Text Mining Tutorial 5:

Matrix Factorization: on Movie Recommender

Prof. Hsing-Kuo Pao
Teaching Assistant: Ghaluh Indah Permata Sari

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Outline

- Introduction to Matrix Factorization
- Task: Recommendation System
  - Case: Netflix

$$A \cdot B = \begin{pmatrix}1 & 3\\4 & 2\end{pmatrix} \cdot \begin{pmatrix}2 & 7\\3 & 5\end{pmatrix}$$

Introduction

- Matrix factorization is a simple embedding model.
- Given the feedback matrix $A \in \mathbb{R}^{m \times n}$, where m is the number of users (or queries) and n is the number of items, the model learns:
  - A user embedding matrix $U \in \mathbb{R}^{m \times d}$, where row i is the embedding for user i.
  - An item embedding matrix $\mathrm{V} \in \mathbb{R}^{m \times d}$, where row j is the embedding for item j.

Netflix Recommendation

<u>Users</u>


Gloom


Smiley


Happy


Chill

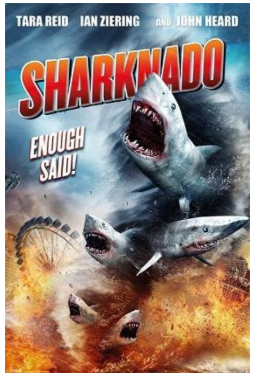
Movie 1

Movie 2

Movie 3

Movie 4

Movie 5



Netflix Ratings

How they obtain the rating?

Gloom

★★★★☆

Movie 5

	M1	M2	M3	M4	M5
					4

Rating Database

	M1	M2	M3	M4	M5
	1	3	2	5	4
	2	1	1	1	5
	3	2	3	1	5
	2	4	1	5	2

How do humans behave?

Table 1

	M1	M2	M3	M4	M5
😢	3	3	3	3	3
🙂	3	3	3	3	3
😂	3	3	3	3	3
😋	3	3	3	3	3

Table 2

	M1	M2	M3	M4	M5
😢	3	1	1	3	1
🙂	1	2	4	1	3
😂	3	1	1	3	1
😋	4	3	5	4	4

Table 3

	M1	M2	M3	M4	M5
😢	1	3	2	5	4
🙂	2	1	1	1	5
😂	3	2	3	1	5
😋	2	4	1	5	2

Unreal

😢 = 🙂 = 😂 = 😋

G S H C

M1 = M2 = M3 = M4 = M5

Let's check the dependencies

Let’s take an observe to this table

Table 2

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

The dependencies in Table 2

	M1	M2	M3	M4	M5
	3	1	1	3	1
	3	1	1	3	1

=

G H

	M1	M2	M3	M4	M5
	3			3	
	1			1	
	3			3	
	4			4	

M1 = M4

Movie 1

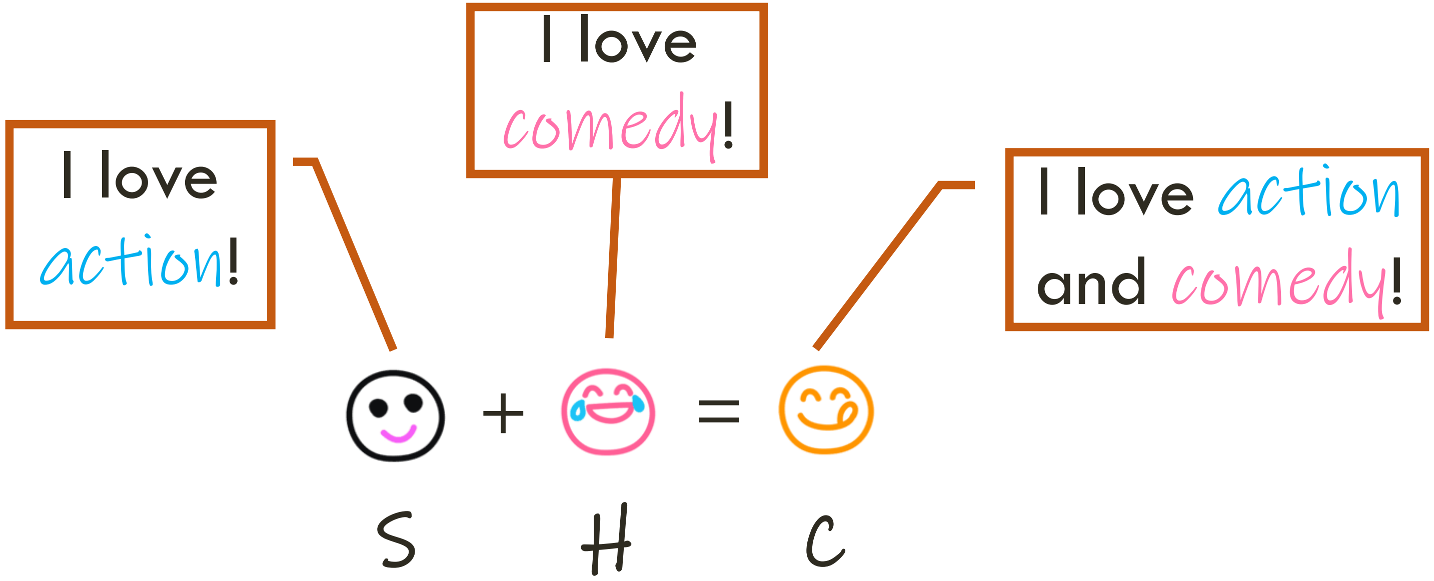
Movie 4

Analysis

- Netflix treated these users as the same person, in terms of preferences.
- Similar movies so that users gave them similar ratings.

The dependencies in Table 2

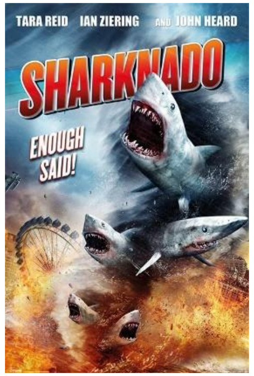
	M1	M2	M3	M4	M5
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4



	M1	M2	M3	M4	M5
		1	1		1
		2	4		3
		1	1		1
		3	5		4

M5 = Average(M2, M3)

