

Where does Mathematics Stand in the Era of AI?

An Optimization Perspective

Wai-Yiu Keung

Department of Computer Science and Engineering



July 17, 2023

Acknowledgement: Prof. Hoi-To Wai

Slides: www.cse.cuhk.edu.hk/~wykeung/slides/ai-math.pdf

Today's Agenda

- **Motivation**
- **Preliminaries**
 - Derivatives
 - Matrix Algebra
- **Mathematical Optimization**
- **A.I. Applications**
 - Pattern Classification
 - Blind Source Separation

Disclaimer: In this talk, I will try to bring in as less math./eqn. as possible. Consequently, some descriptions may not be as accurate as you may see in the literature.
























Have you used them before?



Sourced from the respective company's official websites.

- you won't be surprised if I tell you they all rely on A.I. technologies
- but what if I say they all rely on maths.?

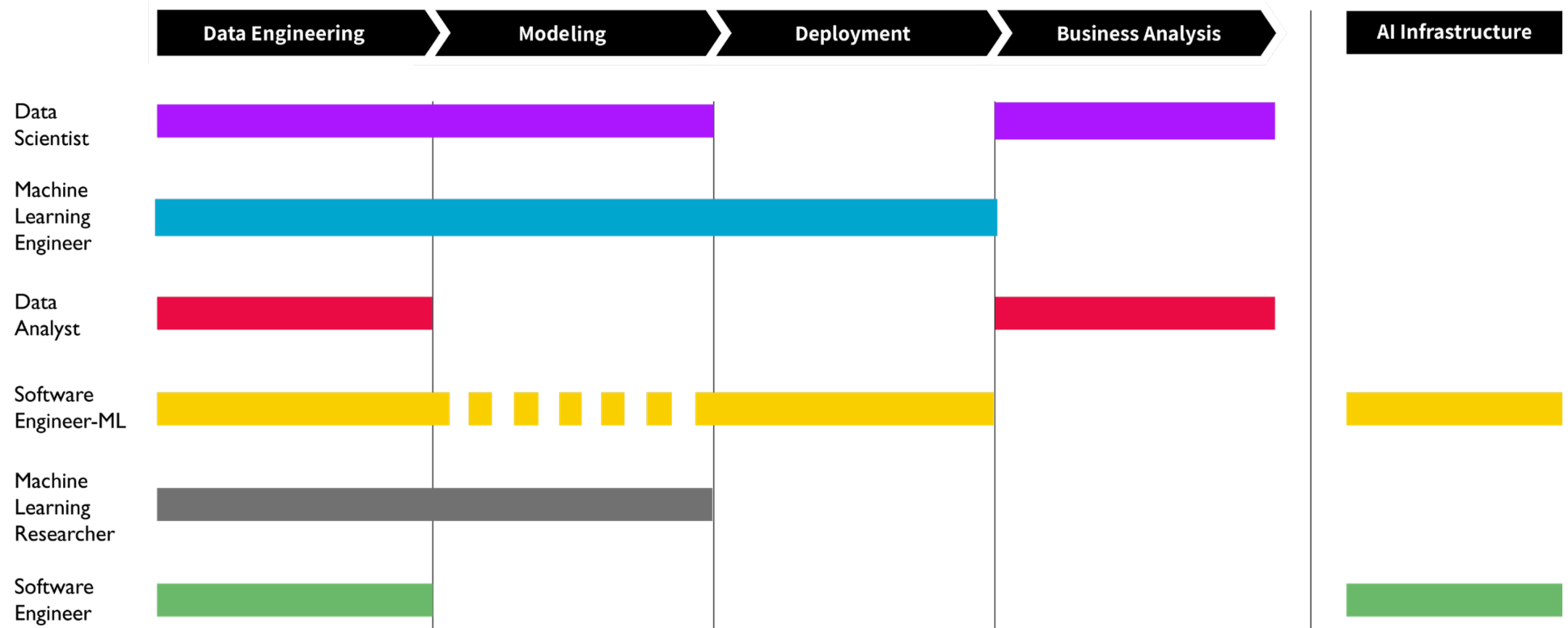
The Job Market

 Machine Learning Engineer Snorkel AI Palo Alto, CA via Lever	 Data Science Engineering, Machine Learning Salesforce Palo Alto, CA via Glassdoor	 Data Scientist / Machine Learning Specialist Genuent Global, LLC London, CA (+1 other) via Dice	 Full Stack Data Engineer Flexion San Ramon, CA via WFH Jobs
 Machine Learning Scientist Amazon Sunnyvale, CA via LinkedIn	 Data Science Engineer, Recent Graduate eBay San Jose, CA via EBay - EBay Inc.	 Senior Product Data Scientist Bill.com Palo Alto, CA via Lever	 Full Stack Data Scientist, Supply DS Wish San Francisco, CA via SmartRecruiters Jobs
 Machine Learning Researcher Samsung Research America Mountain View, CA (+1 other) via Greenhouse	 Machine Learning Research Scientist Capital One San Francisco, CA via Capital One Careers	 Product Analyst, Data Science Google Mountain View, CA via Google Careers	 AI/ML Full Stack Engineer (Quality Engineering) Apple Cupertino, CA via WDTN Jobs
 Data Scientist Waymo Mountain View, CA via Glassdoor	 Senior Machine Learning Analyst Google Sunnyvale, CA via Google Careers	 Senior Business Data Analyst Intuit Corp Mountain View, CA via Intuit Jobs	 Software Engineer / Data Scientist, Fleet Analytics Tesla Palo Alto, CA via LinkedIn
 Data Analyst Snowflake San Mateo, CA via Snowflake Careers	 Research Data Analyst Stanford University Stanford, CA (+1 other) via Inside Higher Ed Careers		 AI Deep Learning Software Engineer Intel Santa Clara, CA via TOPBOTS
 Software Engineer Electronic Arts Inc. Redwood City, CA via Glassdoor	 Machine Learning Research Engineer Embark Trucks, Inc. San Francisco, CA via ZipRecruiter		 Machine Learning Modeler, Cash App Square San Francisco, CA via SmartRecruiters Jobs
	 Software Engineer, Machine Learning, Research Waymo Mountain View, CA via Waymo		

Source: <https://cs230.stanford.edu/>

- go to Google and search for jobs relevant to machine learning/data science
- the job market is going wild on the A.I. industries

Skill Set Requirement for Different A.I. Positions



Source: https://skills.workera.ai/resource_downloads/ai_career_pathways

- let's also look at the basic requirements of different positions
- **data = math!**

Motivation: Admission Requirement

Percentile	CHI	ENG	MATHS	LS	M1/M2	1 st Elective	2 nd Elective	3 rd Elective	Total Reference Score ^
Upper Quartile	4	5**	5**	4	5*	5*	5*	5*	32
Median	3	5	5**	3	5**	5*	5	5	30
Lower Quartile	3	5	5**	4	5	5*	5*	5	29

Source: https://www.cse.cuhk.edu.hk/wp-content/uploads/admission/InfoDay_AIST2022-23.pdf

- some statistics of the JUPAS admission scores of the A.I. programme offered in CUHK
- almost all our freshmen have strong mathematics background

Preliminary: Differentiation

Functions of Interests

- we start by recalling the geometry of one dimensional linear and quadratic functions
- roughly speaking, a function f is a map that leads a domain \mathcal{X} to another \mathcal{Y} ; a domain is simply a collection of numerical objects
- **example:** let $\mathcal{X} = \mathbb{R}$ be an input domain, the linear function

$$f(x) = \underbrace{A}_{\text{slope}} x + \underbrace{B}_{\text{intercept}}$$

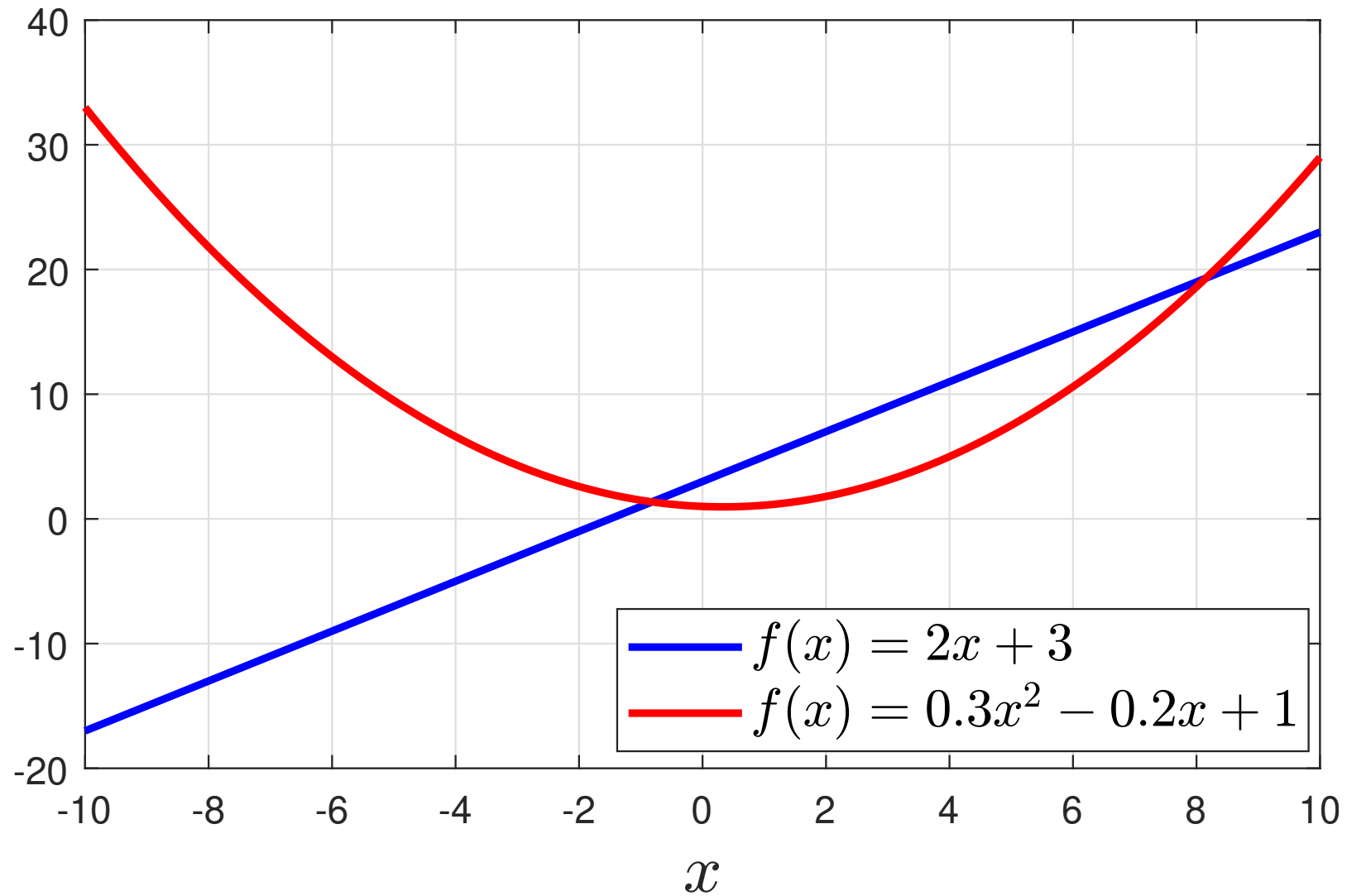
leads to the output domain of $\mathcal{Y} = \mathbb{R}$ as well; often we will use the form $y = f(x)$ to describe the above relationship

