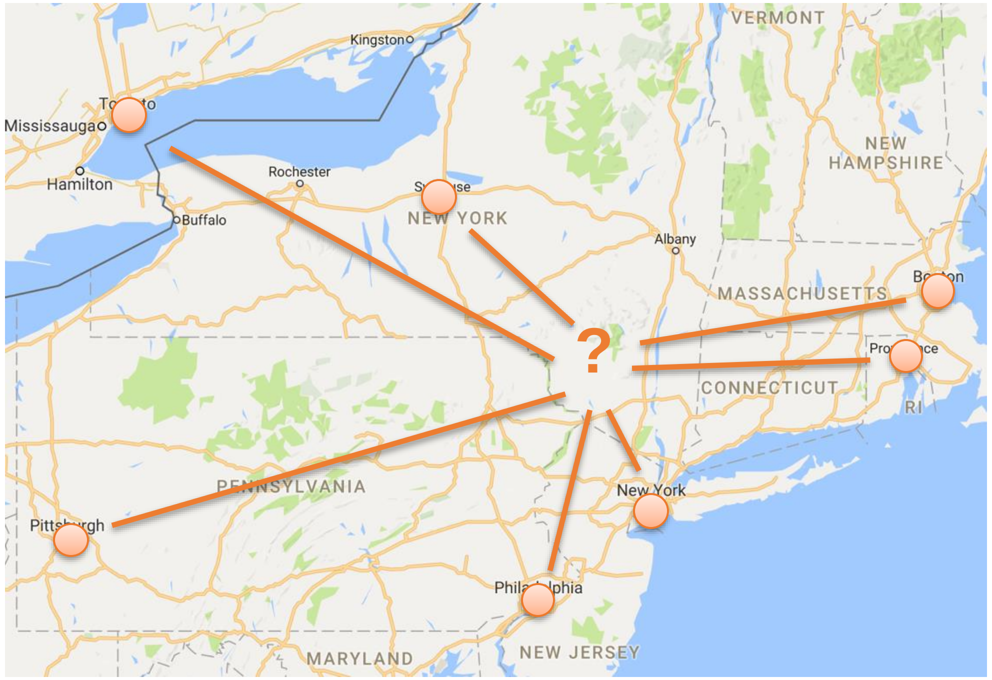
Example: Weber point

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 $y^{(1)}, \ldots, y^{(m)}$

Write as the optimization problem:

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



Example: image deblurring

(a) Original image. (b) Blurry, noisy image. (c) Restored image.

Figure from (O'Connor and Vandenberghe, 2014)

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

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

where $K *$ denotes convolution with a blurring filter

Example: image deblurring

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

$$\underset{X}{\text{minimize}} \underbrace{\sum_{i,j} \left|Y_{ij} - (K * X)_{ij}\right|}_{\textit{Reverse image blurring}} + \lambda \underbrace{\sum_{i,j} \left(\left(X_{ij} - X_{i,j+1}\right)^2 + \left(X_{i+1,j} - X_{ij}\right)^2\right)^{\frac{1}{2}}}_{\textit{"prior" on natural images: nearby pixels are similar}}$$

where $K *$ denotes convolution with a blurring filter

Total variation image deblurring

Example: robot trajectory planning

We want a sequence of control inputs that take a robot from start to goal

Why tricky? Some obstacles, and also can't simply move along any coordinate at will (joint limits)

One approach: model this as a search problem (sample points in the space, discard ones that hit obstacles and

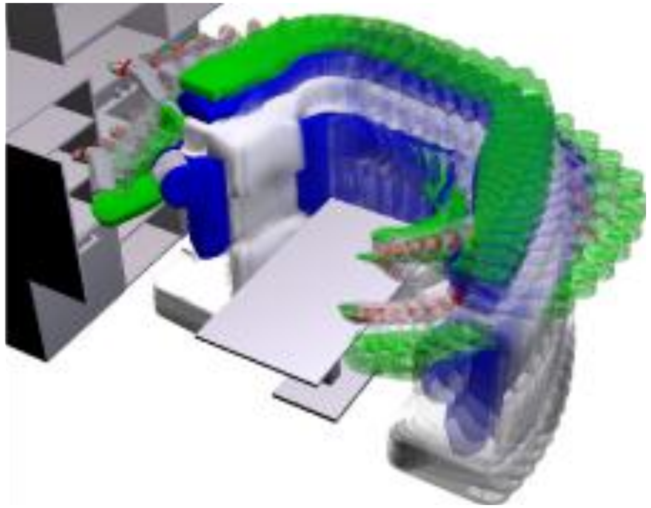


Figure from (Schulman et al., 2014)

Example: robot trajectory planning

Robot state x_t and control inputs u_t

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

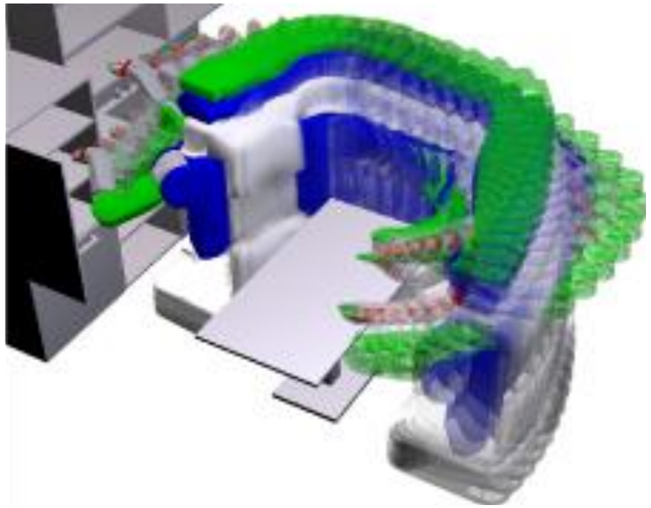


Figure from (Schulman et al., 2014)