Movie 5



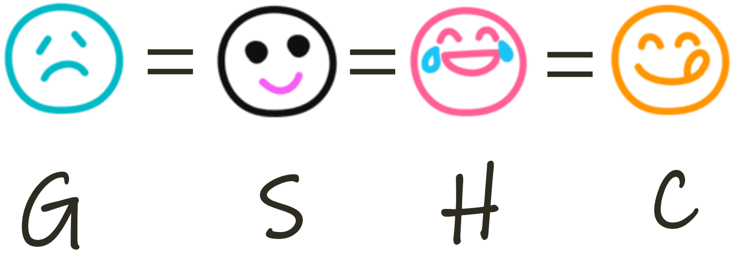
Movie 2

Movie 3

Rating prediction based on user's behavior

Table 1

	M1	M2	M3	M4	M5
	3	3	3	3	3
	3	3	3	3	3
	3	3	3	?	3
	3	3	3	3	3

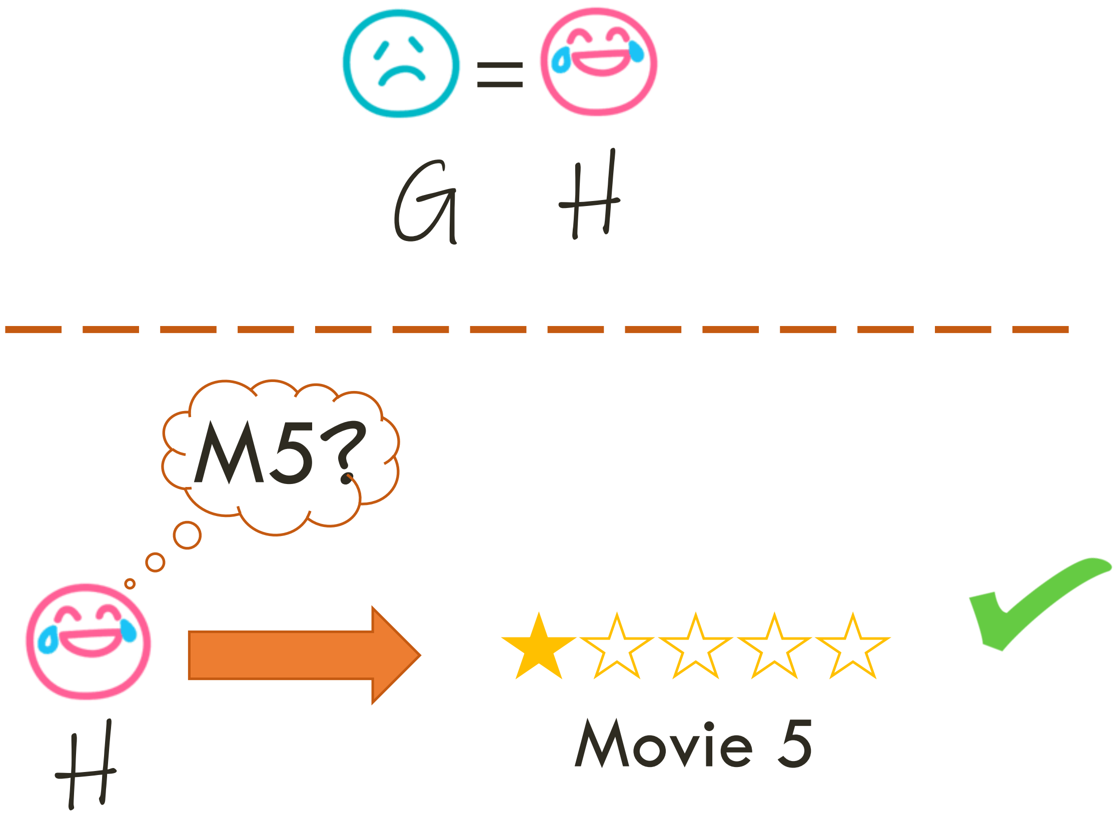


M1 = M2 = M3 = M4 = M5

Rating prediction based on user's behavior

Table 2

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	?
	4	3	5	4	4



Question: How do we figure out all these dependencies?

Answer: Matrix Factorization!!

Matrix Factorization

Factorization

$$7 \times 5 = 35$$

Matrix Factorization

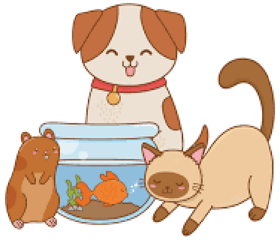
This x That =

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Features (attributes)

Big hurricanes

Words



Cute pets

Action

Leonardo Dicaprio

Drama

Comedy

J.K. Rowling

Horror

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Dot product

Example 1

	Comedy	Action	Rating
Gloom	✓	✗	
 Movie 1	3	1	★★★☆☆ 3

Dot product

Example 2

	Comedy	Action
Smiley	✗	✓
Movie 1	3	1

Rating

★☆☆☆☆

1

Dot product

Example 3

	Comedy	Action
Chill	✓	✓
 Movie 1	3	1

Rating

★★★★☆

4

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Examples

Categories

	Comedy	Action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

	Comedy	Action
	✓	✗
	✗	✓
	✓	✗
	✓	✓

✓ ✗

3 + 1

✓ ✗

3 + 1

✓ ✓

3 + 1

Movie 1

★★★☆☆

Movie 4

★★★☆☆

★★★★☆

Matrix Factorization

Factorization
$7 \times 5 = 35$

	Comedy	Action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

	Comedy	Action
	✓	✗
	✗	✓
	✓	✗
	✓	✓

=

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Matrix Factorization

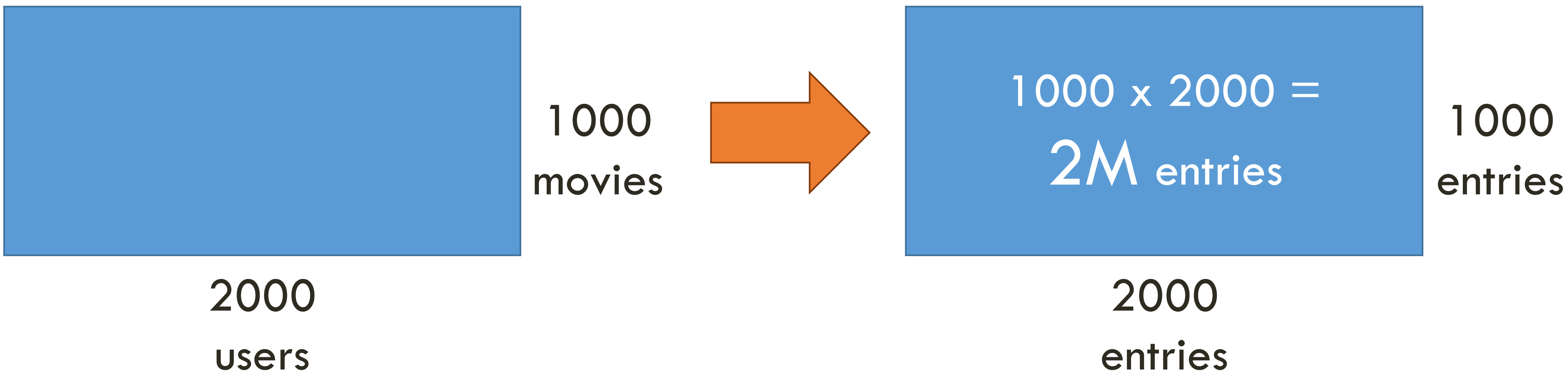
	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	2

	Comedy	Action
	✓	✗
	✗	✓
	✓	✗
	✓	✓

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Benefit Matrix Factorization