- **example:** same settings as above but replace

$$f(x) = \underbrace{A_2}_{\text{quad. term}} x^2 + \underbrace{A_1}_{\text{lin. term}} x + \underbrace{B}_{\text{intercept}} ;$$

it also has the output set $\mathcal{Y} = \mathbb{R}$

Function Types of Interests



Univariate Differentiation

- (with mild assumptions) let $f(x)$ be a function that maps \mathbb{R} to \mathbb{R} , define

$$\phi(t) = \frac{f(t) - f(x)}{t - x}, \quad t \in \mathbb{R}, t \neq x,$$

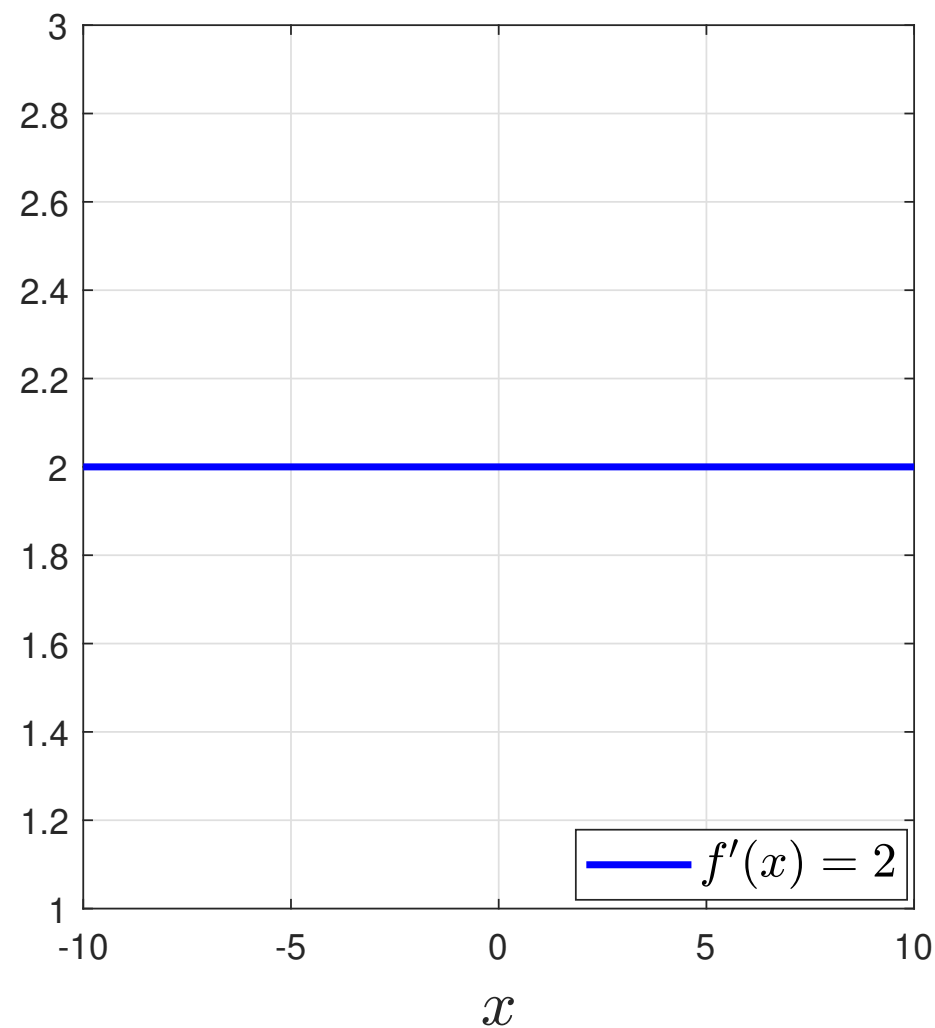
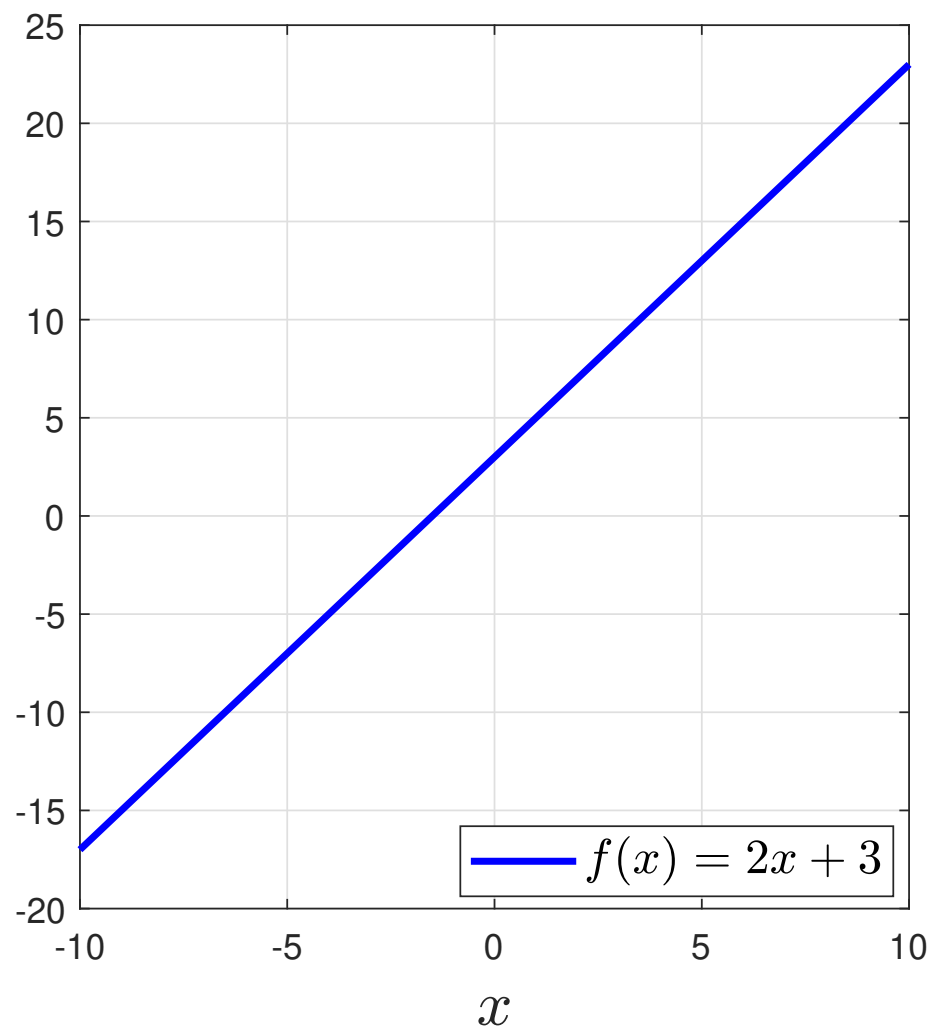
the **derivative** of $f(x)$ is defined as $f'(x) = \lim_{t \rightarrow x} \phi(t)$

- example: $f(x) = Ax + B$ is a linear function; following the above:

$$f'(x) = \lim_{t \rightarrow x} \phi(t) = \lim_{t \rightarrow x} \frac{(At + B) - (Ax + B)}{t - x} = \lim_{t \rightarrow x} \frac{A(t - x)}{(t - x)} = \lim_{t \rightarrow x} A = A$$

- **physical interpretation:** $f'(x)$ captures the slope of $f(x)$

Example: $f'(x)$ is the slope of $f(x)$



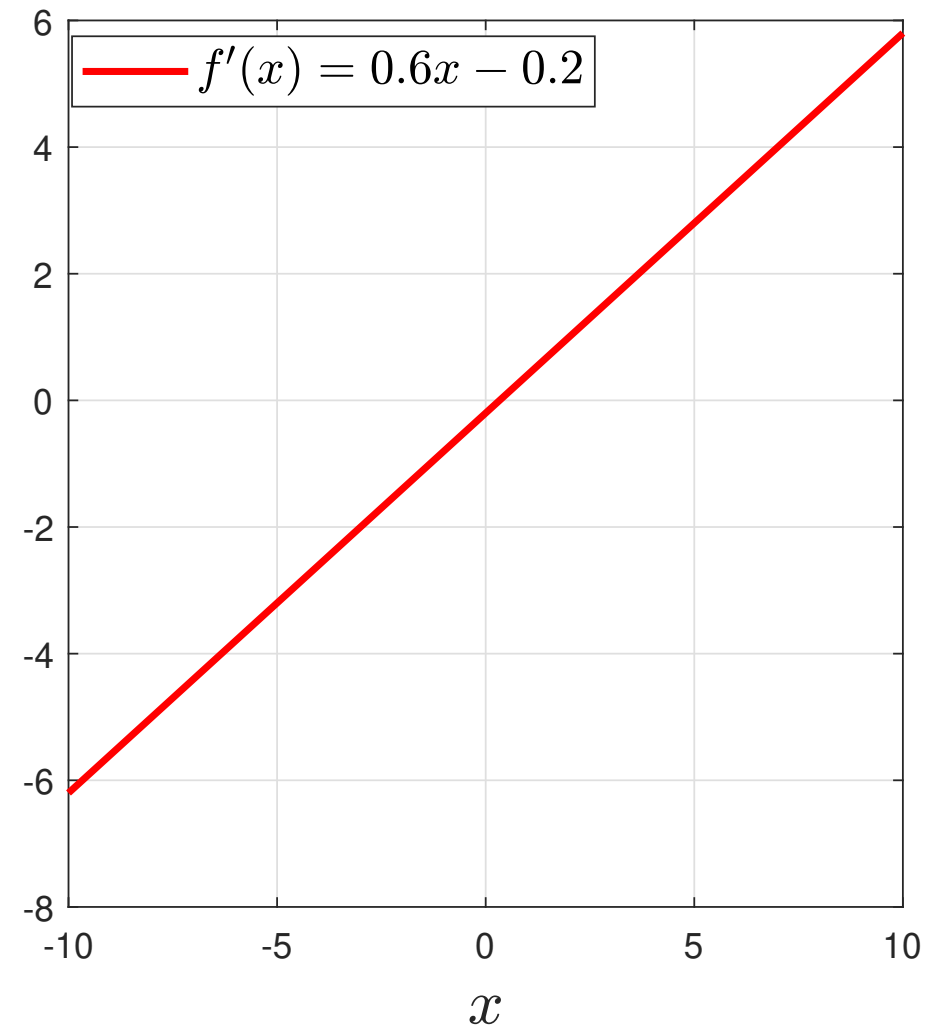
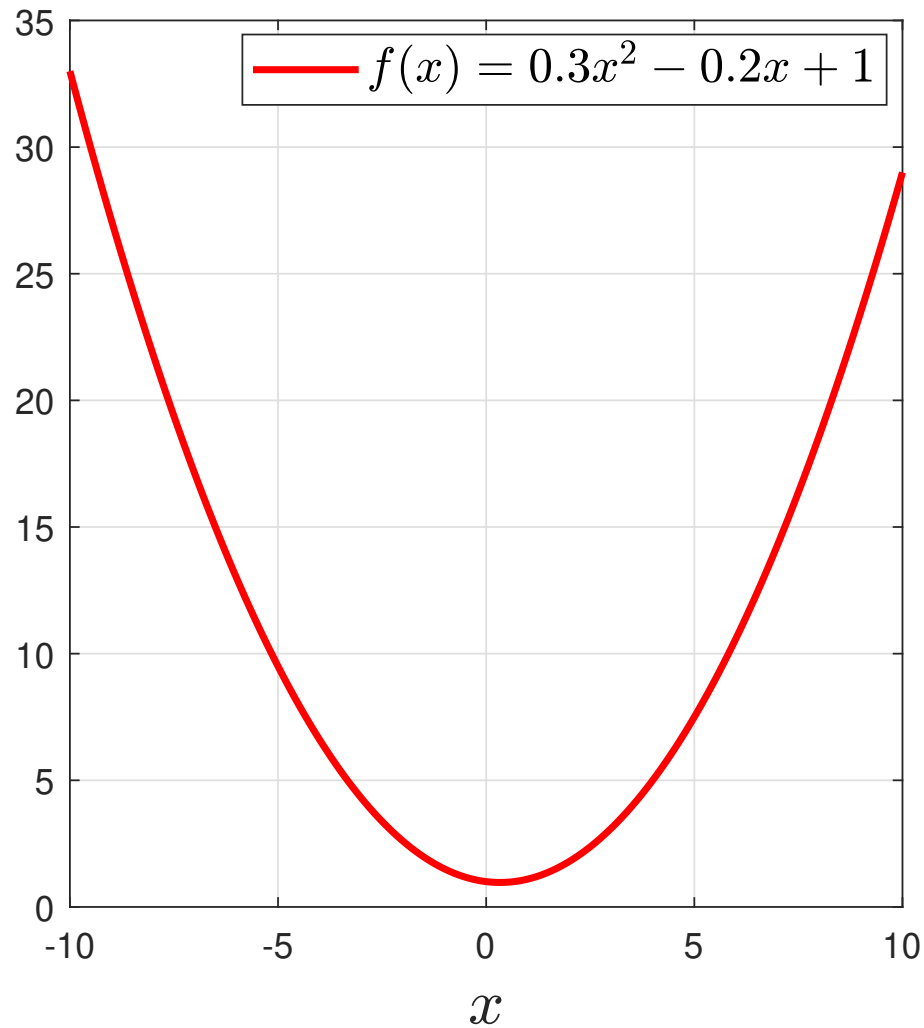
Univariate Differentiation