Example: learning from examples

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

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) \quad + \lambda \text{ regularizer}$$

Where $x^{(i)} \in \mathcal{X}$ are inputs, $y^{(i)} \in \mathcal{Y}$ are outputs, ℓ is a loss function, ad h_θ is a hypothesis function parameterized by θ, which are the parameters of the model we are optimizing over

Find a model that best fits the observed data

Much more on this from next lecture…

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

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

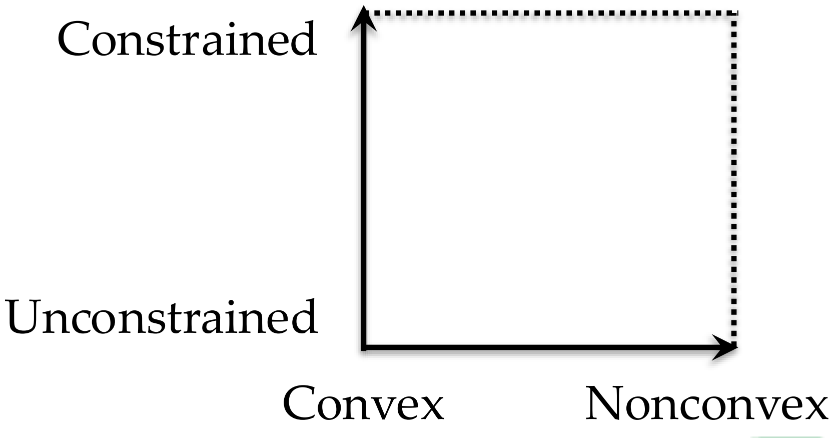
Classes of optimization problems

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

We're instead going to focus on two dimensions: convex vs. nonconvex and constrained vs. unconstrained

Which problems are easy?

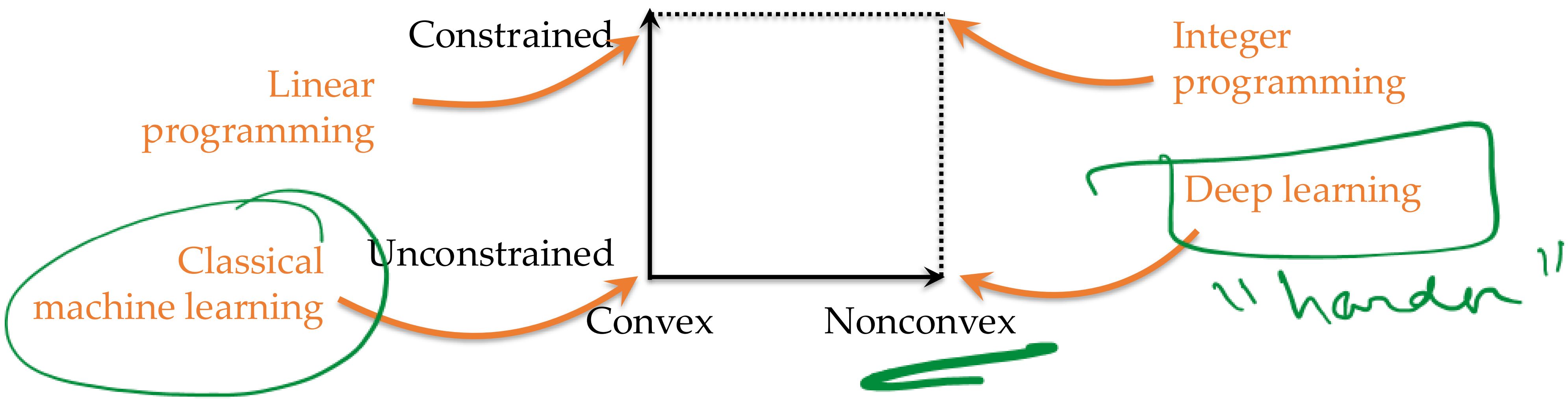
$x \in C$



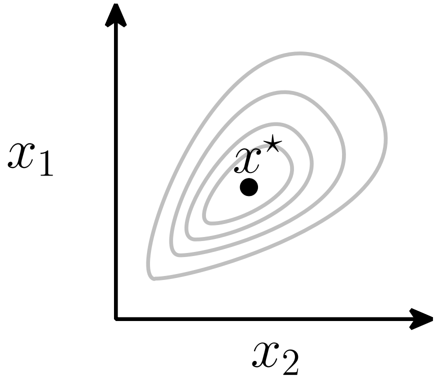
Classes of optimization problems

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

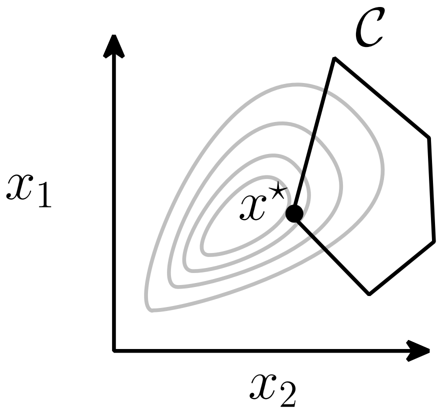
We're instead going to focus on two dimensions: convex vs. nonconvex and constrained vs. unconstrained