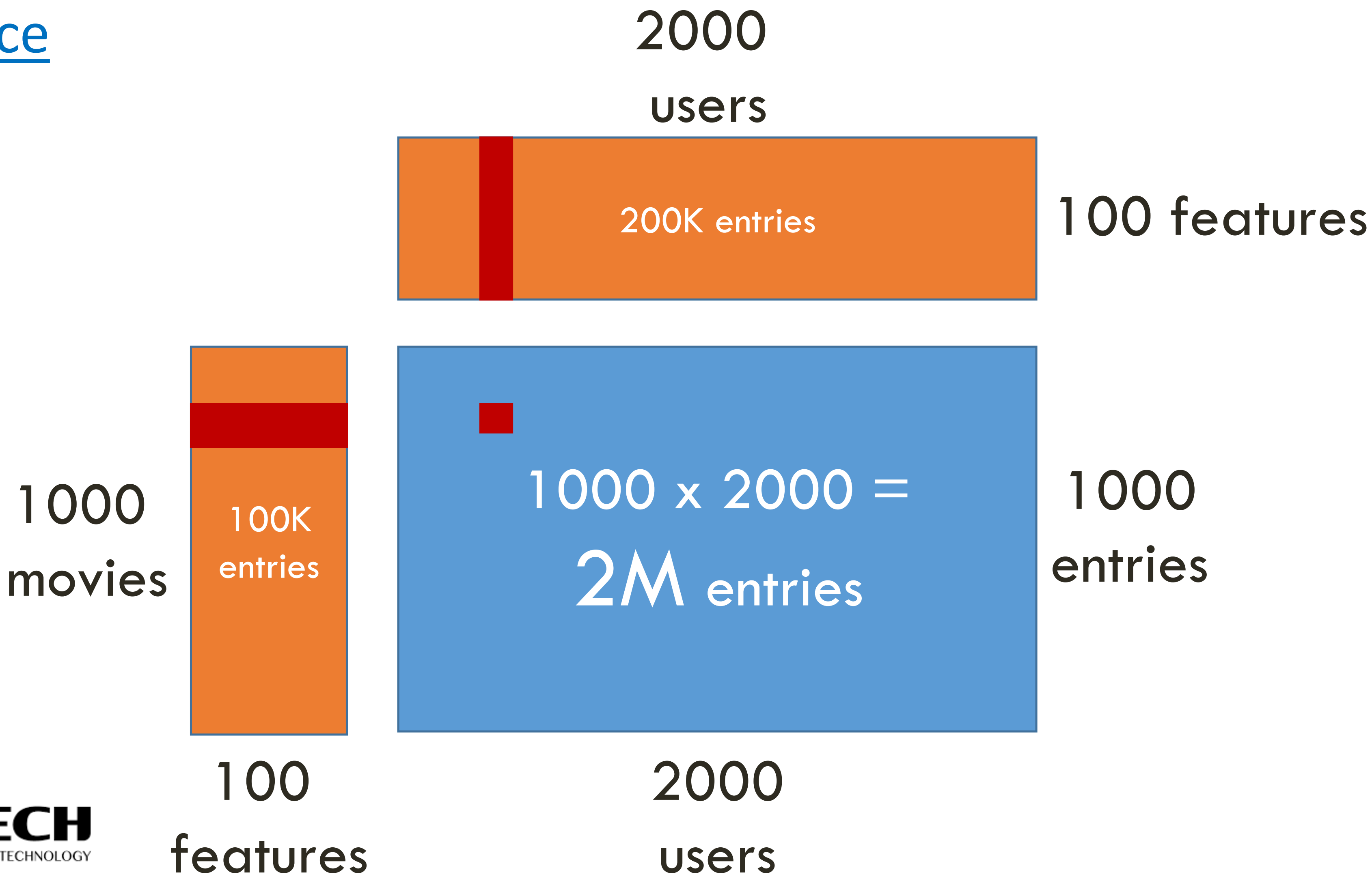
Save space



Matrix assumption

Benefit Matrix Factorization

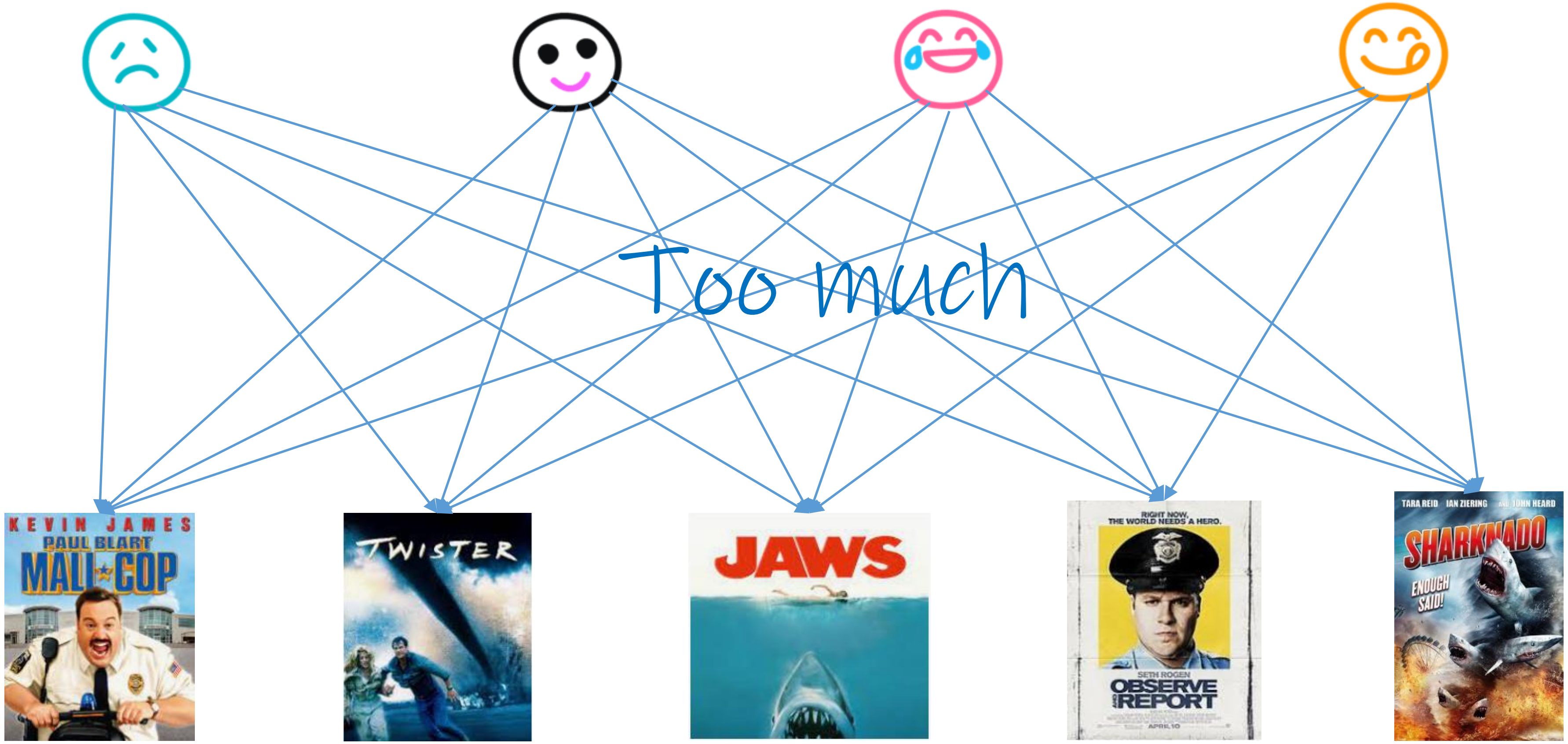
Save space



Benefit Matrix Factorization

Graph assumption

Save space

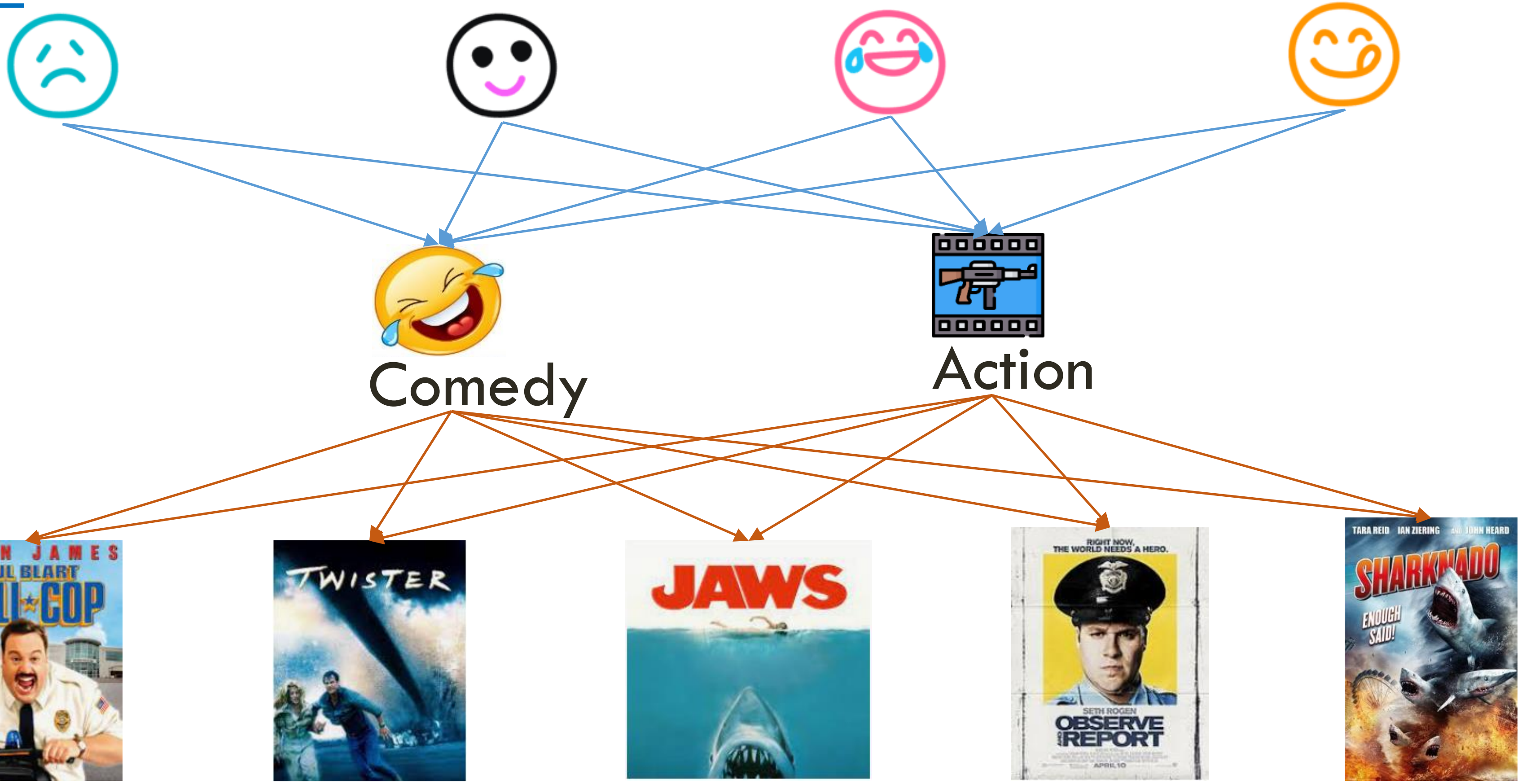


TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Benefit Matrix Factorization