- example: $f(x) = A_2x^2 + A_1x + B$ is quadratic, it's derivative is

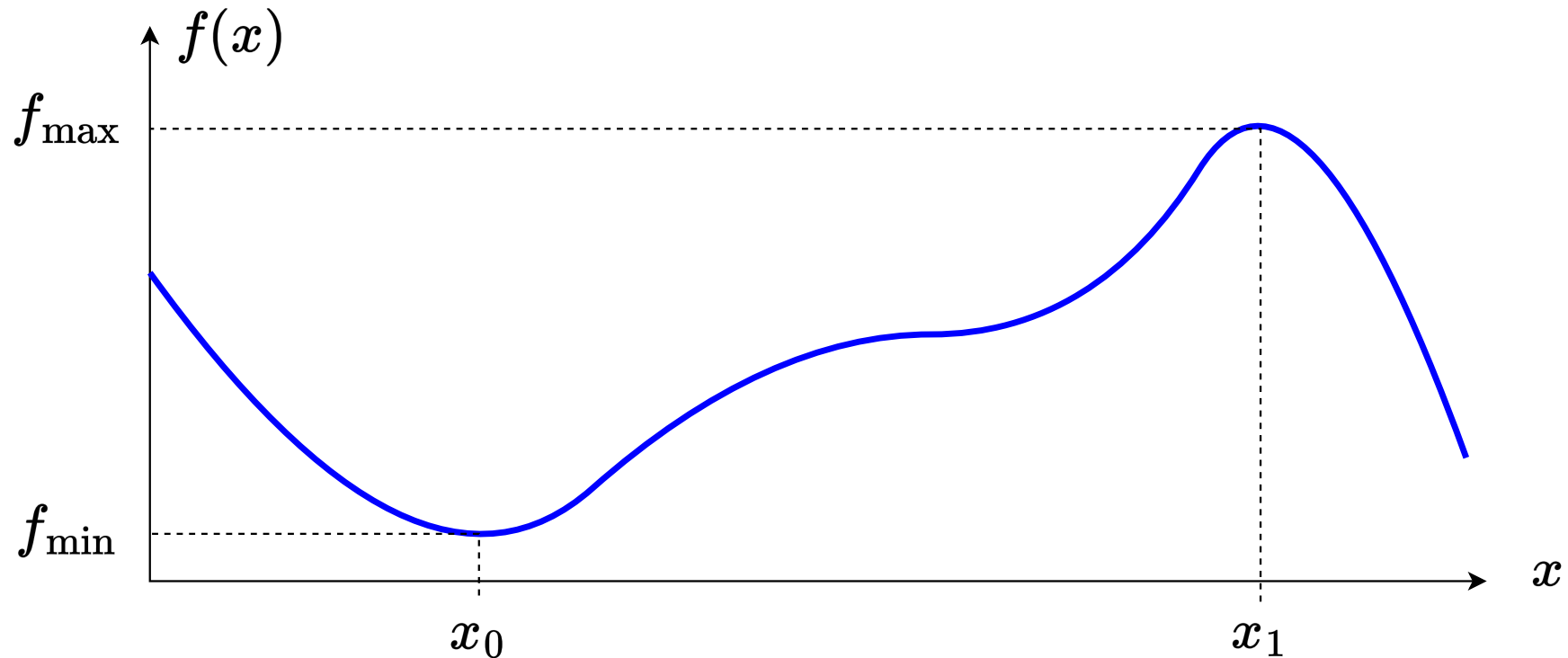
$$\begin{aligned} f'(x) &= \lim_{t \rightarrow x} \frac{(A_2t^2 + A_1t + B) - (A_2x^2 + A_1x + B)}{t - x} \\ &= \lim_{t \rightarrow x} \frac{A_2(t^2 - x^2) + A_1(t - x)}{t - x} \\ &= \lim_{t \rightarrow x} A_1 + A_2 \frac{(t + x)(t - x)}{t - x} \\ &= A_1 + 2A_2 \cdot x \end{aligned}$$

- **observation:** $f'(x)$ is a function in x too!
- **implication:** the slope of a function can change with respect to x !
- **think:** suppose a function is bounded, what is the slope of a functional extremum for quad.?

Example: $f'(x)$ is the slope of $f(x)$, but also varies with x

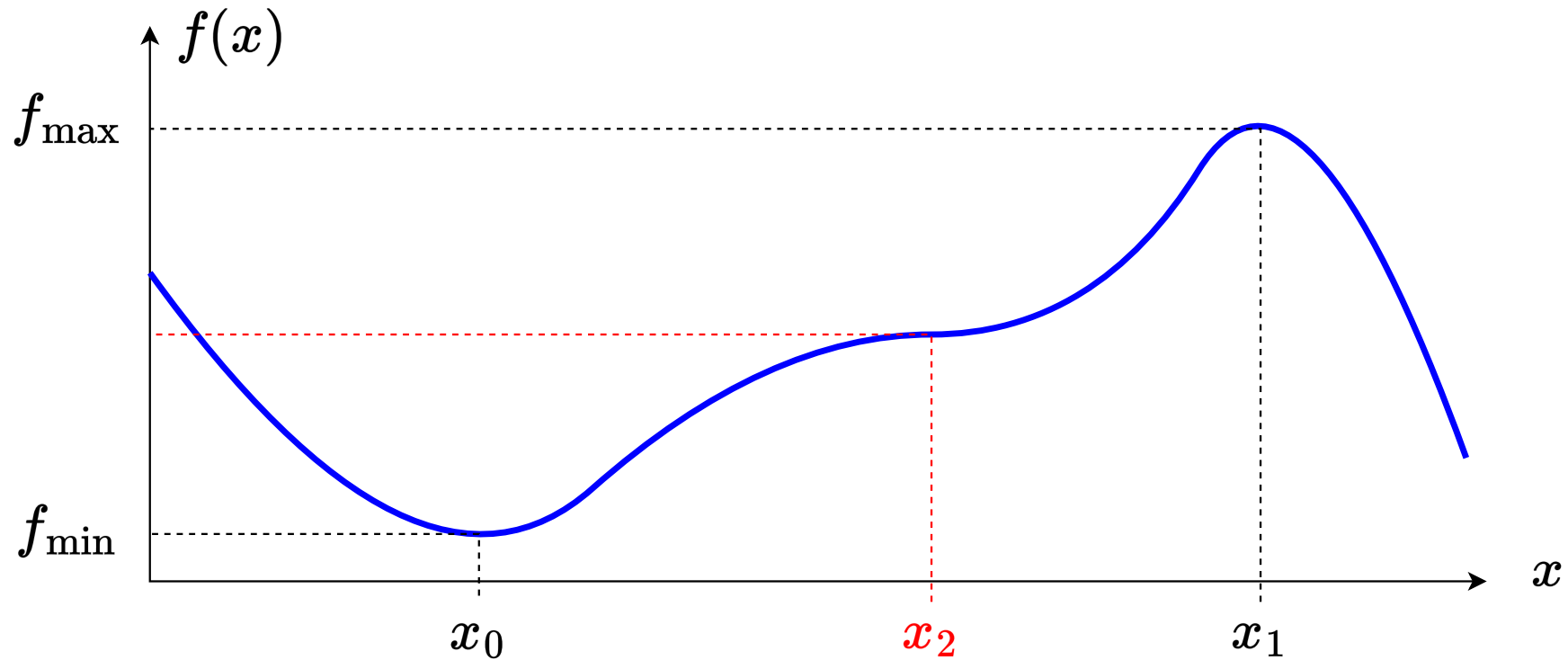


Functional Extrema



- observe that $f'(x_0) = f'(x_1) = 0$
- implication: if we search for min/max value of $f(x)$, we can look for points $x^* \in \mathbb{R}$ that has $f'(x^*) = 0$
- but we don't know whether it is min or max if no further assumptions on $f(x)$ is made

Functional Extrema



- formally: if $f(x)$ has a min/max point at x^* , then $f'(x^*) = 0$
- but the converse is not true; e.g. $f'(x_2) = 0$ does not imply $f(x_2)$ is a min/max point

Preliminary: Matrix Algebra

Matrix, Vector and Scalar

- $x \in \mathbb{R}$ is called a **scalar**; it contains one value only

- $\mathbf{x} = (x_1, x_2, \dots, x_n) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ is called a **vector**, which contains multiple scalars

- $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_m^\top \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{bmatrix} \in \mathbb{R}^{m \times n}$ is a **matrix**

– the transposition $(\cdot)^\top$ is to horizontalize a vector

– see: \mathbf{X} is a **collection of vectors** $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$

- observe: **vector** is a special case of **matrix**

Vector as Data

$$\boldsymbol{x} = (x_1, x_2, \dots, x_n) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- numbers stored inside a vector \boldsymbol{x} may contain different physical meaning
- physicist's point of view: two dimensional displacement
- programmer's point of view: just an ordered linear list
- computer vision point of view: color coding such as RGB
- **word count**: x_i is the number of times word i appears in a document

Example: Word Count Vector

Word count vectors are used in computer based document analysis. Each entry of the word count vector is the number of times the associated dictionary word appears in the document.

- **word count:** x_i is the number of times word i appears in a document
- a small dictionary leads to a word count vector to represent the above passage:

$$\begin{array}{l} \text{word} \\ \text{in} \\ \text{number} \\ \text{horse} \\ \text{the} \\ \text{document} \end{array} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \\ 4 \\ 2 \end{bmatrix} = x$$

- btw this is one of the foundations of how conversational A.I.'s got trained!

Basic Operators for Matrices

- **Addition/subtraction:** Given $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$,

$$\begin{aligned}\mathbf{A} + \mathbf{B} &= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}\end{aligned}$$

whereas $\mathbf{A} - \mathbf{B} = \mathbf{A} + (-\mathbf{B})$ is defined in a similar manner.

- **Scalar multiplication:** Given a scalar term $c \in \mathbb{R}$,

$$c\mathbf{A} = c \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \begin{bmatrix} ca_{11} & ca_{12} & \dots & ca_{1n} \\ ca_{21} & ca_{22} & \dots & ca_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ ca_{m1} & ca_{m2} & \dots & ca_{mn} \end{bmatrix}$$

Basic Operators for Matrices

- **Inverse matrix:** For $A \in \mathbb{R}^{n \times n}$, if there exists $B \in \mathbb{R}^{n \times n}$ such that

$$AB = BA = I$$

then $B = A^{-1}$ is said to be the inverse matrix of A . Some remarks:

- only square matrices are eligible to have an inverse, but not all square matrices have one; e.g. one can show that $\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$ has no inverse
- computationally we assume programmes (such as MATLAB) is capable of computing an inverse of a given matrix
- for non-square data matrices one may consider using the Penrose-Moore inverse (details skipped)

Basic Operators for Matrices

- **Matrix multiplication:** The product between 2 matrices is defined more strictly —it may not always exist. Let $\mathbf{C} \in \mathbb{R}^{n \times m}$, then the product

$$\mathbf{AC} = m \text{ rows} \left\{ \underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}}_{n \text{ columns}} \underbrace{\begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{bmatrix}}_{m \text{ columns}} \right\} n \text{ rows}$$

and the (i, j) th element of \mathbf{AC} is defined as

$$[\mathbf{AC}]_{ij} = \sum_{k=1}^n a_{ik} c_{kj}$$

Note that even if $\mathbf{A}, \mathbf{C} \in \mathbb{R}^{n \times n}$, $\mathbf{AC} \neq \mathbf{CA}$ in general.

- Alternatively, I found this statement easier to interpret: $[\mathbf{AC}]_{ij} = \mathbf{a}_i^\top \mathbf{c}_j$.

Numerical Examples

Suppose there are three matrices

$$\mathbf{A} = \begin{bmatrix} -8 & -4 & -1 \\ 2 & -6 & 6 \\ -3 & 5 & -1 \\ 0 & 4 & 3 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 7 & -2 \\ 1 & -5 \\ 7 & 2 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} -2 & -2 & 2 \\ -1 & -1 & -2 \\ 1 & 1 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

It is straight forward to see that

$$\mathbf{A} - \mathbf{C} = \begin{bmatrix} -6 & -2 & -3 \\ 3 & -5 & 8 \\ -4 & 4 & 0 \\ -1 & 4 & 3 \end{bmatrix}, \mathbf{A} + \mathbf{C} = \begin{bmatrix} -10 & -6 & 1 \\ 1 & -7 & 4 \\ -2 & 6 & -2 \\ 1 & 4 & 3 \end{bmatrix}, 0.3\mathbf{A} + 0.6\mathbf{C} = \begin{bmatrix} -3.6 & -2.4 & 0.9 \\ 0 & -2.4 & 0.6 \\ -0.3 & 2.1 & -0.9 \\ 0.6 & 1.2 & 0.9 \end{bmatrix}$$

Also, it is clear that the sum $\mathbf{A} + \mathbf{B}$ or $\mathbf{C} + \mathbf{B}$ do not exist due to dimensional mismatch. The same goes for all the difference.

The matrix products

$$\mathbf{AB} = \begin{bmatrix} -67 & 34 \\ 50 & 38 \\ -23 & -21 \\ 25 & -14 \end{bmatrix}, \mathbf{CB} = \begin{bmatrix} -2 & 18 \\ -22 & 3 \\ 1 & -9 \\ 7 & -2 \end{bmatrix}$$

follows from the definition of matrix multiplication. Other products \mathbf{AC} , \mathbf{CA} , \mathbf{BC} and \mathbf{BA} do not exist due to, again, mismatch in dimensionality.

Application: Solving For System of Linear Equations

Consider the following system of linear equality:

$$\begin{aligned}8x_1 + 3x_2 + 0x_3 &= 30 \\0x_1 - 3x_2 + 6x_3 &= 0 \\-2x_1 + 2x_2 + 2x_3 &= 10\end{aligned}$$

This can be rewritten into a matrix-vector product form:

$$\underbrace{\begin{bmatrix} 8 & 3 & 0 \\ 0 & -3 & 6 \\ -2 & 2 & 2 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} 30 \\ 0 \\ 10 \end{bmatrix}}_{\mathbf{b}}$$

where \mathbf{A} , \mathbf{b} are known data; \mathbf{x} is a vector storing all the variables; this can be solved easily by matrix inverse (see my MATLAB demo):

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = (2, 14/3, 7/3)$$

Mathematical Optimization

The Remainder (and basically the spirit) of This Talk

- will be focusing on mathematical optimization
- motivate the usage of optimization model to handle problems arising in A.I. (pattern classification, signal and image source separation)
- go through a demonstration of how to use software tools (MATLAB + CVX) in solving opt.