Constrained vs. unconstrained



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



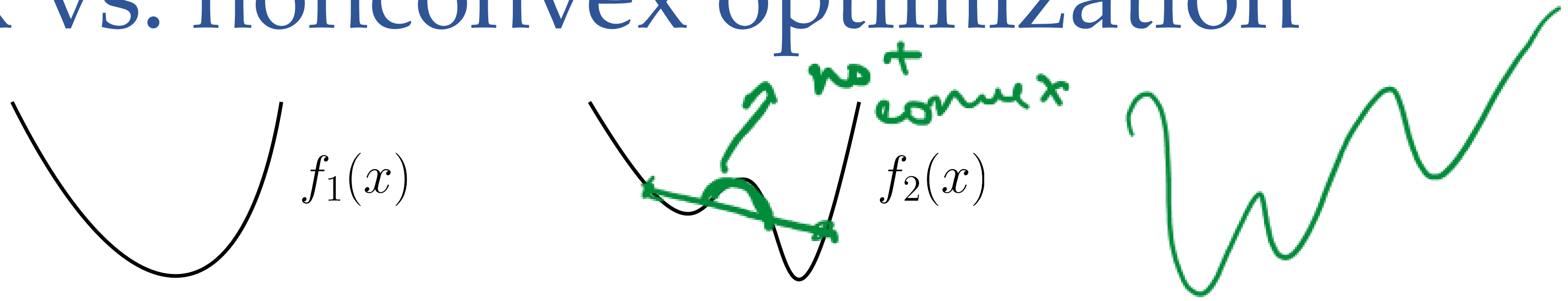
$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

allowed

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, may be hard to even *find* a point $x \in \mathcal{C}$

Convex vs. nonconvex optimization



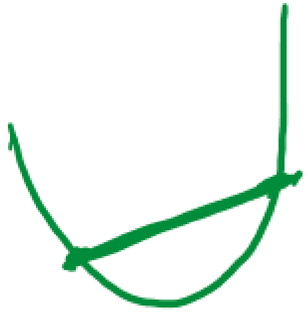
Convex function **Nonconvex function**

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:

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



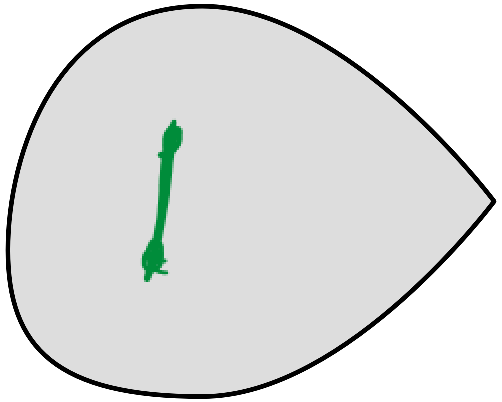
Where f is a convex function and $\mathcal{C}$ is a convex set

Convex sets

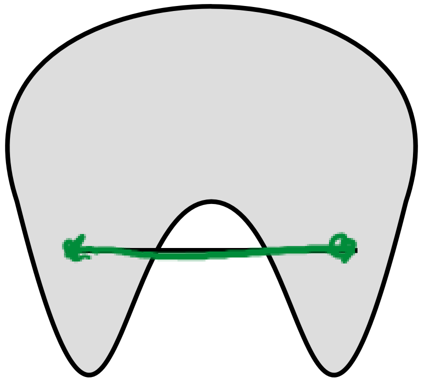
A set $\mathcal{C}$ is convex if, for any $x, y \in \mathcal{C}$ and $0 \leq \theta \leq 1$

$$\theta x + (1-\theta) y \in \mathcal{C}$$

Line segment between two points in the set is also in the set



Convex set



Nonconvex set

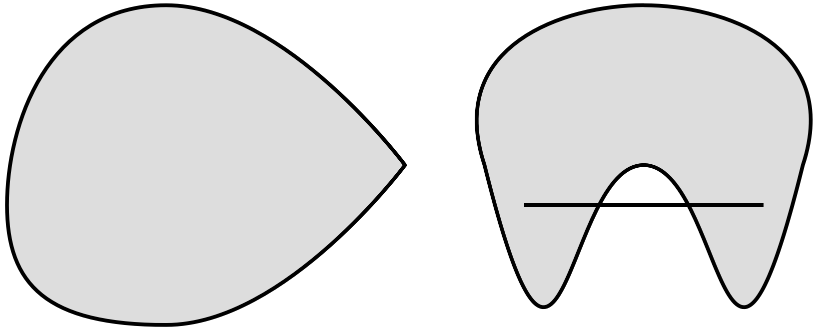
Convex sets

$[x_1 \quad x_2]$ $\lambda x_1 + (1-\lambda) z_1$

$(z_1 \quad z_2)$ $\lambda x_2 + (1-\lambda) z_2$

Examples:

- All points $\mathcal{C} = \mathbb{R}^n$
- Intervals $\mathcal{C} = \{x \in \mathbb{R}^n | \, l \leq x \leq u\}$ (elementwise inequality)
- Linear equalities $\mathcal{C} = \{x \in \mathbb{R}^n | Ax = b\}$ (for $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$)
- Norm balls
- Intersection of convex sets $\mathcal{C} = \bigcap_{i=1}^{m} \mathcal{C}_i$

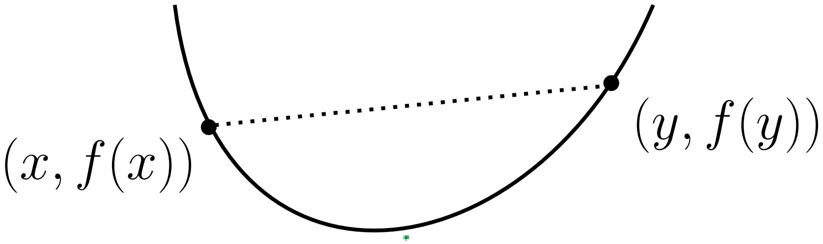


Convex set **Nonconvex set**

Convex functions

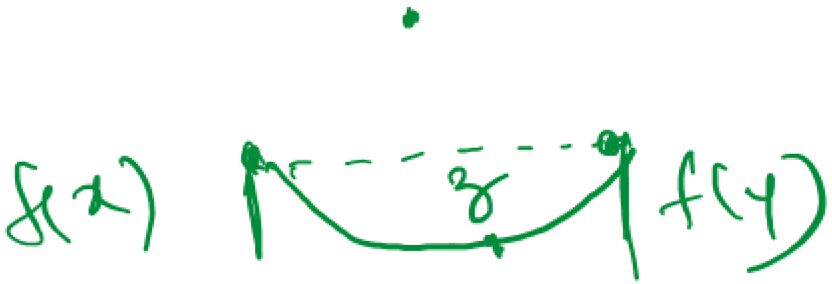
A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if, for any $x, y \in \mathbb{R}^n$ and $0 \leq \theta \leq 1$