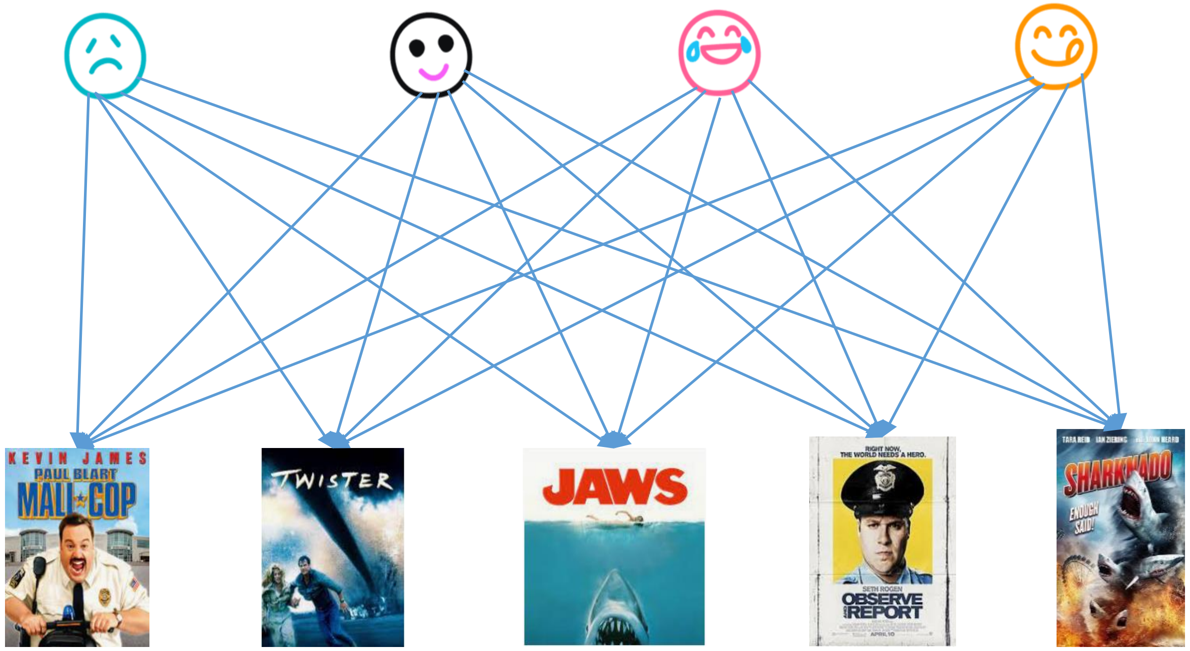
Graph assumption

Save space

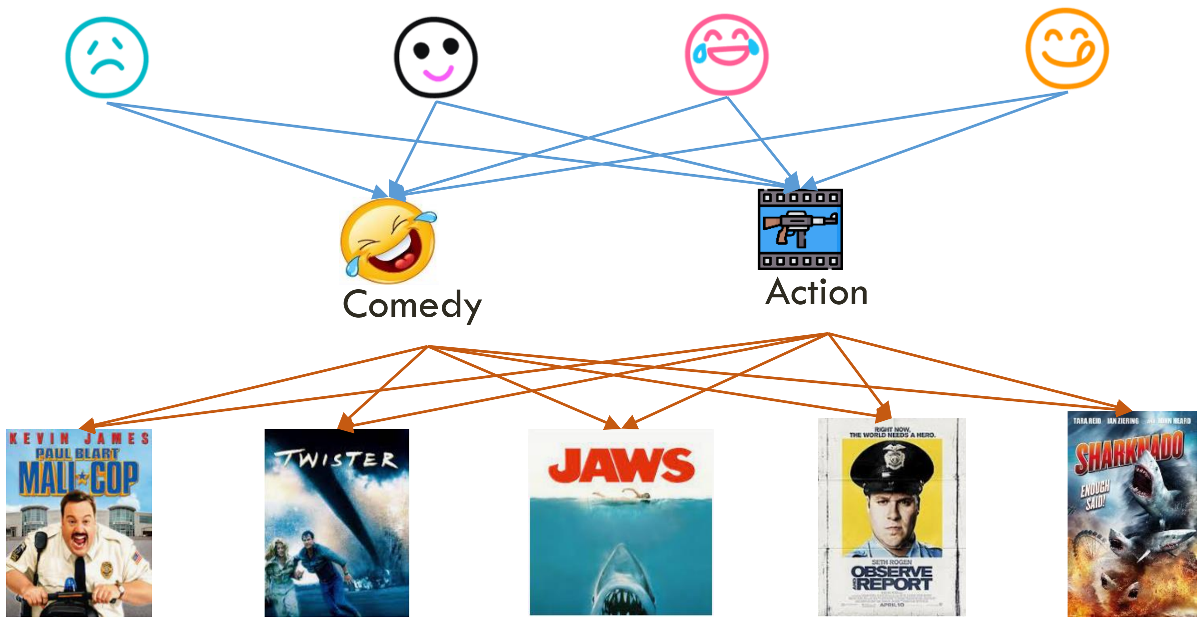


Benefit Matrix Factorization

Save space

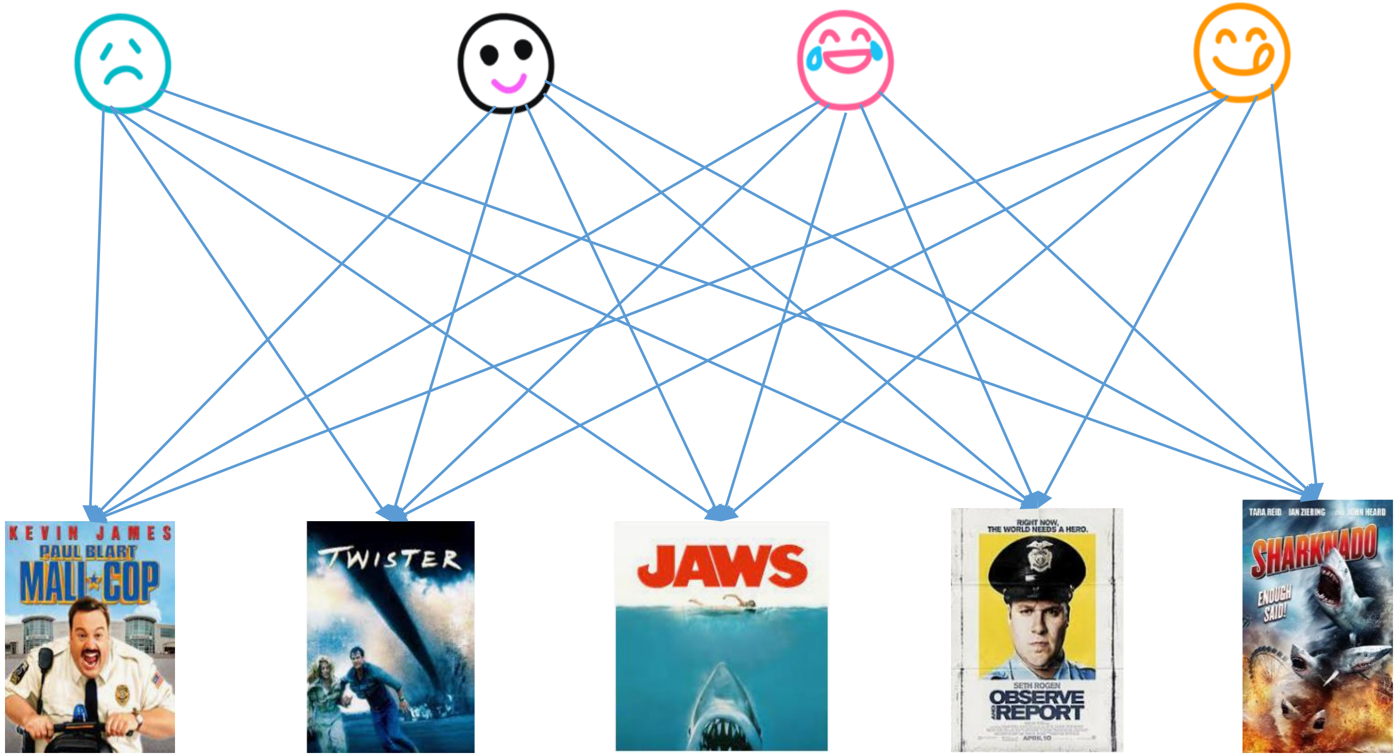