Mathematical Optimization

- **General Formulation of Opt.:**

$$\begin{array}{ll}\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \dots, M\end{array}$$

- \boldsymbol{x} is the optimization variable (vector)
 - $f(\boldsymbol{x})$ is the objective function, and is scalar-valued
 - $g_i(\boldsymbol{x})$ are the constraint functions, also scalar-valued
- **goal:** find a vector \boldsymbol{x}^* that minimizes f whilst satisfying all the constraints g_i 's
 - **observation:** variable, objective and constraints are the key elements in optimization

High Level Examples

- **investment strategy:** let x be the portion of money you spend on buying n different stock products, we can formulate:

$$\begin{array}{ll}\underset{x \in \mathbb{R}^n}{\text{minimize}} & \text{investment risk} \\ \text{subject to} & \text{total investment} \leq \text{total budget} \\ & \text{investment on each product} \leq \text{max spending on each product} \\ & \text{expected return} \geq \text{total budget}\end{array}$$

the core issue rests in how to quantify risk and expected return (need finance knowledge)

- **desktop assembling:** let x be the budget you spend on buying n different parts of a desktop computer, and you have some specifications on the parts...

$$\begin{array}{ll}\underset{x \in \mathbb{R}^n}{\text{minimize}} & \text{total budget} = x_1 + x_2 + \cdots + x_n \\ \text{subject to} & \text{CPU} \geq \text{i5-12th gen, RAM} \geq 16\text{GB, motherboard} \geq \dots \text{ etc.} \\ & \text{compatibility constraints}\end{array}$$

the constraints on parts can be translated into $x_i \geq C_i$ where C_i = price of the least acceptable component; whereas the compatibility constraints require hardware knowledge

Solution to Optimization Problems

- general optimization problems are uneasy to solve
- exceptions: unconstrained least square, linear programmes, **convex optimization**
 - least square: $\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|$ is just solving for sys. of lin. eqn. (seen earlier)
 - linear programme: given data $\mathbf{A}, \mathbf{b}, \mathbf{c}_i, d_i$, find \mathbf{x} according to:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{Ax} - \mathbf{b}\| \\ \text{s.t.} \quad & \mathbf{c}_i^\top \mathbf{x} \leq d_i, \quad i = 1, \dots, M \end{aligned}$$

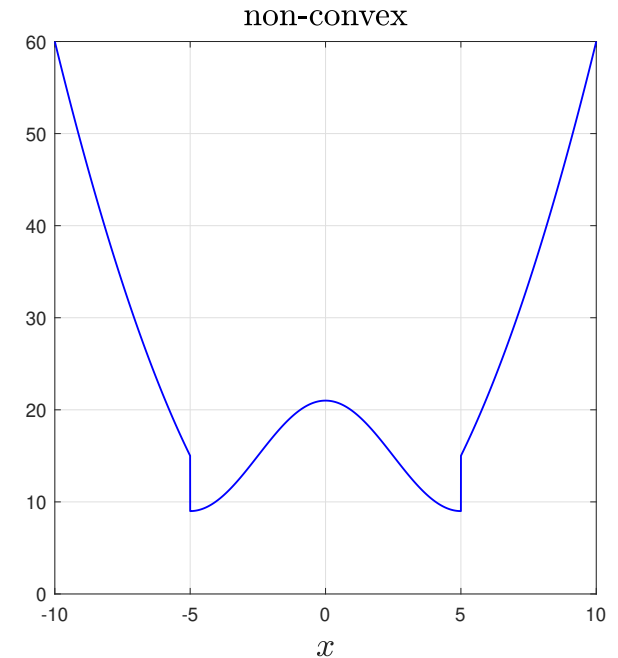
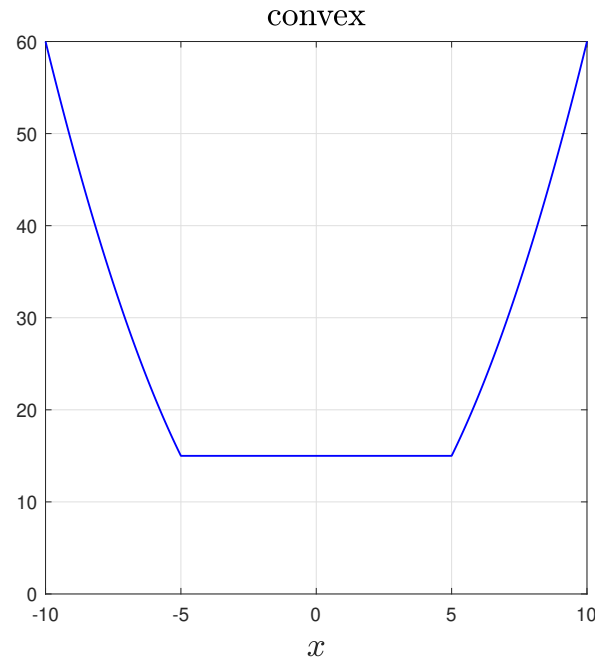
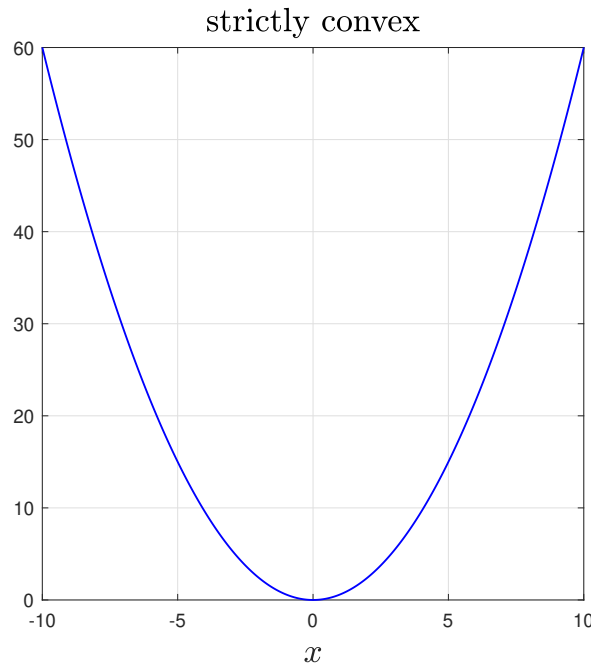
is a well studied problem; but sometimes hard to recognise the problem as LP

- **convex programme**: given functions f and g_i 's are convex in \mathbf{x} , find:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq d_i, \quad i = 1, \dots, M \end{aligned}$$

is a generalization of both LS and LP above; also well studied and almost considered as a mature tech.

Convexity



- formally speaking, a function f is said to be convex if

$$f(\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1 - \alpha) f(\mathbf{x}_2)$$

for any vector $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ & $0 \leq \alpha \leq 1$

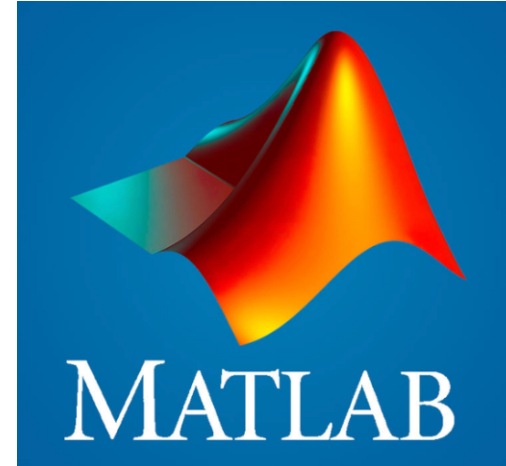
- key feature: convex function f has **unique minimum value**
- but there can be multiple minimizers achieving the same min. value

Merits of Convex Optimization

$$\begin{array}{ll}\min_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq d_i, \quad i = 1, \dots, M\end{array}$$

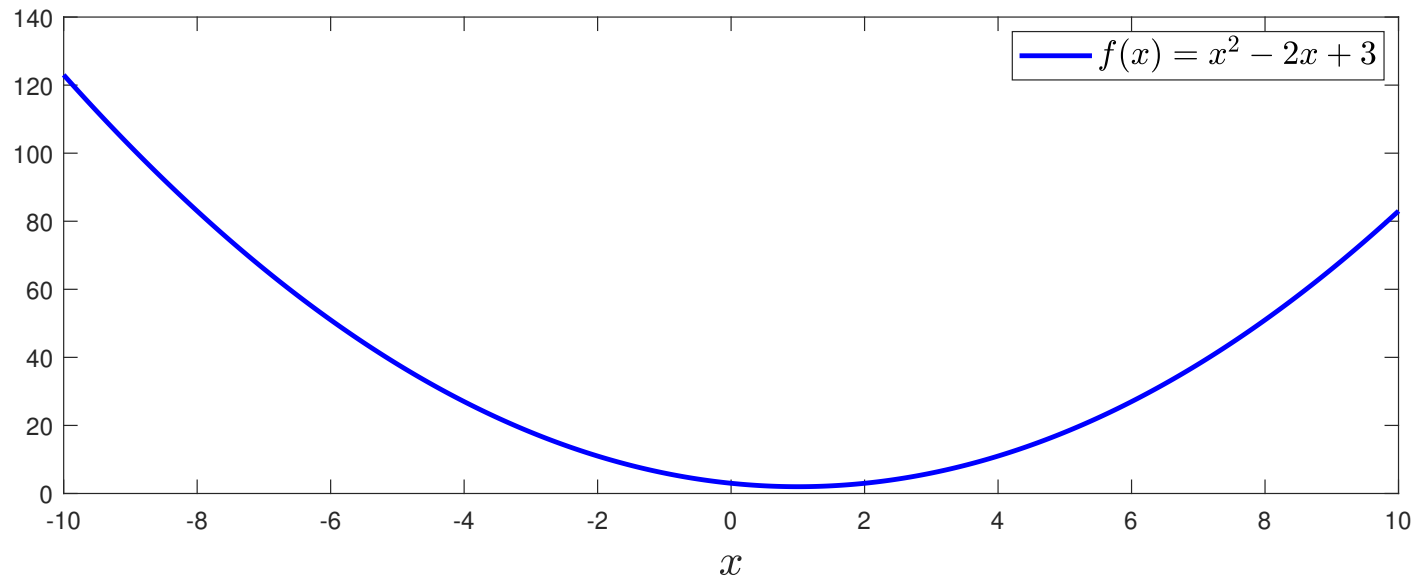
- once solved, it provides a **globally optimal solution**
- (recall that min. value locates at where $f'(\boldsymbol{x}) = 0$!)
- luckily, there are many efficient solvers for convex programmes available
- realistic problems can often be formulated into convex opt.!

CVX: A General Purpose Convex Optimization Solver



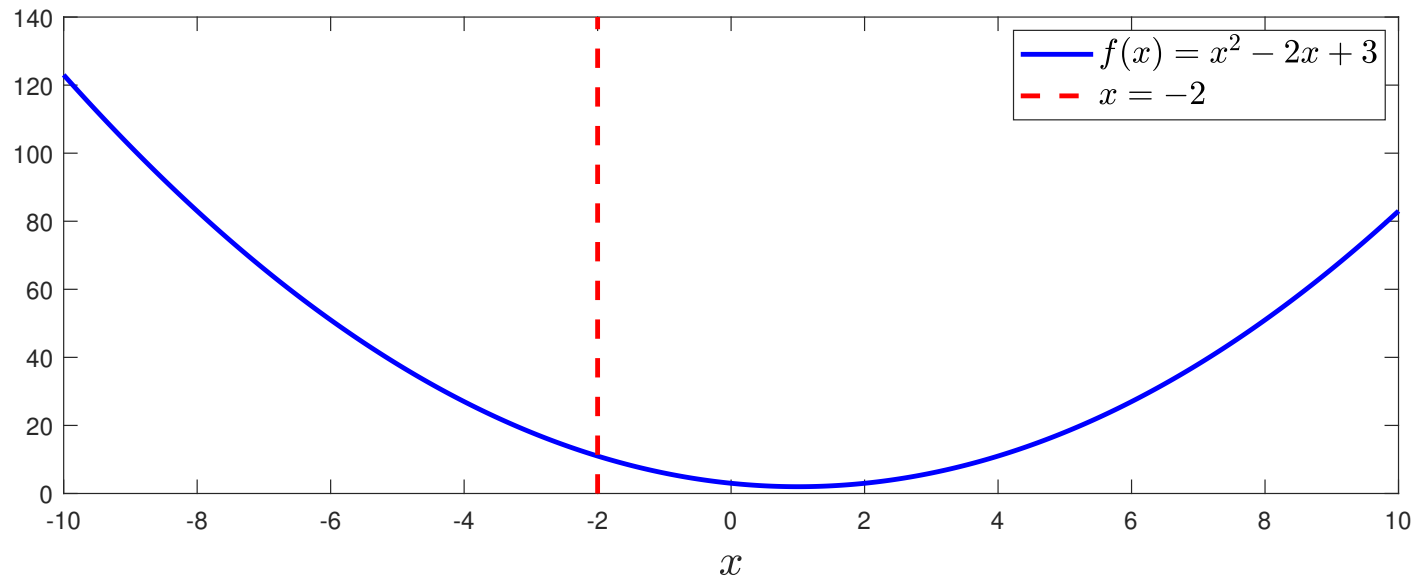
- CVX is a MATLAB-based modeling system for convex optimization problems
- assuming you have MATLAB licences, you may install the cvx package freely as instructed on its official website: <http://cvxr.com/cvx/>
- easy to use and can be applied to many optimization problems; prototyping without requiring a lot of coding/math skills
- allows us to directly write an optimization problem and solve it immediately

Numerical Example



- suppose we want to minimize $f(x) = x^2 - 2x + 3$

Numerical Example



- suppose we want to minimize $f(x) = x^2 - 2x + 3$
- we constraint x such that $3x - 1 \leq -7$, or equivalently, $g(x) = x \leq -2$
- the min. value should be to the LHS of the red dash line
- by inspection: $x^* = -2$ and $f(x^*) = 11$

Numerical Example

- let us formulate the above example into standard opt. form:

$$\begin{aligned} \min_x \quad & x^2 - 2x + 3 \\ \text{s.t.} \quad & 3x - 1 \leq -7 \end{aligned}$$

which can be easily solved via CVX and MATLAB:

```
cvx_begin
    variable x(1)
    minimize x^2 - 2*x + 3
    subject to
        3*x - 1 <= -7
cvx_end
```