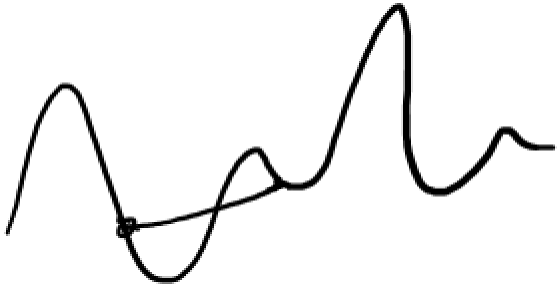
$$\boxed{f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)}$$



Function is below the line joining two points

Convex functions "curve upwards" (or at least not downwards)



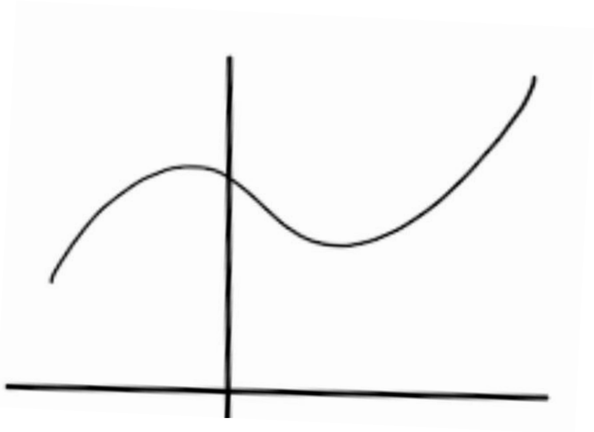


Convex functions

If f is convex then $-f$ is concave (*curves downwards now*)

If f is both convex and concave, it is affine, must be of form:

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



Function can be neither convex nor concave

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

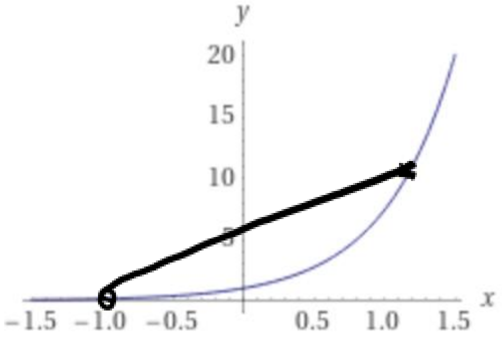
Euclidean norm: $f(x) = \|x\|_2$

$\frac{1}{2}\|w\|_2^2$

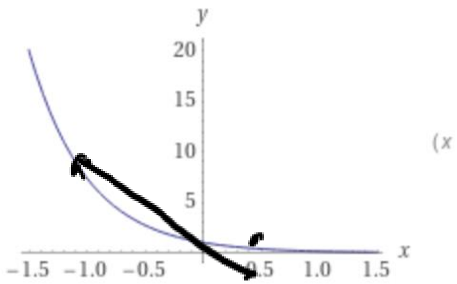
Convex function of affine function $f(x) = g(Ax + b)$, g convex

Non-negative weighted sum of convex functions

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



Exponential



Negative log

Examples of convex functions

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

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

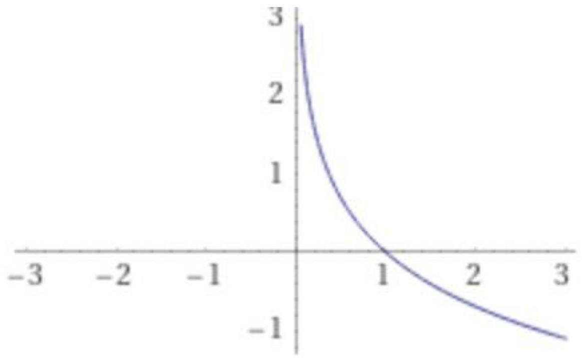
Squared Euclidean norm: $f(x) = \|x\|_2^2 \equiv x^T x \equiv \sum_{i=1}^n x_i^2$

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

Convex function of affine function $f(x) = g(Ax + b)$, g convex

Non-negative weighted sum of convex functions

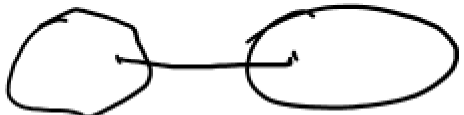
$$f(x) = \sum_{i=1}^{m} w_i f_i(x), \qquad w_i \geq 0, f_i \text{ convex}$$



Negative log

Convex sets and functions: piazza

$\theta x + (1-\theta) y \in C \quad \forall\, x, y \in C$

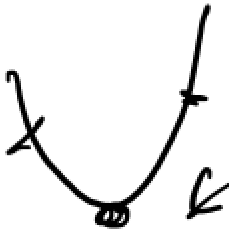


1. The union of two convex sets $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ Not convex
2. The set $\{x \in \mathbb{R}^2 \mid x \geq 0, x_1 x_2 \geq 1\}$
3. The function $f: \mathbb{R}^2_+ \to \mathbb{R}, \underline{f(x) = x_1 x_2}$ with domain $x > 0$
4. The function $f: \mathbb{R}^2 \to \mathbb{R}, f(x) = x_1^2 + x_2^2 + x_1 x_2$

counter example

$(2,4) \quad (4,2)$

$(3,3) = 9$



not $\leq$

$$\underbrace{\frac{1}{2}(x_1 + x_2)^2}_{\text{convex}} + \underbrace{\frac{1}{2}x_1^2}_{\text{convex}} + \underbrace{\frac{1}{2}x_2^2}_{\text{convex}}$$