20 parameters



18 parameters

How many parameters can be saved?

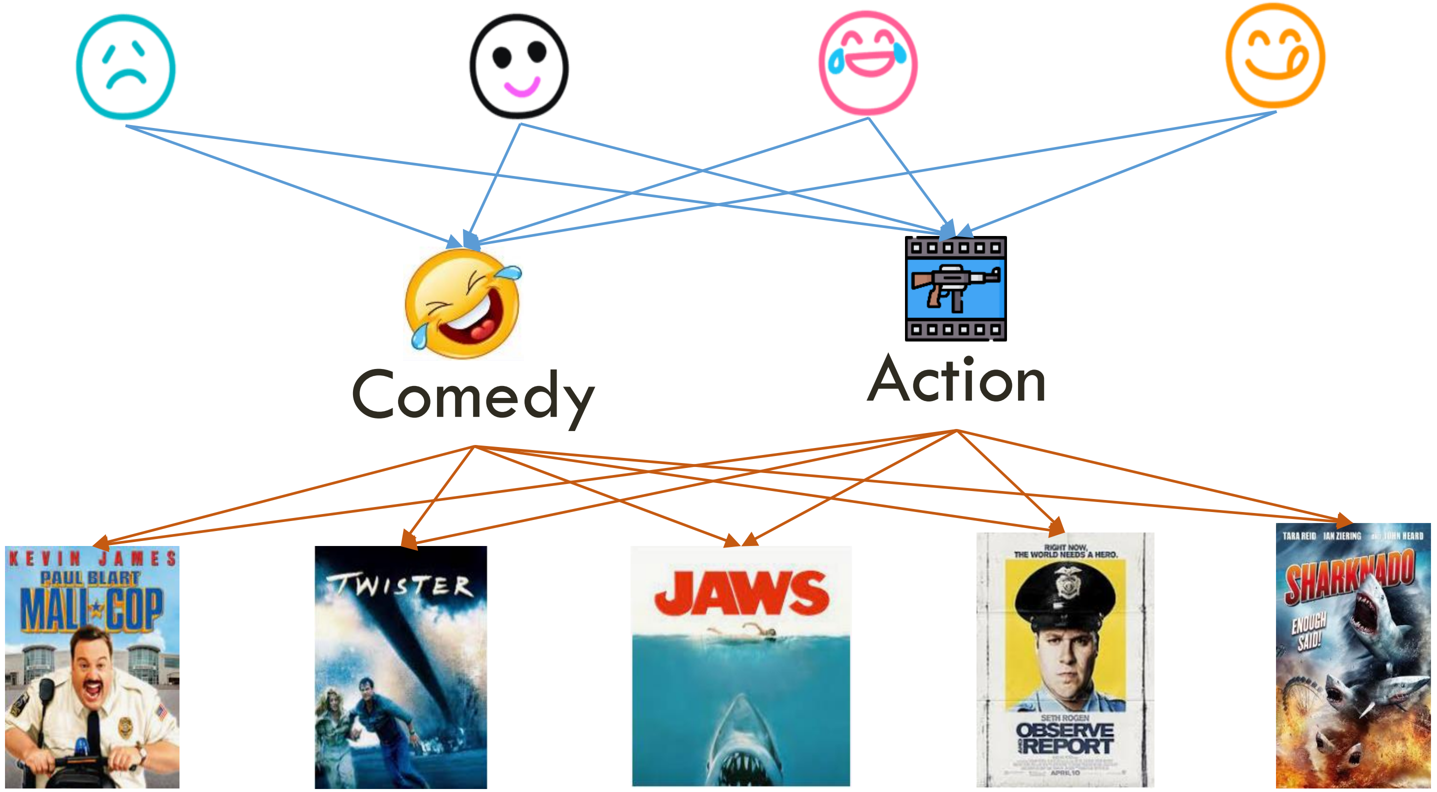


2000 users

2000 x 1000 = 2M parameters

1000 movies

How many parameters can be saved?