- on-screen demonstration will show you the optimal value is $f_{\min} = 11$, and is achieved when $x = -2$, which agrees with our discussion above
- imaginably, additional constraints are easily insert-able below 'subject to'

Applications in Artificial Intelligence

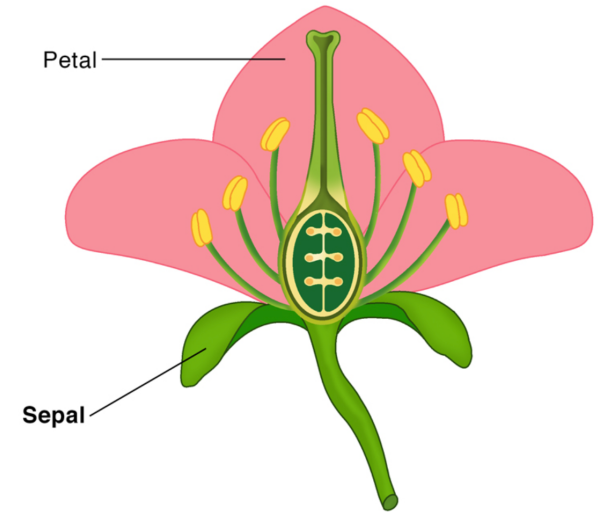
Application 1: Pattern Classification



Rose



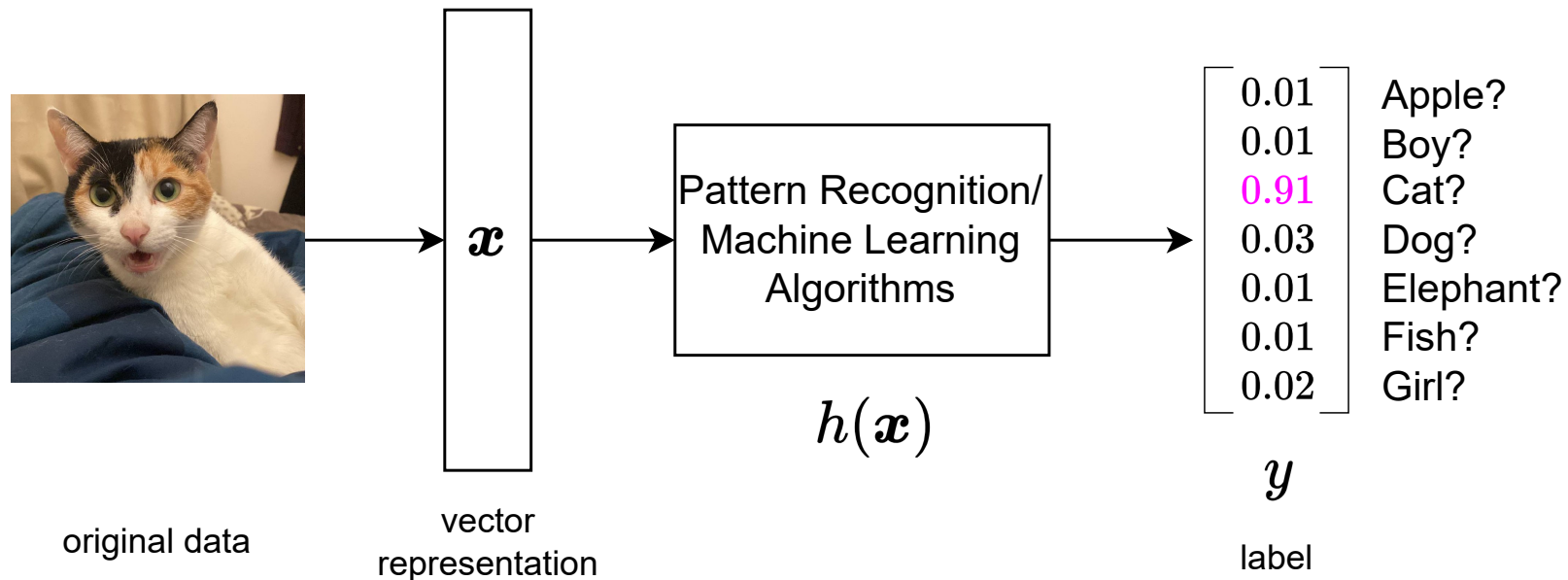
Sunflower



Flower's Structure

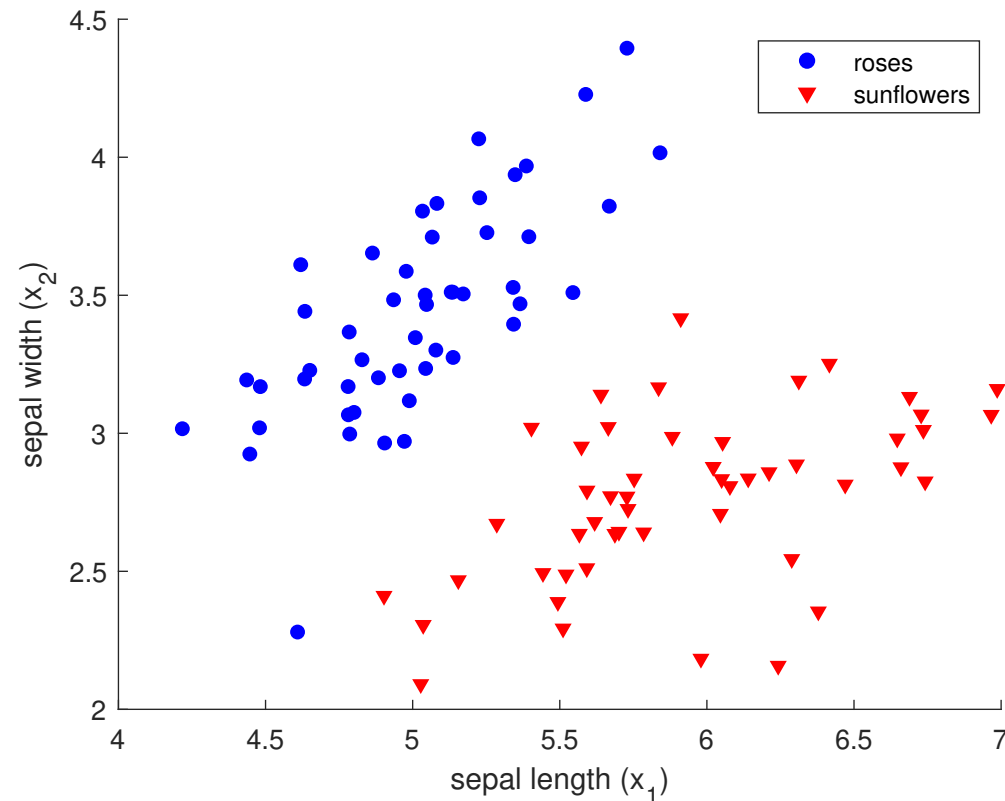
- suppose you are given a large amount of flowers, some of which are roses and the rest are sunflowers; and you have measured and recorded the sepal's length and width respectively
- from the data collected, we **learnt** how does the sepal of rose/sunflower look like
- can we **teach** a machine to classify a new subject base on these data?

Pattern Classification



- the term **pattern recognition/classification** and **machine learning** are generally used **interchangeably**; although “machine learning” is actually more popular in this era of A.I.
- the basic idea is to **train/learn** a **hypothesis function** $h(x)$ that gives an estimation on the **likelihood** of the data belong to a certain class
- the function $h(x)$ achieves better performance if we could give a better design process that involves **domain knowledge**

Pattern Classification



- the data we collected can be symbolised by a vector $\mathbf{x} = (x_1, x_2)$
- we can also assign a numeric value to the class y of rose ($y = +1$)/ sunflowers ($y = -1$)
- we have a total of 100 data above and we call them $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{100}$
- **goal:** find a good function $h(\mathbf{x}^{(i)}) \approx y^{(i)}$

Linear Classifier

- consider a simple yet powerful decision model: **linear classifier**

$$h(\mathbf{x}; \mathbf{w}, b) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$$

where $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are **model parameters** to be **learnt**

- we can translate the problem as finding (\mathbf{w}, b) such that

$$\text{sgn}(\mathbf{w}^\top \mathbf{x}^{(i)} + b) = y^{(i)} \quad \text{for all } i = 1, 2, \dots, 100.$$

- steering at the above equality allows us to cast the optimization problem as:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & 0 \\ \text{s.t.} \quad & y^{(i)} \cdot (\mathbf{w}^\top \mathbf{x}^{(i)} + b) > 0, \quad i = 1, \dots, 100 \end{aligned}$$

(I will show the derivation in class if we have time)

Linear Classifier by Convex Optimization

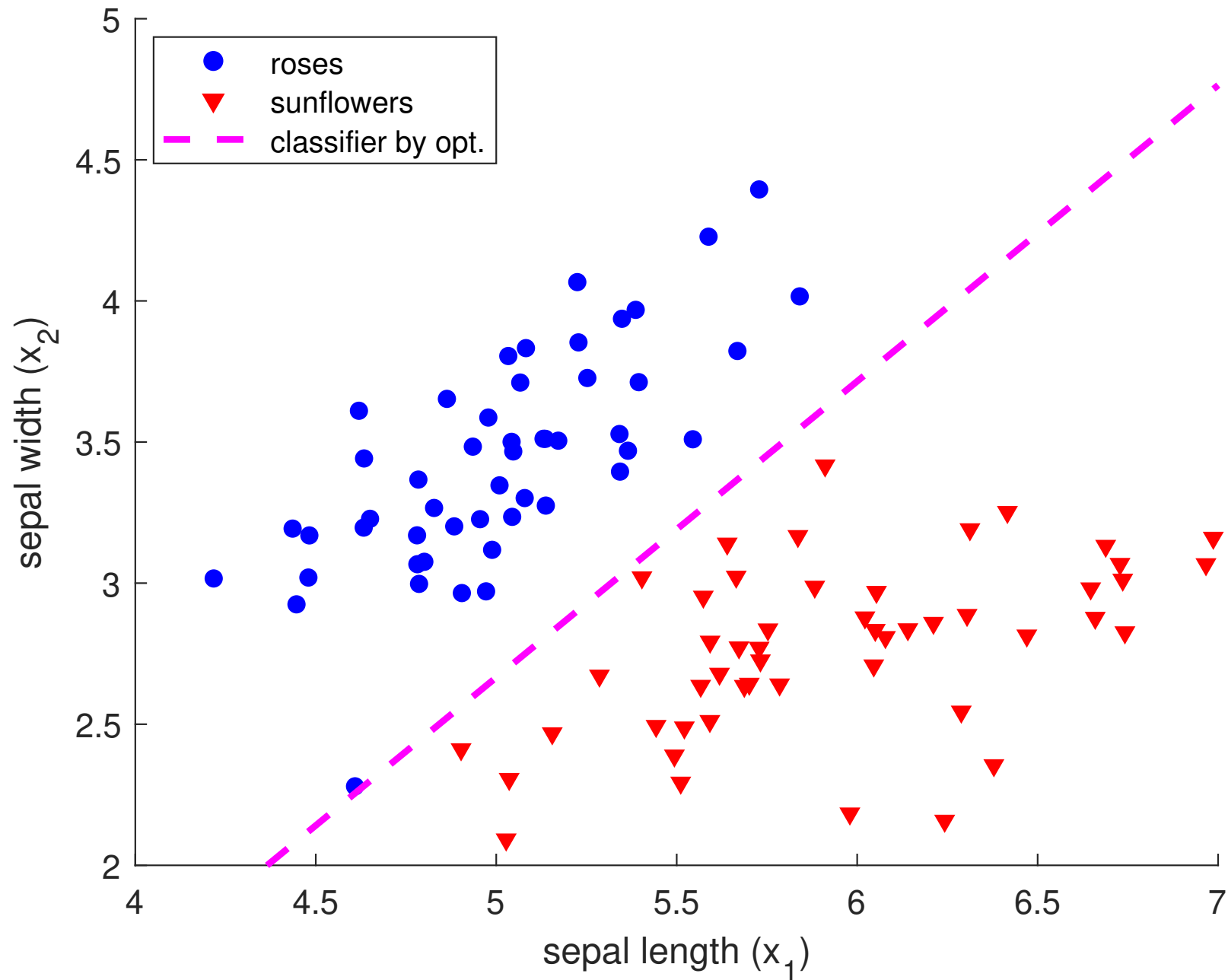
$$\begin{aligned} \min_{\mathbf{w}, b} \quad & 0 \\ \text{s.t.} \quad & y^{(i)} \cdot (\mathbf{w}^\top \mathbf{x}^{(i)} + b) > 0, \quad i = 1, \dots, 100 \end{aligned}$$

- it is straightforward to implement the above opt. in cvx:

```
cvx_begin
    variables w(2) b(1)
    minimize 0
    subject to
        for i = 1:100
            y(i) * (X(i, :)*w + b) > 0;
        end
cvx_end
```