Convex optimization

The key aspect of convex optimization problems that make them tractable is that *all local optima are global optima*

Convex optimization

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 there is no feasible y with $\|x - y\|_2 \leq R$, $f(y) < f(y)$

Theorem: for a convex optimization problem all locally optimal points are globally optimal

Picture proof

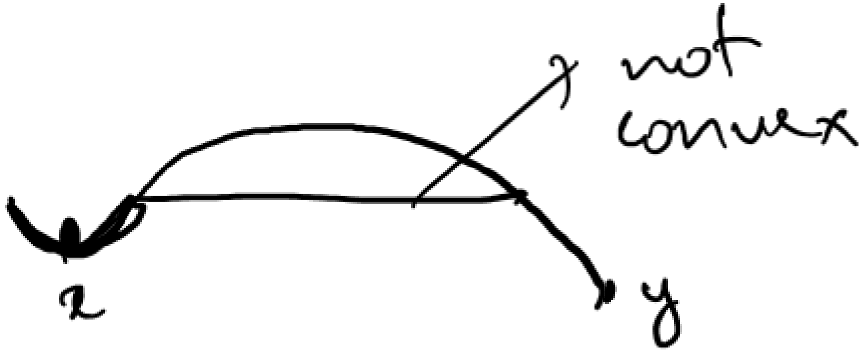
local optimality $\Rightarrow$ global optimality

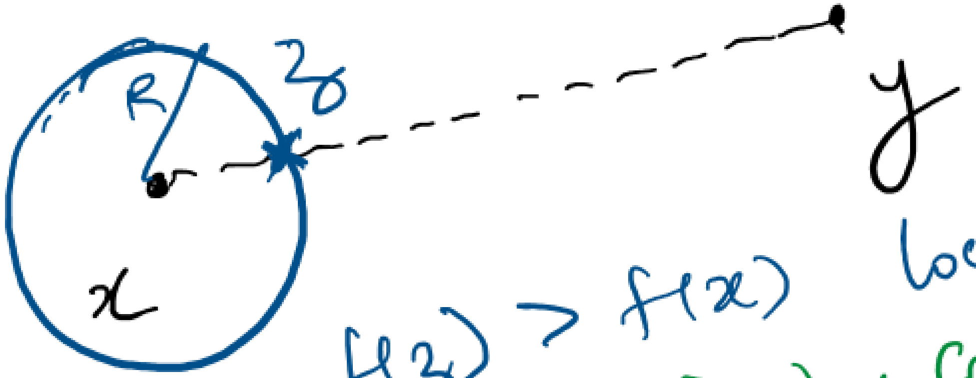
Suppose x is locally opt but not global

x, $f(x)$

y, $f(y)$

$f(y) < f(x)$





local opt

$f(z) > f(x)$

by convexity $f(z) < f(x)$

$\Rightarrow$ contradiction

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

pt in the ball around x

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

z

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:
$$f(z) = f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) < f(x)$$
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

■

Local optimality => global optimality

If you are not at the minima of a convex function, there is a direction that reduces your function value locally

We used both convexity of feasible set and convexity of objective

Outline

Introduction to optimization

Types of optimization problems, convexity

Solving optimization problems

The gradient

A key concept in solving optimization problems is the notation of the gradient of a function (multi-variate analogue of derivative)

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

Partial derivative: A partial derivative of a function of several variables is derivative with respect to one of those variables with rest constant

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x + h\mathbf{e_i}) - f(x)}{h}$$

only i^{th} coordinate is changed by h, rest are unchanged

$f(x_1, x_2) = x_1^2 + 2x_2^3$

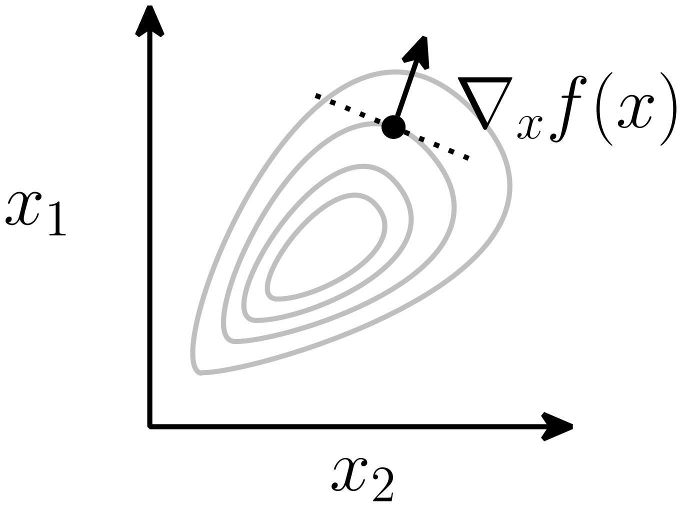
∇f at $(1, 1)$

The gradient

$$\nabla f = \begin{bmatrix} 2x_1 = 2 \\ 6x_2^2 = 6 \end{bmatrix}$$

For $f: \mathbb{R}^n \to \mathbb{R}$, gradient is defined as vector of partial derivatives