2000 users

2000 x 100 = 200K parameters

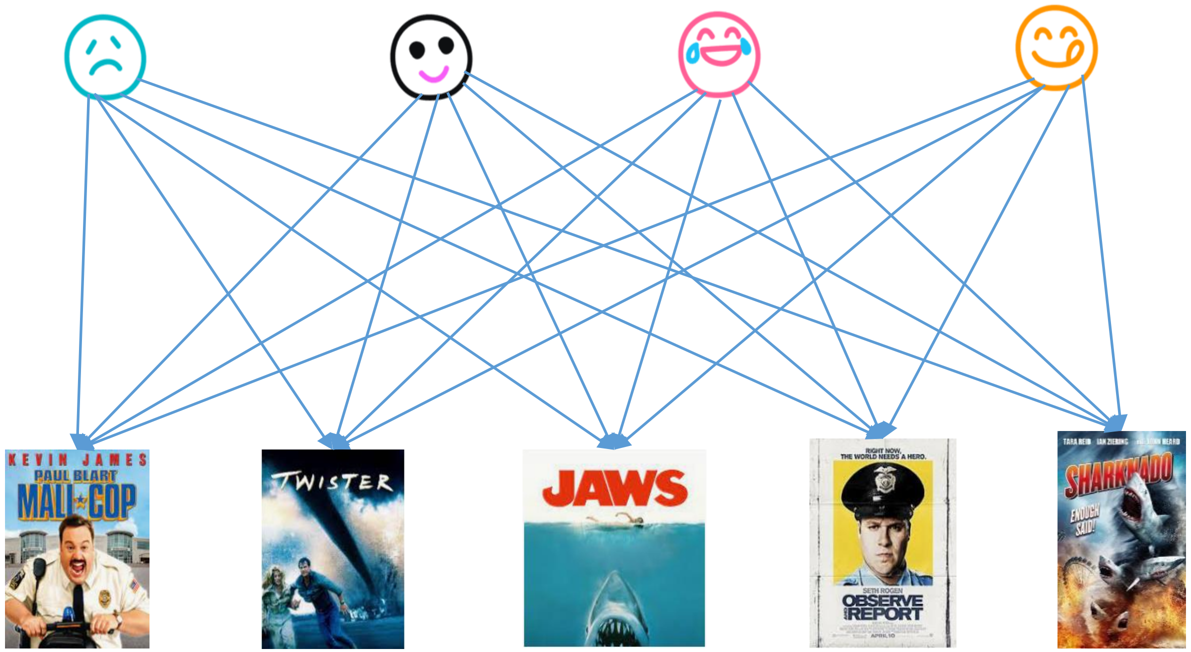
100 features

1000 x 100 = 100K parameters

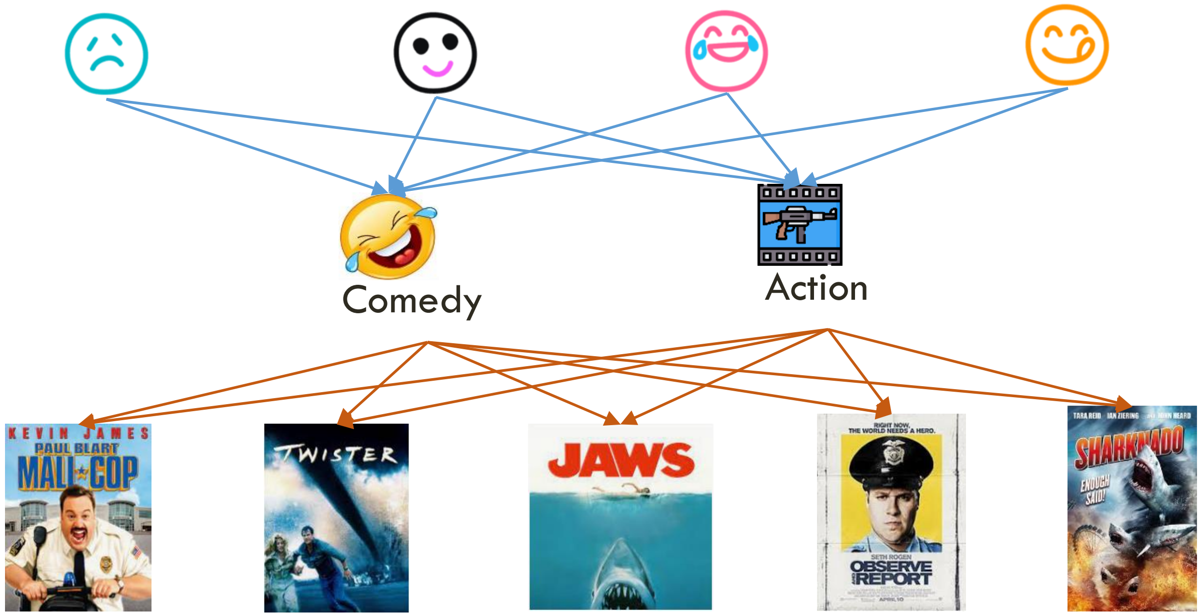
1000 movies

Total 300K features

How many parameters can be saved?

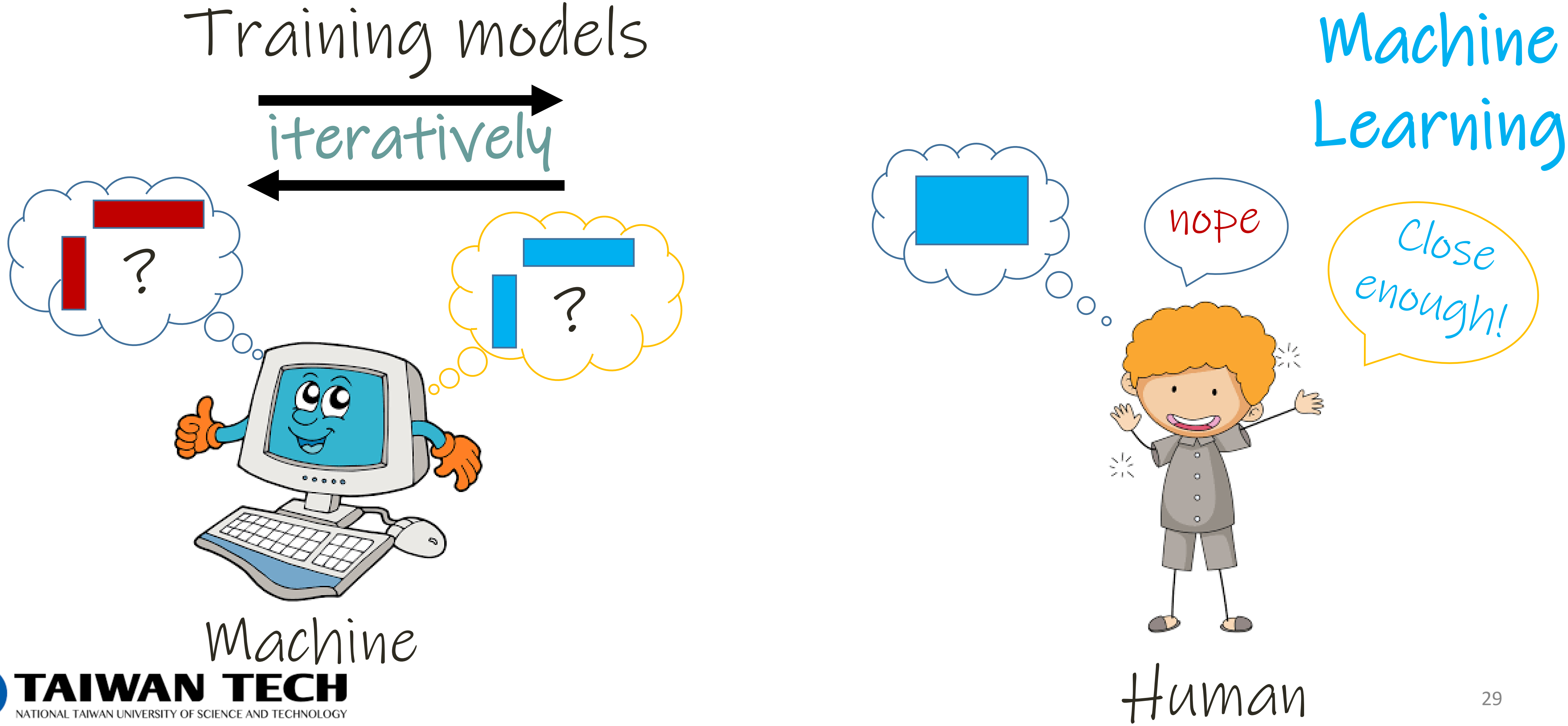


2M parameters



300K parameters

How to find the right factorization?