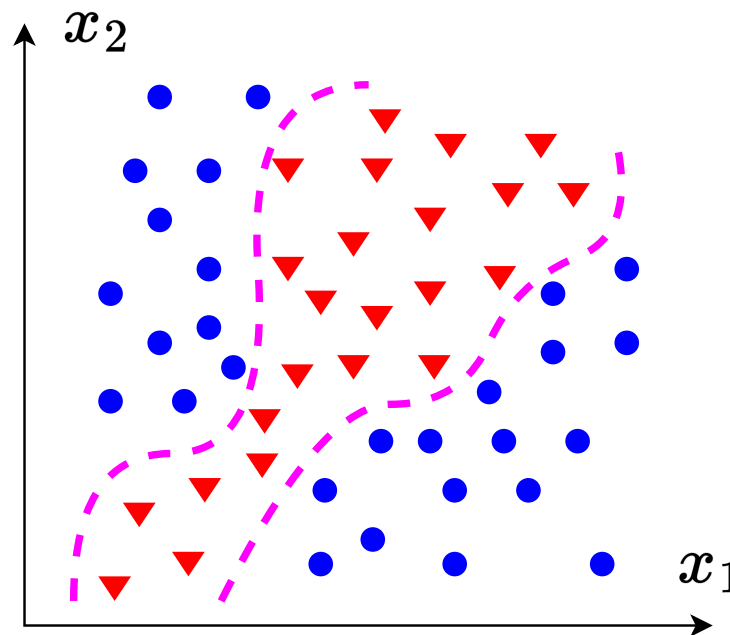
- let us visualize what is given by cvx on the next slide

Linear Classifier by Convex Optimization



Extension: Non-linear Classifier

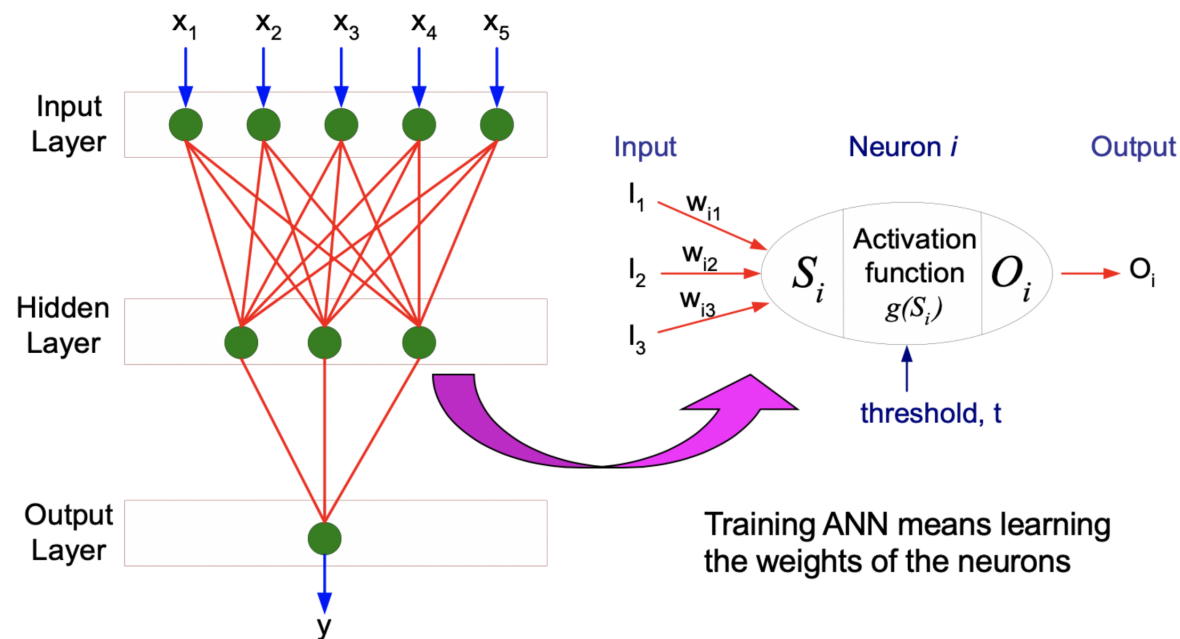
- it is clear that machines can use the pink line to serve as a decision next time in determining whether a subject is rose or sunflower
- the same applies to many other applications: handwritten digit recognition, image classification and even credit card transaction fraud detection
- however, most tasks in practice are non-linearly separable...



implying that a “straight line” cannot classify the data accurately

Artificial Neural Network

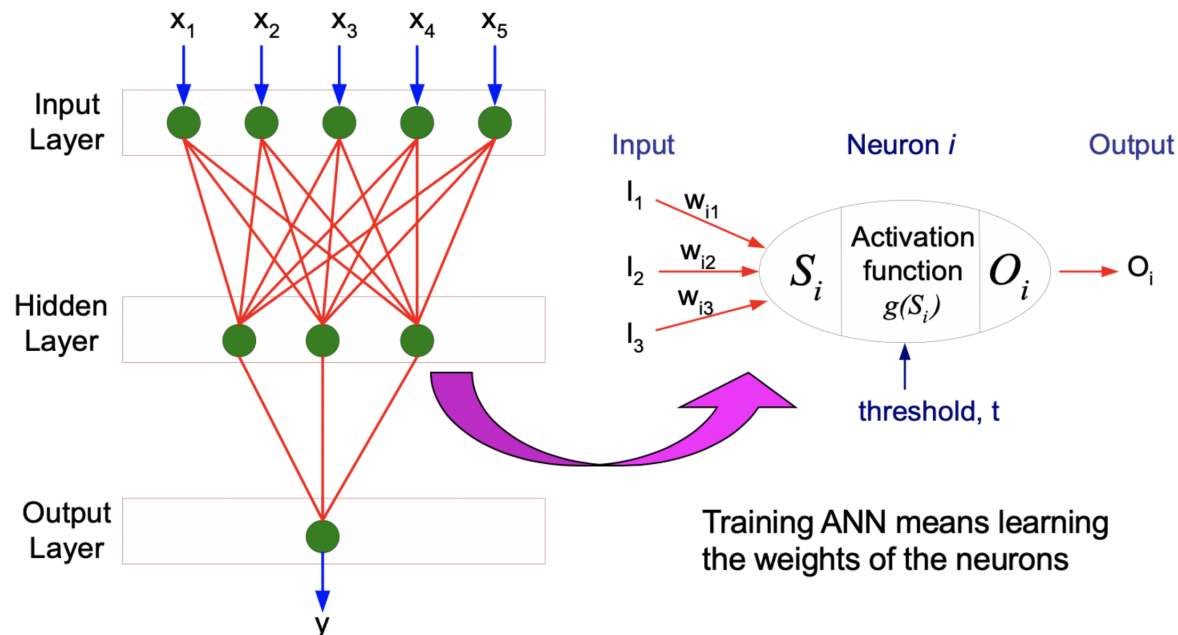
- recall the decision model $h(x) = \text{sgn}(\mathbf{w}^\top \mathbf{x} + b)$ is linear inside the signum function
- the linear function does not exist for the dataset in the previous slide; we thus aim at using a nonlinear function, e.g., an **artificial neural network (ANN)**



which considers a **non-linear** decision model

$$h(\mathbf{x}) = \text{sgn}[\mathbf{w}_2^\top \text{sgn}(\mathbf{w}_1^\top \mathbf{x} + b_1) + b_2]$$

Artificial Neural Network



- we can **train** the ANN by the optimization formulation

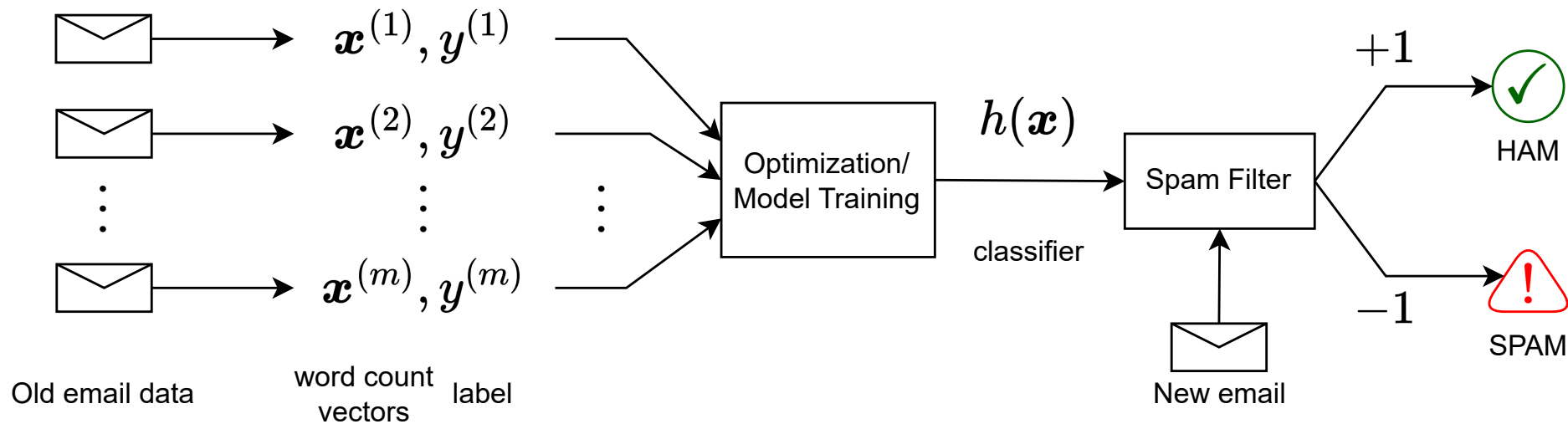
$$\min_{\mathbf{w}_1, \mathbf{b}_1, \mathbf{w}_2, \mathbf{b}_2} \sum_{i=1}^m |y^{(i)} - h(\mathbf{x}^{(i)})|^2$$

$$\text{s.t. } h(\mathbf{x}^{(i)}) = \text{sgn}[\mathbf{w}_2^T \text{sgn}(\mathbf{w}_1^T \mathbf{x}^{(i)} + \mathbf{b}_1) + \mathbf{b}_2], \quad i = 1, \dots, m$$

which means a minimization over the **sum-of-squared error**

- it is however a **non-convex optimization** problem, which is much harder than the linear case discussed before

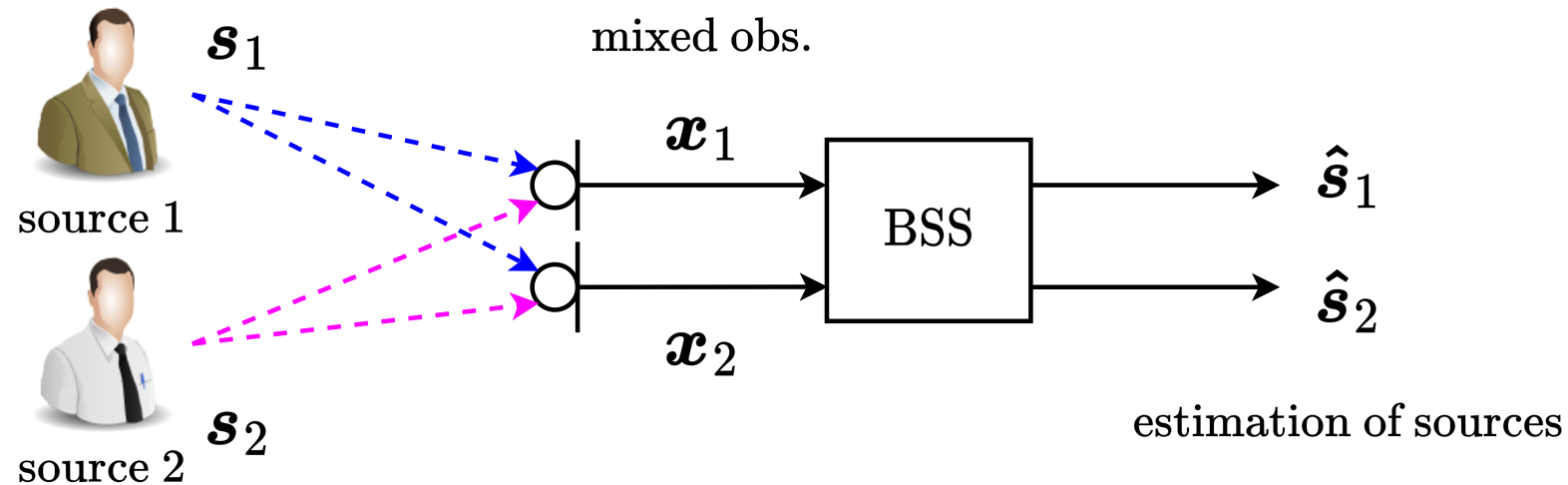
Real-Life Application: Spam Mail Filtering



- a daily-life example of classification is spam mail detector
- existing emails are converted into word count vectors \mathbf{x} , and Google will label them as either spam ($y = -1$), or ham ($y = +1$)
- the classifier $h(\mathbf{x})$ can then be trained according to our previous discussions

Application 2: Blind Source Separation

- in many data retrieval tasks sources are mixed upon arrival of sensor's observation
- blind source separation (BSS) is a method to unmix those data
- example: the classical cocktail party problem