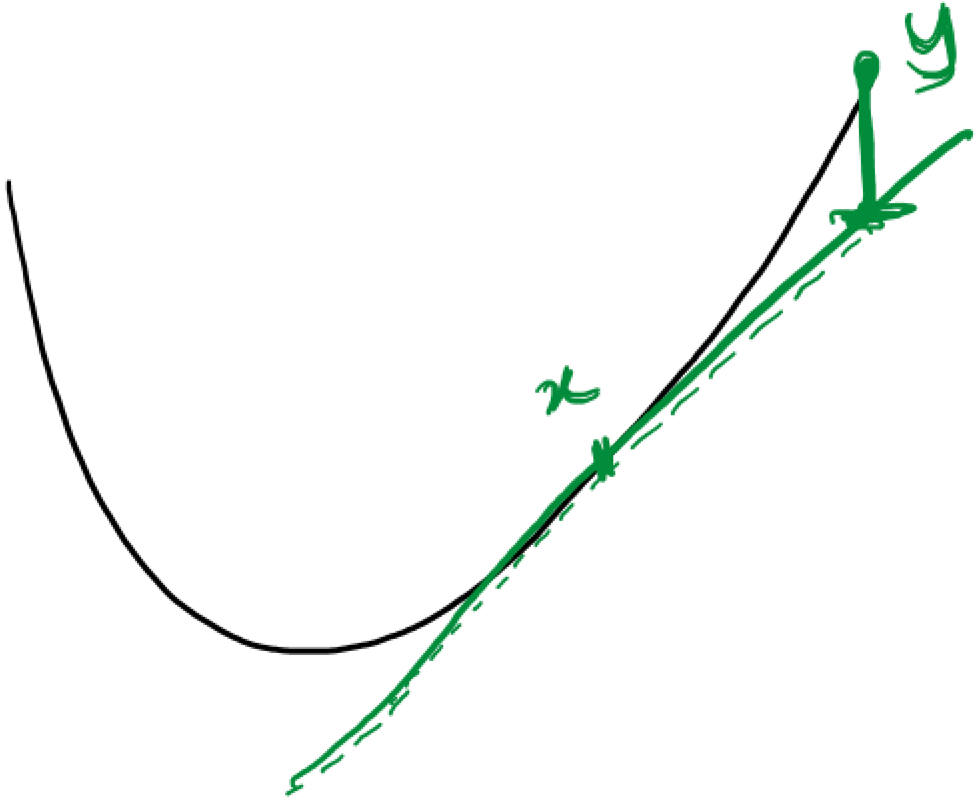
$$\nabla_x f(x) \in \mathbb{R}^n = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

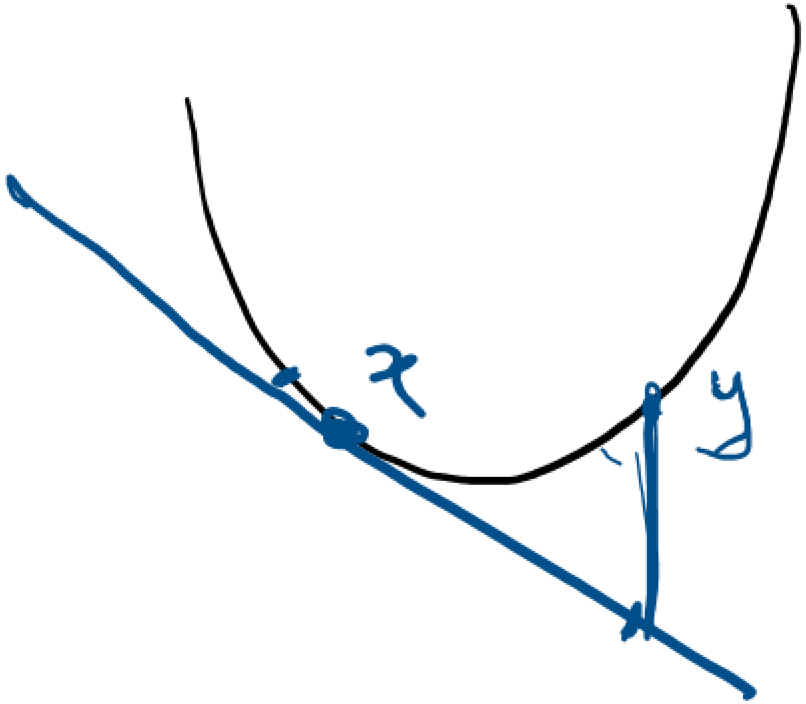


Points in “steepest direction” of increase in function f

First-order condition for convexity

$$f(y) \geq f(x) + (y - x)^\top \nabla f(x)$$

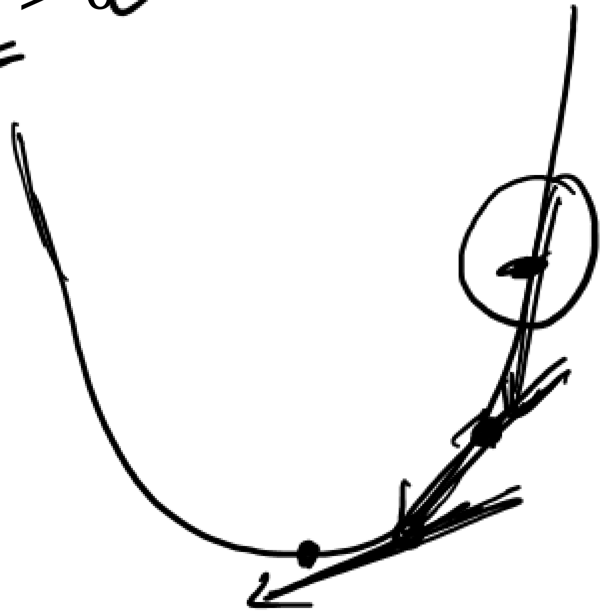




Gradient descent

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

Algorithm: Gradient Descent
Given:
  Function f, initial point x_0, step size $\cancel{\alpha}\ \eta > 0$
Initialize:
  $x := x_0$
Repeat until convergence:
  $x := x - \eta\,\cancel{\alpha} \nabla_x f(x)$



"Convergence" can be defined in a number of ways

Why does gradient descent work?