Example from Table 2

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	2

✓ 1

✗ 0

	Comedy	Action
	1	0
	0	1
	1	0
	1	1

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Example from Table 2

Calculate dot product

dot product

(1.2x0.2)+(2.4x0.5)=1.44

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

	F1	F2
	0.2	0.5
	0.3	0.4
	0.7	0.8
	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.37	2.26	0.7	0.59
	1.32	1.53	1.85	0.91	0.5
	2.76	3.37	3.73	2.07	1.02
	1.68	1.99	2.32	1.2	0.63

F1 & F2 = features (comedy& action)

Example from Table 2

Optimize the value

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

	F1	F2
	0.2	0.5
	0.3	0.4
	0.7	0.8
	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.83			

New values

$(1.4 \times 0.3) + (2.5 \times 0.6) = 1.92$

Assume: Close enough

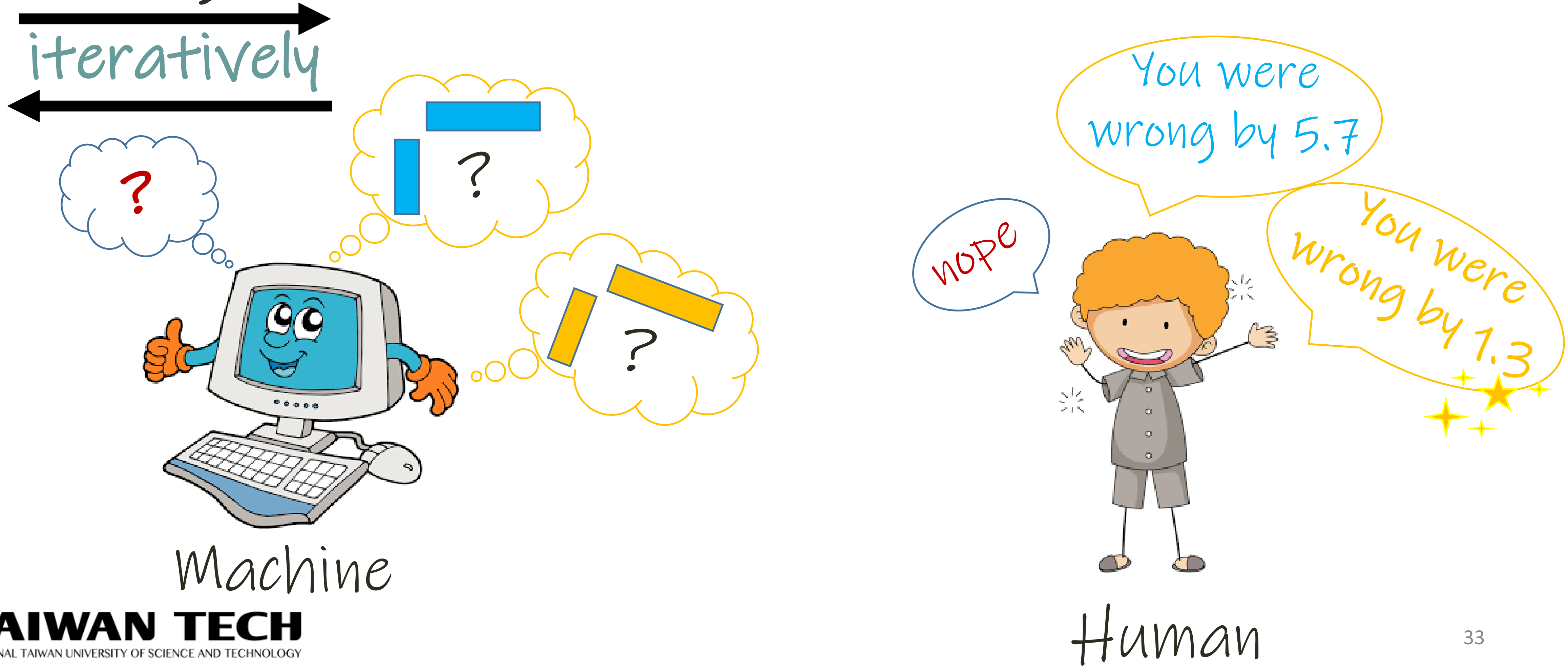
	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Target metrics

Let's learn about Error Function

Training models

Tell the machine, how badly is doing



Error function with Gradient Descent

Gradient Descent explanation:

https://www.youtube.com/watch?v=sDv4f4s2SB8

$$\text{Error} = (3 - 1.44)^2 + (1 - 1.37)^2 + \ldots$$

Derivative

Gradient Descent

	M1	M2	M3	M4	M5
F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

	F1	F2
	0.2	0.5
	0.3	0.4
	0.7	0.8
	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.37			

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Target metrics

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

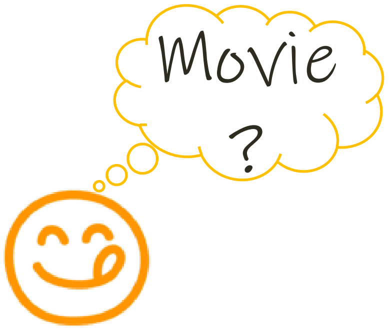
How to apply it in the movie recommender?

	M1	M2	M3	M4	M5
F1	3	1	1	3	1
F2	1	2	4	1	2

Let's fill in the blank

	F1	F2
	1	0
	0	1
	1	0
	1	1

	M1	M2	M3	M4	M5
	3		1		1
	1		4	1	
	3	1		3	1
		3		4	4



How to apply it in the movie recommender?

	M1	M2	M3	M4	M5
F1	3	1	1	3	1
F2	1	2	4	1	2

Let's fill in the blank

	F1	F2
	1	0
	0	1
	1	0
	1	1

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Recommend
M3

The highest rating

Gradient Descent

Gradient descent:

- an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function ($J(\theta) where\ \theta$ parameters (weights) of the model), typically a loss or cost function. *to maximize function: Gradient Ascent
- Commonly used in machine learning or deep learning (e.g. in Linear regression).

Function requirements:

- Differentiable
- Convex

Don't meet the requirement???

- Regularization (for non-convex, using $L1\ norm$ or $L2\ norm$)
- Non-differentiable functions (e.g subgradient descent)
- Non-convex functions (e.g. Stochastic Gradient Descent (SGD), Adam opt., Root Mean Square Propagation (RMSProp))
- Consider using another method: simulated annealing, particle swarm opt., genetic algorithm

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Gradient Descent

Differentiable

- If a function is differentiable it has a derivative for each point in its domain.

    *not all functions meet these criteria. First, let's see some examples of functions meeting this criterion:

