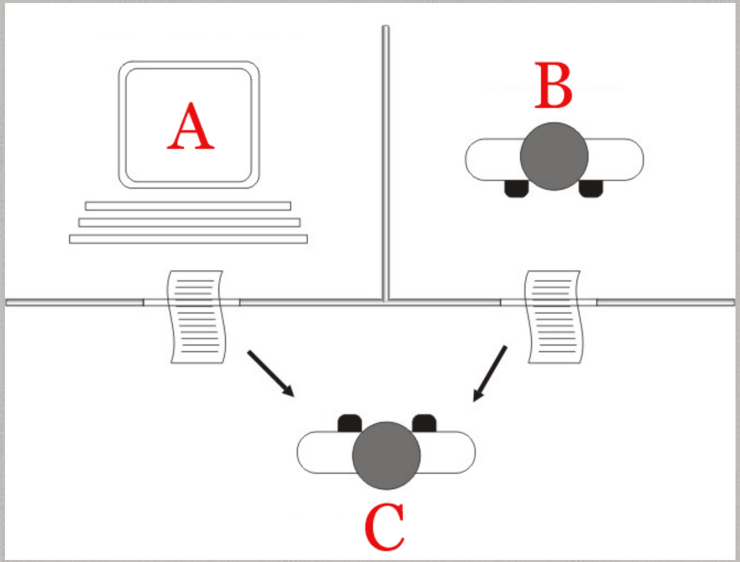
Machine Learning and Artificial Intelligence (Turing Test)

*The **Turing Test:*** proposed by ***Alen Turing*** (1950) in his paper "*Computer Machinery and Intelligence*".

Turing is widely considered to be the *father of theoretical computer science and artificial intelligence.*

- A test of a machine's ability to exhibit intelligent behavior equivalent to a human
- A human evaluator would judge natural language conversation between a human and a machine
- The evaluator would be aware that one of the two partners in conversation is a machine
- All participants would be separated from each other
- The conversation would be limited to a text-only channel
- If the evaluator can not reliably tell the machine from the human, the machine is said to have passed the test.



How about these?

God asked Abraham to sacrifice his son Isaac because he wanted to test his faith. Whose son and whose faith are we talking about?

Would you rather sacrifice one adult to save two children, or two children to save five adults, and why?

The following sentence is true.

The previous sentence is false.

Which of those two sentences is true?

To pass a Turing test, what capabilities do you think a computer program should possess?

To pass the Turing test, what capabilities do you think the computer should possess?

- Natural language processing (to communicate successfully)
- Knowledge representation (to store what is known or learnt)
- Automated reasoning (to use stored knowledge to answer questions and to draw new conclusions)
- Machine learning (to adapt to new situations, to detect new patterns and make prediction)

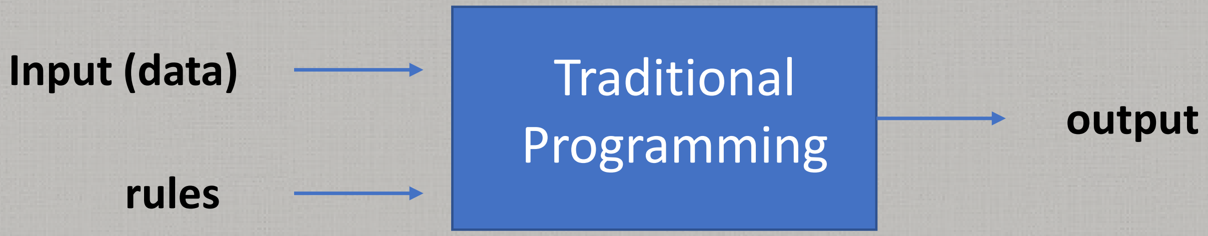
- ***Artificial Intelligence (AI)*** traditionally refer to an artificial creation of human-like agents that can learn, reason, plan, perceive or process natural language
- ***“general AI”*** : hypothetical and not domain specific, but can learn and perform tasks anywhere
- ***“narrow AI”*** : perform specific tasks within a domain (e.g., language translation, playing chess, prove mathematical theorems, diagnose diseases, auto driving, etc.)
- ***AI*** includes ***acquiring knowledge*** and ***applying knowledge*** to make decision
- ***ML*** only handles how to *acquire knowledge*: finding hidden patterns behind massive data

What is Machine Learning?

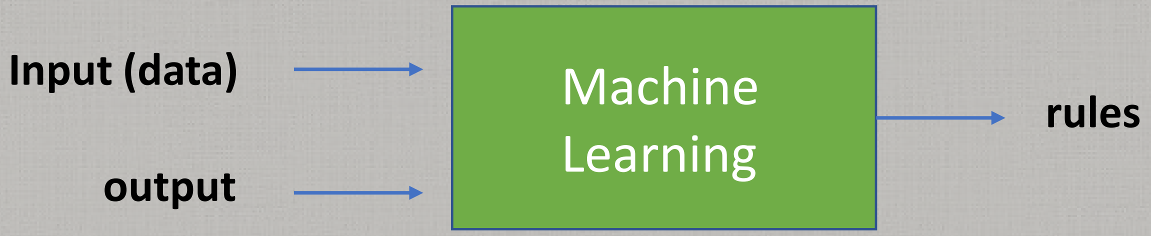
What is Machine Learning (loan approval example)

贷款审批

expert system



- Credit experts identify rules for loan approval
- Build the rules into computer program
- Run the program to approve or deny new applications



- Collect past loan application sample data, approved or denied
- Build a computer program to automatically identify the rules
- Use the rules to process new applications

What is Machine Learning

- Machine learning is *an application of artificial intelligence* (AI) that provides systems the *ability to automatically* ***learn and improve from experience*** *without explicitly programmed*

- Machine learning(ML) is *the scientific study of* ***algorithms*** and ***statistical models*** *that computer systems use to perform a specific task without using explicit instructions*, relying on patterns and inference instead.