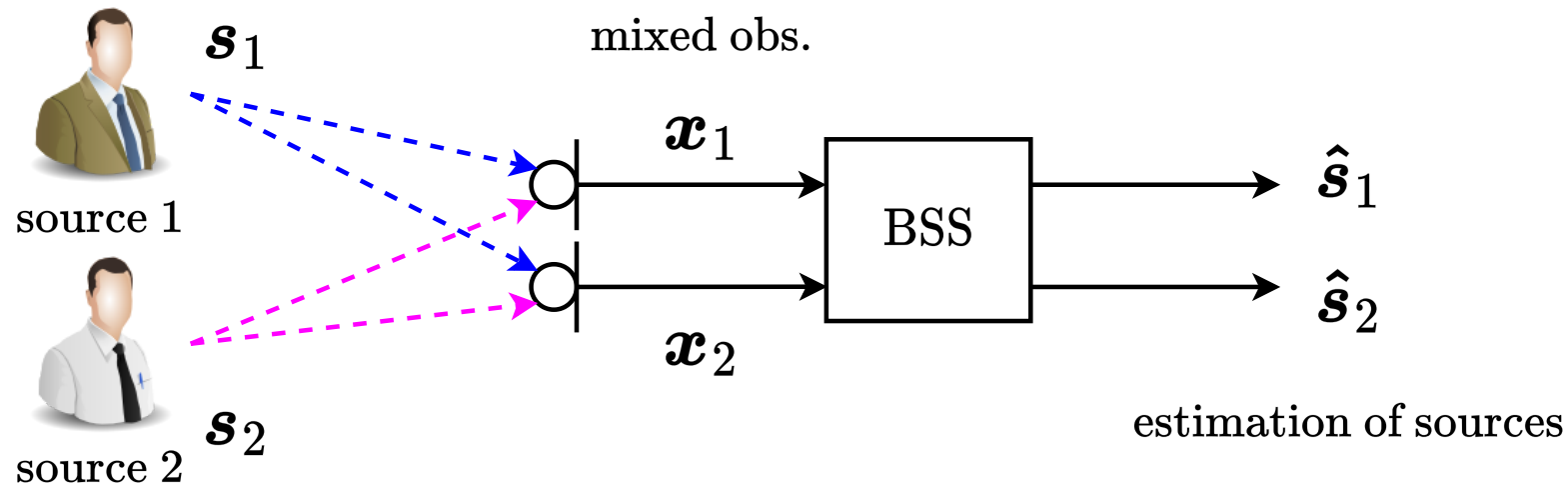
in which we would like to take back an estimation of the sources given mixed observations

$$x_1 = a_{1,1}s_1 + a_{2,1}s_2$$

$$x_2 = a_{1,2}s_1 + a_{2,2}s_2$$

without knowing the mixing coefficients $a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}$

Blind Source Separation: Problem Set-up



- the observation signal model can be rewritten compactly as

$$\underbrace{\begin{bmatrix} x_1^\top \\ x_2^\top \end{bmatrix}}_X = \underbrace{\begin{bmatrix} a_{1,1} & a_{2,1} \\ a_{1,2} & a_{2,2} \end{bmatrix}}_A \underbrace{\begin{bmatrix} s_1^\top \\ s_2^\top \end{bmatrix}}_S$$

where $\{s_i, x_i\}_{i=1,2}$ are length T vectors (imagine a fragment of audio)

- question:** given X , can we take back **both** A and S ? (seems pretty handy...)
- strategy:** what about we solve for one unknown at a time?

One At A Time...?

$$\boxed{X = AS}$$

- suppose we have some **initial guesses** on the mixture matrix A , e.g.

$$A_{\text{est}} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$$

which means the signal sources are uniformly mixed

- we can then formulate an optimization problem

$$S_{\text{est}} \leftarrow \min_S \|X - A_{\text{est}} S\|$$

which will give **the best estimate** of S when $A = A_{\text{est}}$, i.e. **the guess is accurate**

- we then use S_{est} to optimize the estimation of A

$$A_{\text{est}} \leftarrow \min_A \|X - AS_{\text{est}}\|$$

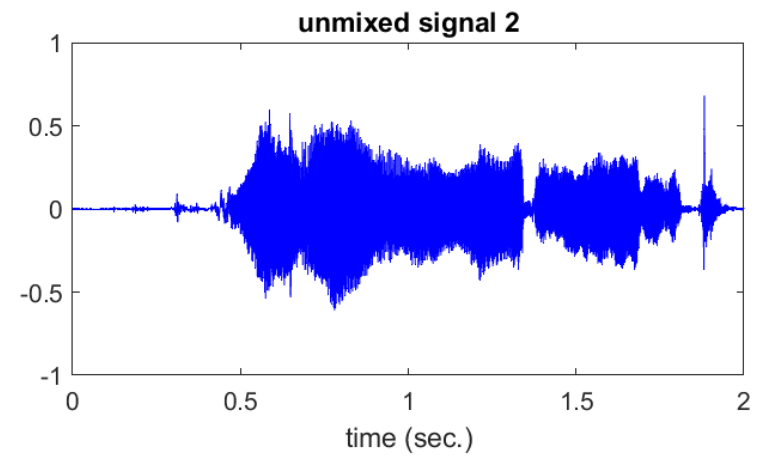
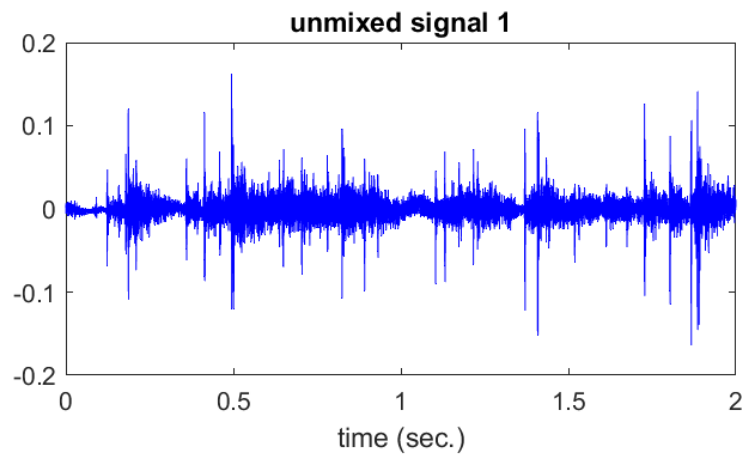
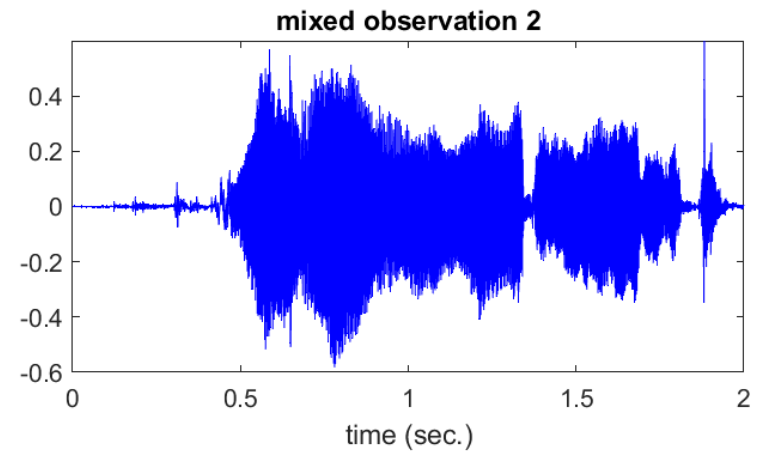
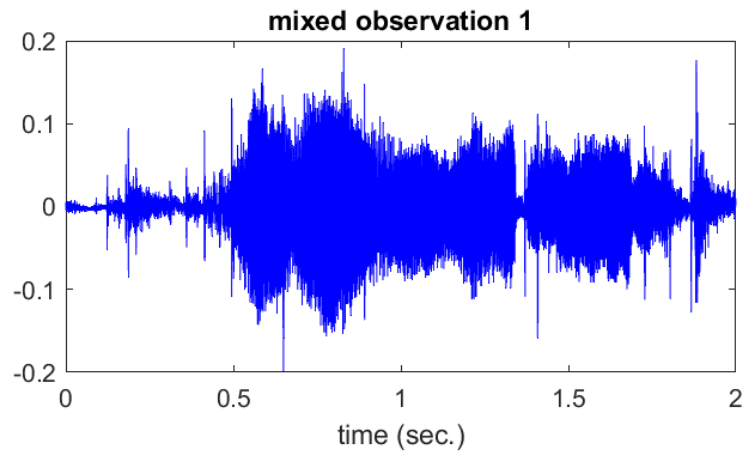
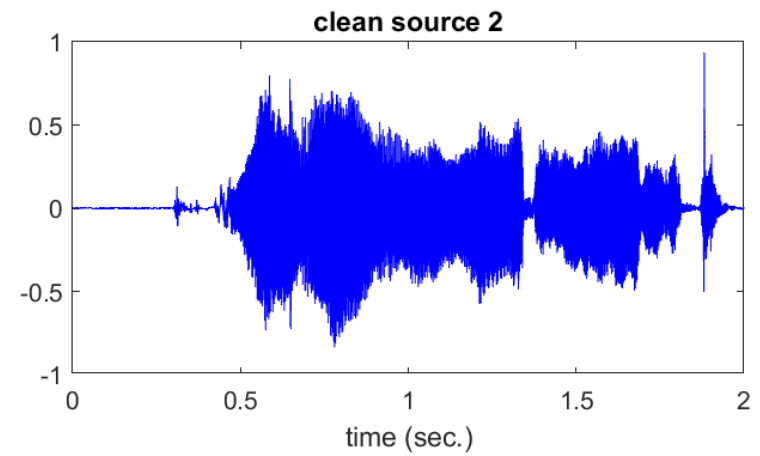
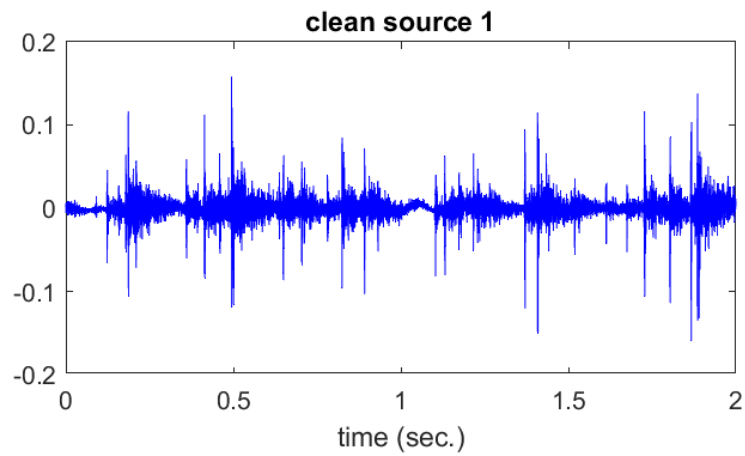
which will give an **update** for the poor guess above

The One-At-A-Time Method

Algorithm.

1. Given **input** observations X . Initialize a guess on the mixing matrix A_{est} .
2. **repeat, alternatingly:**
 - $S_{\text{est}} \leftarrow \min_S \|X - A_{\text{est}} S\|$
 - $A_{\text{est}} \leftarrow \min_A \|X - A S_{\text{est}}\|$**until** some convergence criterion is met, or you have waited too long
3. **return:** $S_{\text{est}}, A_{\text{est}}$

- we have successfully built an algorithm using basic optimization idea
- we should also mention that both update steps are convex, i.e. solvable by cvx
- let's test the methodology by the cocktail party problem



BSS with Additional Constraints

- the same BSS method can be further modified to suit a different application
- consider mixture of multiple images this time, that is, in our model,

$$\mathbf{X} = \mathbf{A}\mathbf{S}$$

both \mathbf{X} and \mathbf{S} are visual or image data

- we should stress the fact that image data contains **non-negative values only**; in other words, the model should satisfy

$$\mathbf{X} \geq 0, \quad \mathbf{I} \geq \mathbf{A} \geq 0, \quad \mathbf{S} \geq 0$$

- the problem has numerous more applications, e.g. medical imaging, hyperspectral un-mixing and video de-ghosting