Smooth function: Derivative exists everywhere, and gradient is L-Lipschitz

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

Taylor expansion: $f(y) \leq f(x) + (y-x)^T \nabla_x f(x)$

$+ \underbrace{\frac{1}{2}\|\nabla^2 f(\xi)\|_2^2}_{} \leq \frac{L}{2}\|y-x\|_2^2$

x_{t+1} $\qquad x_t$

$$\boxed{f(y) \leq f(x) + (y - x)^\top \nabla_x f(x) + \frac{L}{2}\|y - x\|_2^2}$$

Why does gradient descent work?

Smooth function: Derivative exists everywhere, and gradient is L-Lipschitz

$$\boxed{x_{t+1} = x_t - \eta \nabla f(x_t)} \qquad \eta = \frac{1}{L}$$

$y = x_{t+1}$

$x = x_t$

from prev slide

$$f(x_{t+1}) \le f(x_t) - \eta \nabla f(x_t)^\top \nabla f(x_t) + \frac{L\eta^2}{2} \|\nabla f(x_t)\|_2^2$$

$$\boxed{f(x_{t+1}) \le f(x_t) - \frac{1}{2L} \|\nabla f(x_t)\|_2^2}$$

Function value decreases

Why does gradient descent work?

Convexity: First-order condition $f(y) \geq f(x) + (y-x)^\top \nabla f(x)$

$$\underbrace{f(x^\star) \geq f(x_t) + (x^\star - x_t)^\top \nabla f(x_t)}_{\text{Convexity}} \qquad \underbrace{f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|_2^2}_{\text{Smoothness}}$$

x^*: optimum

Why does gradient descent work?

Convexity: First-order condition $\quad f(y) \geq f(x) + (y-x)^\top \nabla f(x)$

$$\underbrace{f(x^\star) \geq f(x_t) + (x^\star - x_t)^\top \nabla f(x_t)}_{\text{Convexity}} \qquad \underbrace{f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|_2^2}_{\text{Smoothness}}$$

$$\|x_{t+1} - x^\star\|_2^2 = \|x_t - x^\star\|_2^2 + \frac{1}{L^2}\|\nabla f(x_t)\|_2^2 + \frac{2}{L}(x^\star - x_t)^\top \nabla f(x_t)$$

$u: x_t - x^*$

$v: \eta \nabla f(x_t)$

$$x_{t+1} = x_t - \eta \nabla f(x_t) = x_t - \frac{1}{L} \nabla f(x_t)$$

"no magic"

$$\|u-v\|_2^2 = \|u\|_2^2 + \|v\|_2^2 + 2u^\top v$$

Why does gradient descent work?

Convexity: First-order condition $f(y) \geq f(x) + (y-x)^\top \nabla f(x)$

$$\underbrace{f(x^\star) \geq f(x_t) + (x^\star - x_t)^\top \nabla f(x_t)}_{\text{Convexity}} \qquad \underbrace{f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|_2^2}_{\text{Smoothness}}$$

smoothness

$$\|x_{t+1} - x^\star\|_2^2 = \|x_t - x^\star\|_2^2 + \frac{1}{L^2}\|\nabla f(x_t)\|_2^2 + \frac{2}{L}(x^\star - x_t)^\top \nabla f(x_t)$$ convexity

$$f(x_{t+1}) \leq f(x^\star) + \frac{L}{2}\left(\|x_t - x^\star\|_2^2 - \|x_{t+1} - x^\star\|_2^2\right)$$ → telescopic sums

$$f(x_{t+1}) \le f(x^*) + \frac{L}{2}\left(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2\right)$$

summing over $t = 1$ to T

$x_1 - x^*$ $\quad$ $x_2 \cancel{- x^*}$

$x_2 \cancel{- x^*}$ $\quad$ $x_3 \cancel{- x^*}$

$$f(x_1) + f(x_2) \dots f(x_T) \le T f(x^*) + \frac{L}{2}\left(\|x_0 - x^*\|_2^2 - \|x_T - x^*\|^2\right)$$

$$f(x_1) + f(x_2) \dots f(x_T) \le T f(x^*) + \frac{L}{2}\left(\|x_0 - x^*\|_2^2\right)$$

$$f(x_T) \le f(x_t) \ \forall \ t$$

Why does gradient descent work?

Convexity: First-order condition $f(y) \geq f(x) + (y-x)^\top \nabla f(x)$

$$\underbrace{f(x^\star) \geq f(x_t) + (x^\star - x_t)^\top \nabla f(x_t)}_{\text{Convexity}} \qquad \underbrace{f(x_{t+1}) \leq f(x_t) - \frac{1}{2L}\|\nabla f(x_t)\|_2^2}_{\text{Smoothness}}$$

$$\|x_{t+1} - x^\star\|_2^2 = \|x_t - x^\star\|_2^2 + \frac{1}{L^2}\|\nabla f(x_t)\|_2^2 + \frac{2}{L}(x^\star - x_t)^\top \nabla f(x_t)$$

$$f(x_{t+1}) \leq f(x^\star) + \frac{L}{2}\left(\|x_t - x^\star\|_2^2 - \|x_{t+1} - x^\star\|_2^2\right)$$

$$\boxed{f(x_T) - f(x^\star) \leq \frac{L}{2T}(\|x_0 - x^\star\|_2^2)}$$

as T increases
$f(x_T)$ gets closer
to $f(x^*)$

Why does gradient descent work?

Function value decreases via smoothness assumption

Combining smoothness with convexity, with telescoping sums, we get

$$\boxed{f(x_T) - f(x^\star) \leq \frac{L}{2T}(\|x_0 - x^\star\|_2^2)}$$

When does gradient descent work?

Do you need smoothness?

What we really needed to show function value decreases:

$$f(y) \geq f(x) + (y - x)^\top \nabla f(x)$$

When does gradient descent work?