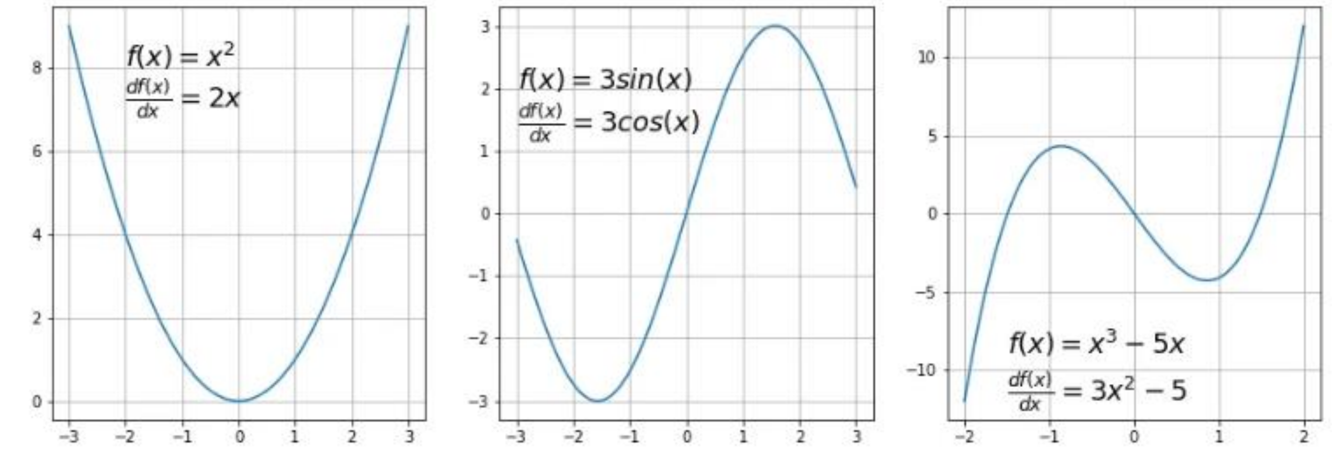


Figure 1. Differentiable function

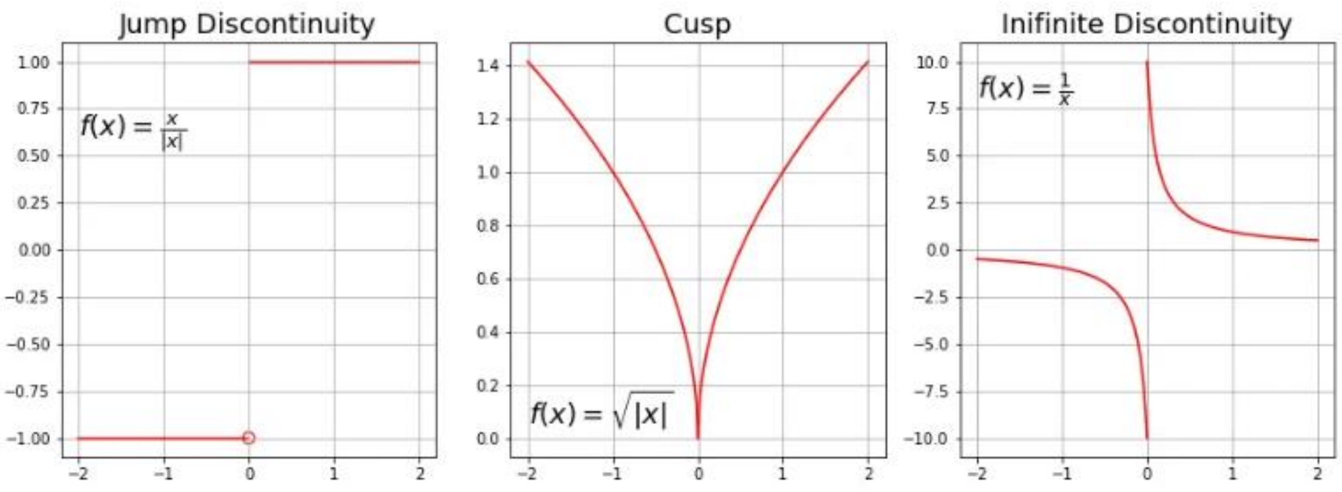


Figure 2. Non-Differentiable function

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Image source: https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

Gradient Descent

Convex

- Function has to be convex. For a univariate function, this means that the line segment connecting two function's points lays on or above its curve (it does not cross it). If it does it means that it has a local minimum which is not a global one.

Mathematically, for two points x_1, x_2 laying on the function's curve this condition is expressed as:

$$f\left(\lambda_{x_1} + (1-\lambda)_{x_2}\right) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

where λ denotes a point's location on a section line and its value has to be between 0 (left point) and 1 (right point), e.g. $\lambda = 0.5$ means a location in the middle.

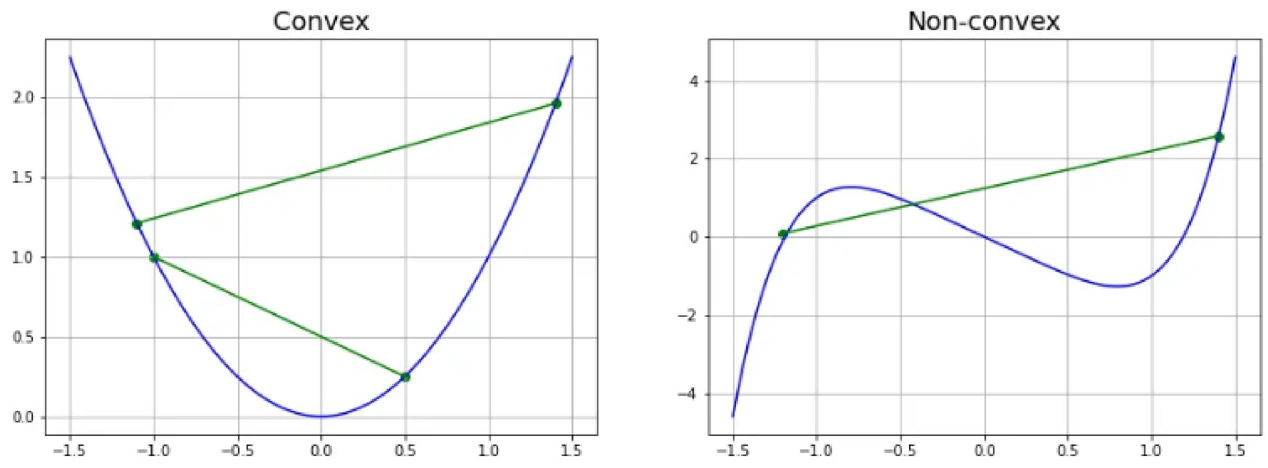


Figure 3. Convex and Non-convex function

Gradient Descent

Convex

Another way to check mathematically if a univariate function is convex is to calculate the second derivative and check if its value is always bigger than 0.

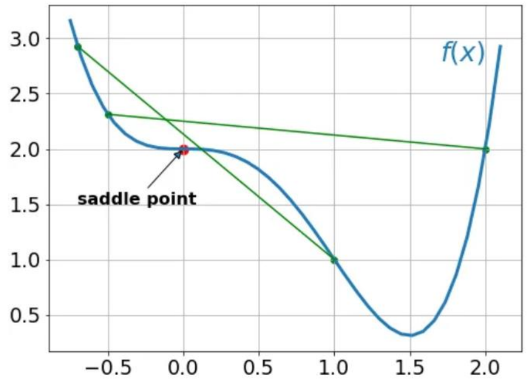
$$\frac{d^2 f(x)}{dx^2} > 0$$

Example of quasi-convex function:

$$f(x) = x^4 - 2x^3 + 2$$

To calculate the convex function, we can apply the quasi-convex function or saddle-points. For multivariate functions the most appropriate check if a point is a saddle point is to calculate a Hessian matrix.

$$\frac{df(x)}{dx} = 4x^3 - 6x^2 = x^2(4x - 6)$$



TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Gradient Descent

A gradient for an n-dimensional function $f(x)$at a given point p is defined as follows:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}$$

∇ is a so-called *nabla* symbol and you read it “del”.

➢ Approach to check convexity in our data:

- Graphical analysis
- Second derivative test
- Convex optimization solvers: CVXPY (python), CVX (matlab)
- Online convexity checkers

Gradient Descent

Steps to do:

1. choose a starting point (initialization)
2. Gradient calculation
   - makes a scaled step in the opposite direction to the gradient (objective: minimize).
3. Parameter update
   - Update rule: $\theta \coloneqq \theta + \alpha \nabla J(\theta)$, where α is the learning rate (hyperparameter to control the size of steps taken in the parameters space).
4. Iterative process
   - repeat points 2 and 3 until one of the criteria is met (Convergence/ stopping criteria):
     - maximum number of iterations reached
     - step size is smaller than the tolerance (due to scaling or a small gradient).

Gradient Descent

Gradient Descent Update Rule $\boxed{p_{n+1} = p_n - \eta \nabla f(p_n)}$

Purpose: to iteratively move the parameter values in the direction of the steepest descent of the objective function f to minimize the function.

Where,
p_{n+1} is an updated parameters value at iteration $n+1$.
p_n is the current parameter value at iteration n.
η is the learning rate
$\eta \nabla f(p_n)$ is the gradient of the objective function f with respect to parameter p_n at the current iteration.

Note:

- The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point.
- The bigger learning rate the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

TAIWAN TECH
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Thank you
Q & A