Example: X-ray Image Unmixing

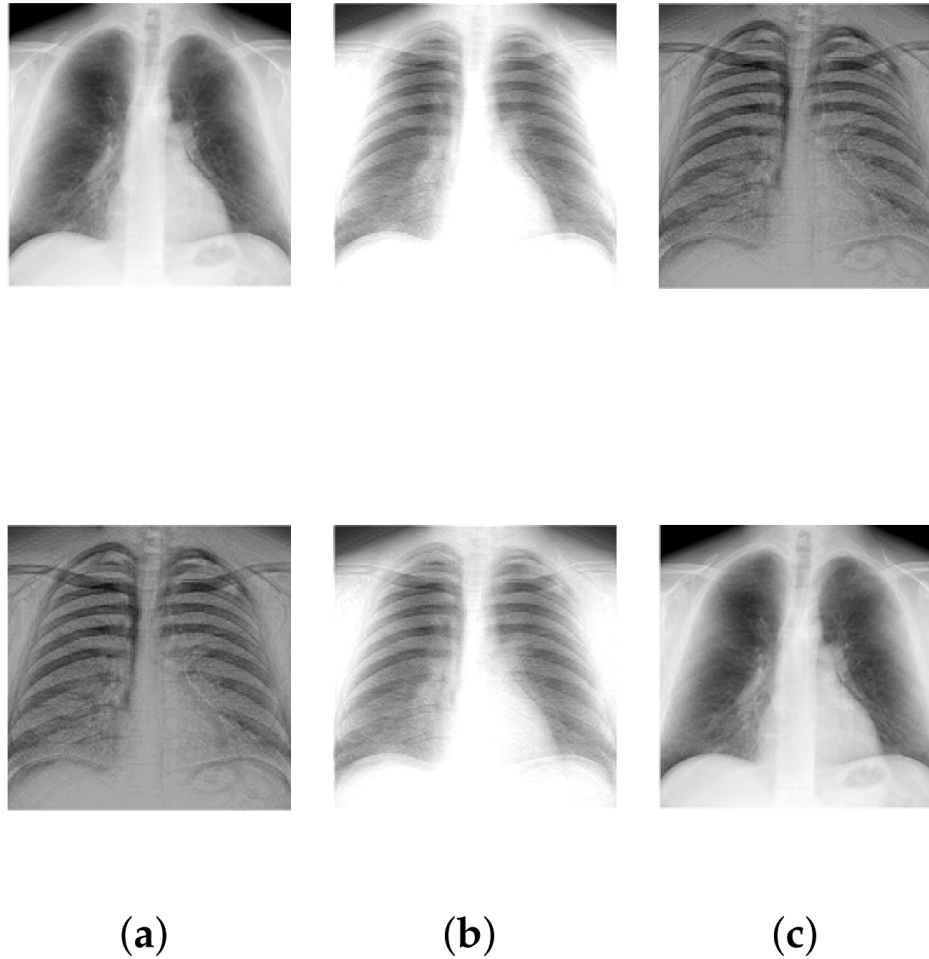
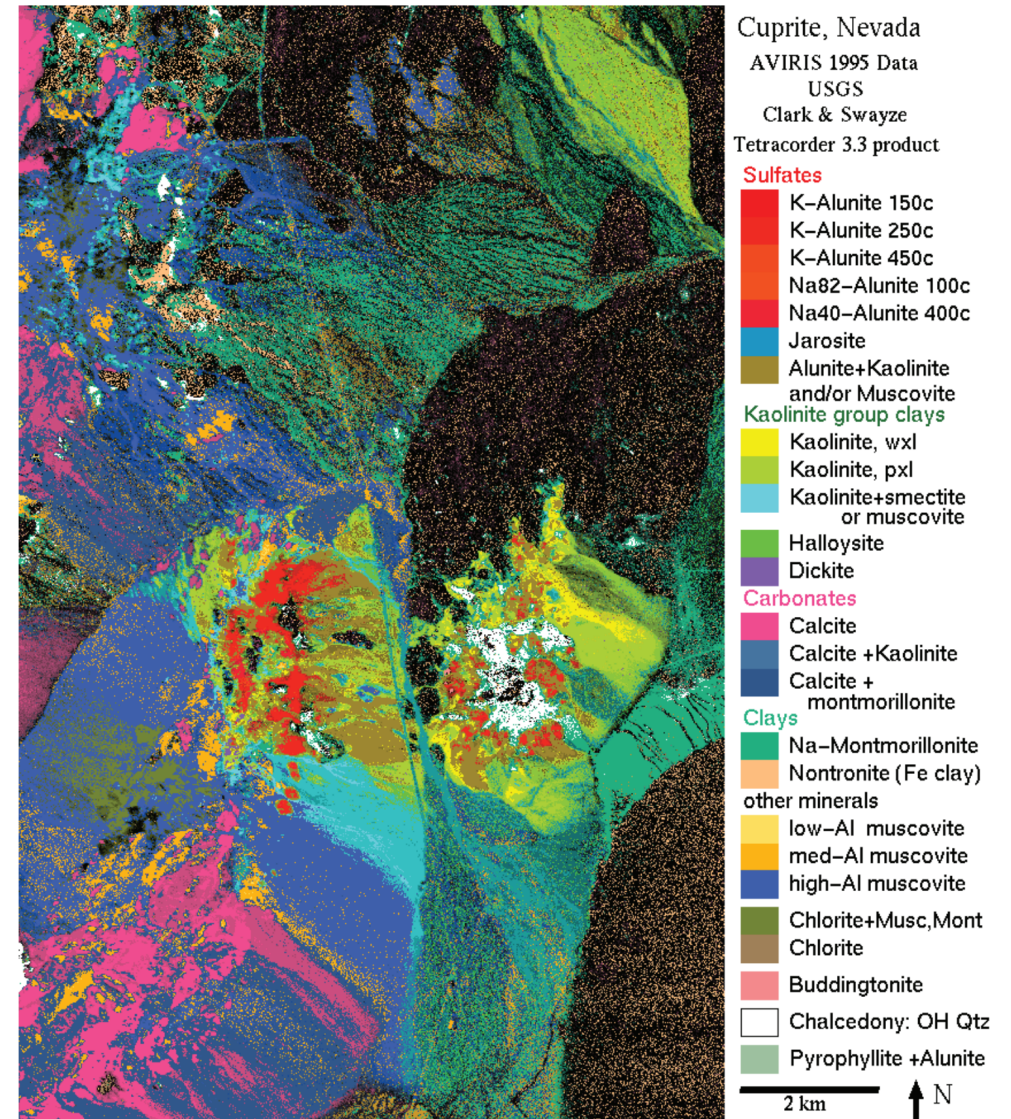
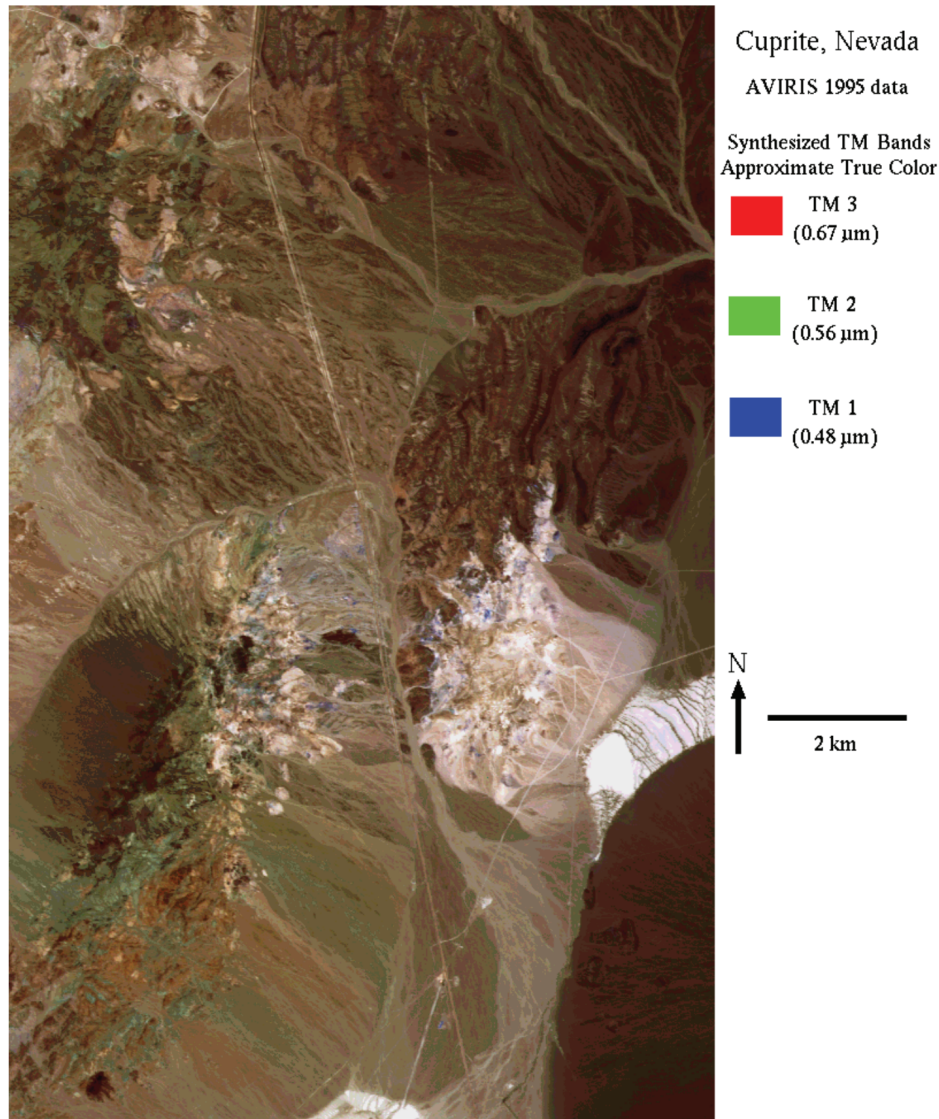


Figure 5. Mixing and unmixing of chest X-rays. (a) Original; (b) Mixture; (c) Recovered.

Source: https://www.researchgate.net/publication/323434106_Big_Data_Blind_Separation/link/5a963e05aca2721405695aca/download

Example: Mineral Identification via Hyperspectral Unmixing



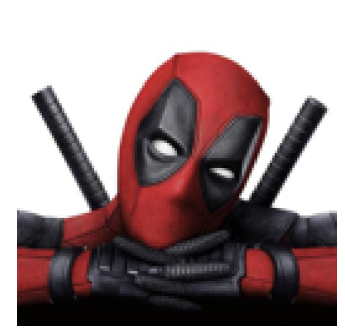
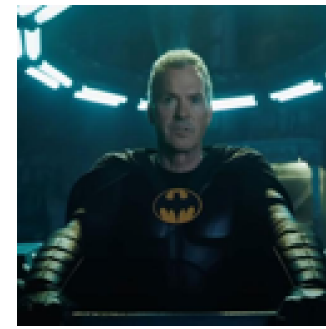
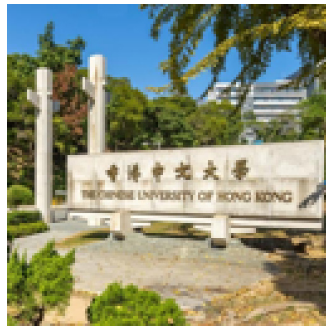
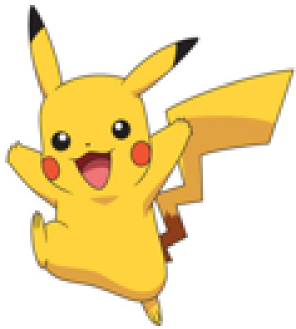
Source: <https://www.usgs.gov/labs/spectroscopy-lab/maps>

The One-At-A-Time Method, Modified

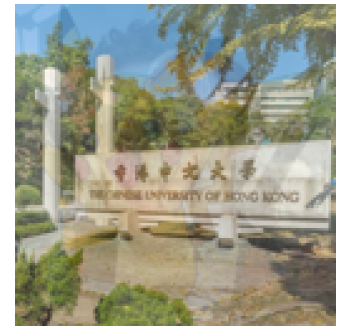
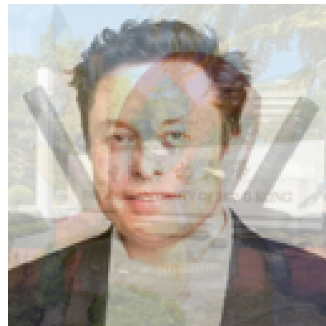
Modified Algorithm.

1. Given **input** observations X . Initialize a guess on the mixing matrix A_{est} .
2. **repeat, alternatingly:**
 - $S_{\text{est}} \leftarrow \min_S \|X - A_{\text{est}} S\|$, s.t. $S \geq 0$
 - $A_{\text{est}} \leftarrow \min_A \|X - A S_{\text{est}}\|$, s.t. $A \geq 0, A\mathbf{1} = \mathbf{1}$**until** some convergence criterion is met, or you have waited too long
3. **return:** $S_{\text{est}}, A_{\text{est}}$

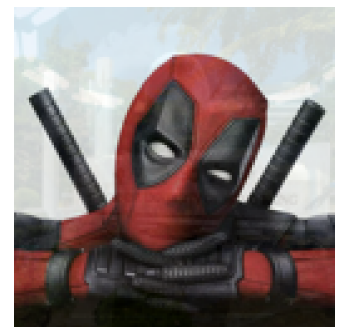
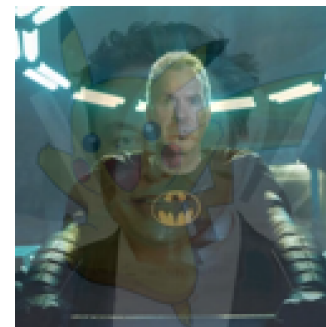
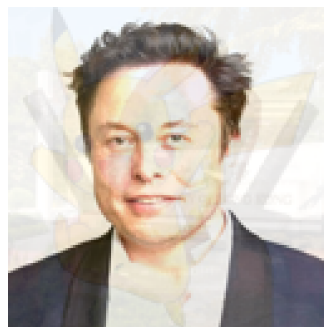
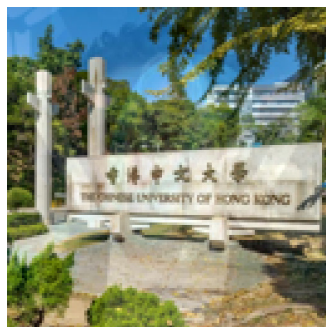
- we simply assert the non-negative constraints on the updating lines
- then we can play with a toy demo as shown the next slide
- FYI: this is called “non-negative matrix factorization” in the literature



(a) The source images



(b) The mixed observations



(c) The separated images

Take Home Points

- we have introduced the role of **calculus and algebra in A.I.**, and have seen their respective **applications in action**
- A.I. is a cool subject to study, probably will be a hot topic for the coming decade; but **mathematics is even more attractive**
- do not underestimate **the beauty of traditional wisdom!**

Take Home Points

- we have introduced the role of **calculus and algebra in A.I.**, and have seen their respective **applications in action**
- A.I. is a cool subject to study, probably will be a hot topic for the coming decade; but **mathematics is even more attractive**
- do not underestimate **the beauty of traditional wisdom!**

That's it! Questions?