Do you need smoothness?

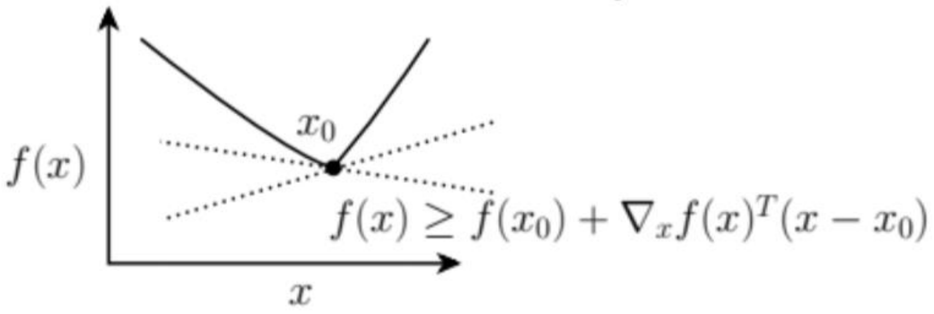
What we really needed to show function value decreases:

$$f(y) \geq f(x) + (y - x)^\top \nabla f(x)$$

Can use subgradients instead

$$f(y) \geq f(x) + g^\top (y - x)$$

A subgradient is something "like" a gradient, in that for a convex function it must lie below the function everywhere



Example: $f(x) = |x|$, subgradients are given by

$$\nabla_x f(x) = \begin{cases} -1 & x < 0 \\ 1 & x > 0 \\ g \in [0, 1] & x = 0 \end{cases}$$

When does gradient descent work?

Do you need smoothness? What we really needed to show function value decreases:

$$f(y) \geq f(x) + (y-x)^\top \nabla f(x)$$

Can use subgradients instead

$$f(y) \geq f(x) + g^\top (y-x)$$

Method: subgradient "descent"

Need a decreasing sequence of step sizes…

Theory and practice of convergence is quite different

When does gradient descent work?

Do you need convexity?

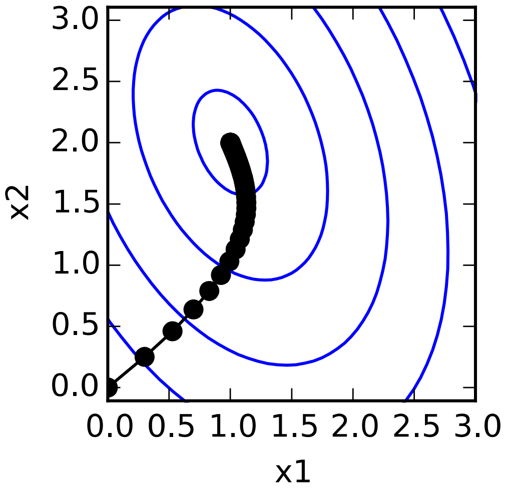
Local optimality no longer implies global optimality

Works surprisingly well in practice…

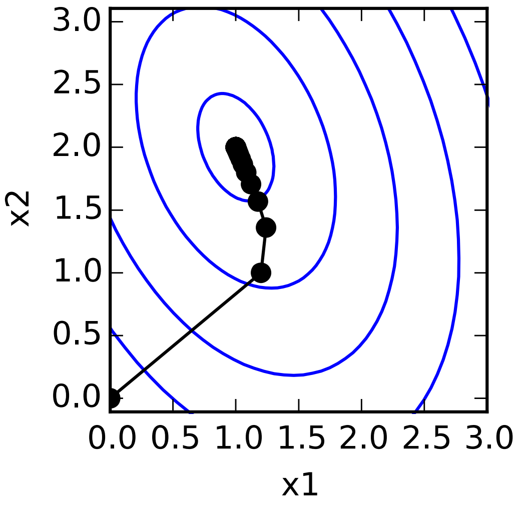
Gradient descent in practice

Choice of α matters a lot in practice:

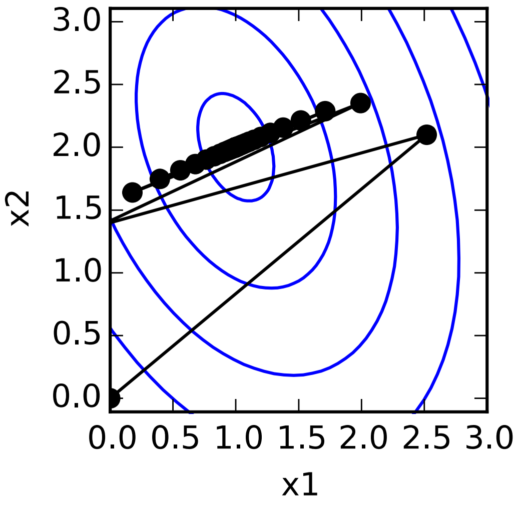
$$\underset{x}{\text{minimize}}\ \ 2x_1^2 + x_2^2 + x_1x_2 - 6x_1 - 5x_2$$



$\alpha = 0.05$



$\alpha = 0.2$



$\alpha = 0.42$

Dealing with constraints

For settings where we can easily project points onto the constraint set $\mathcal{C}$, can use a simple generalization called *projected gradient descent*

$$\text{Repeat: } x \coloneqq P_{\mathcal{C}}\left(x - \alpha \nabla_x f(x)\right)$$

Optimization in practice

We won't discuss this too much yet, but one of the beautiful properties of optimization problems is that there exists a wealth of tools that can solve them using very simple notation

Example: solving Weber point problem using cvxpy (http://cvxpy.org)

```
import numpy as np
import cvxpy as cp

n,m = (5,10)
y = np.random.randn(n,m)
x = cp.Variable(n)
f = sum(cp.norm2(x - y[:,i]) for i in range(m))
cp.Problem(cp.Minimize(f), []).solve()
```