- Machine learning algorithms build a ***mathematical model*** based on ***sample data***, known as “training data”, in order to make ***predictions*** or decisions without being explicitly programmed to perform the task.

- Machine learning is ***a tool for turning data into knowledge***

Examples of Machine Learning Applications

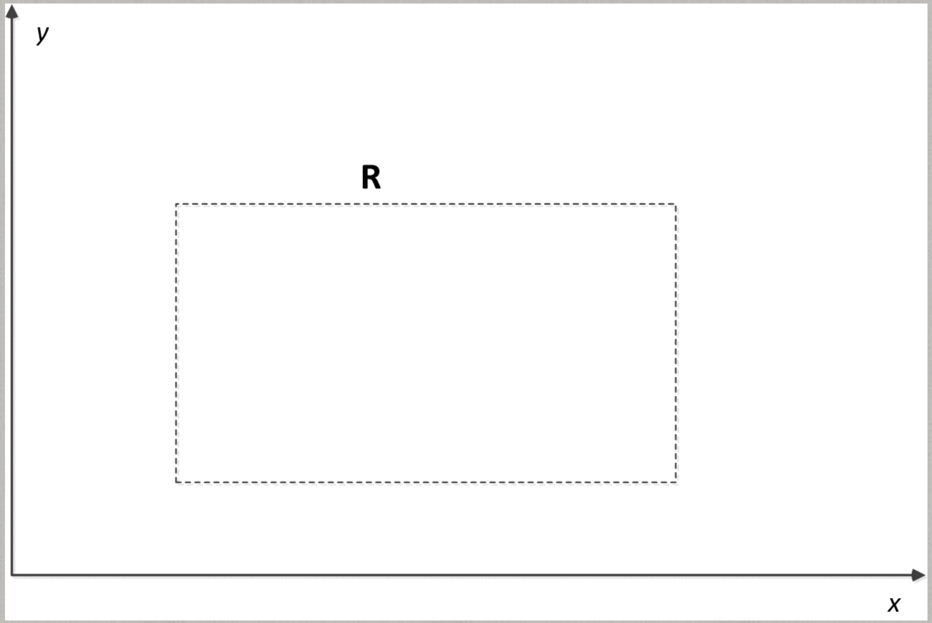
- Optical character recognition
- Email filtering
- Face recognition
- Speech recognition
- Product recommendation

- Loan applications evaluation
- Medical diagnosis
- Outlier (anomaly) detection
- Customer segmentation
- Documents clustering
- Bioinformatics

Key components of a Machine Learning Problem

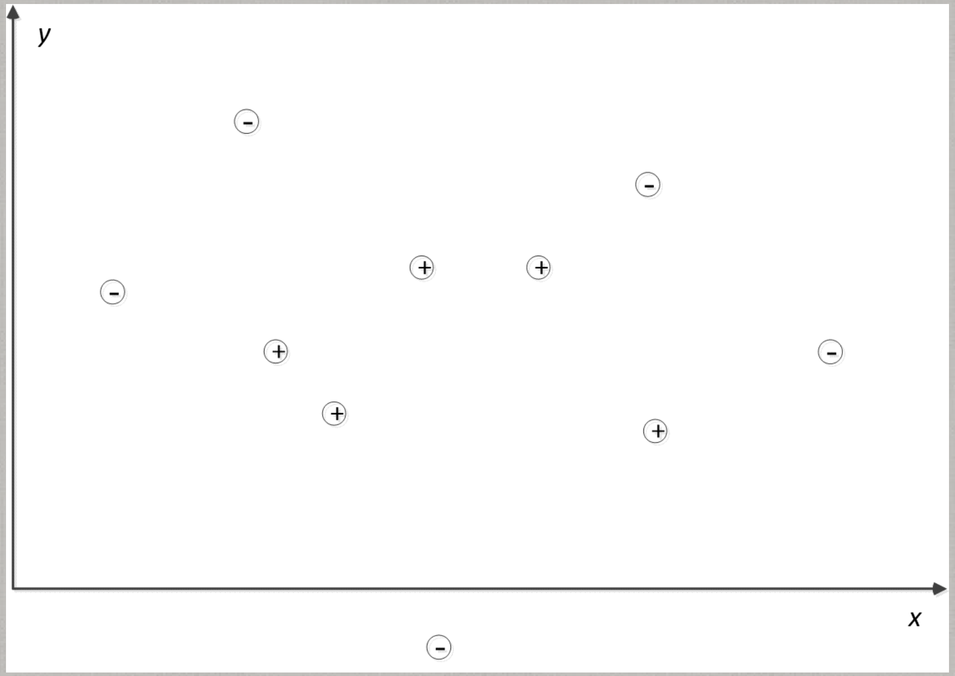
A Rectangle Learning Game

- The objective of the game is to learn an unknown axis-aligned rectangle R – ***target*** rectangle that separates points on the plane into two classes, **positive** (inside the rectangle) and **negative** (outside of the rectangle).



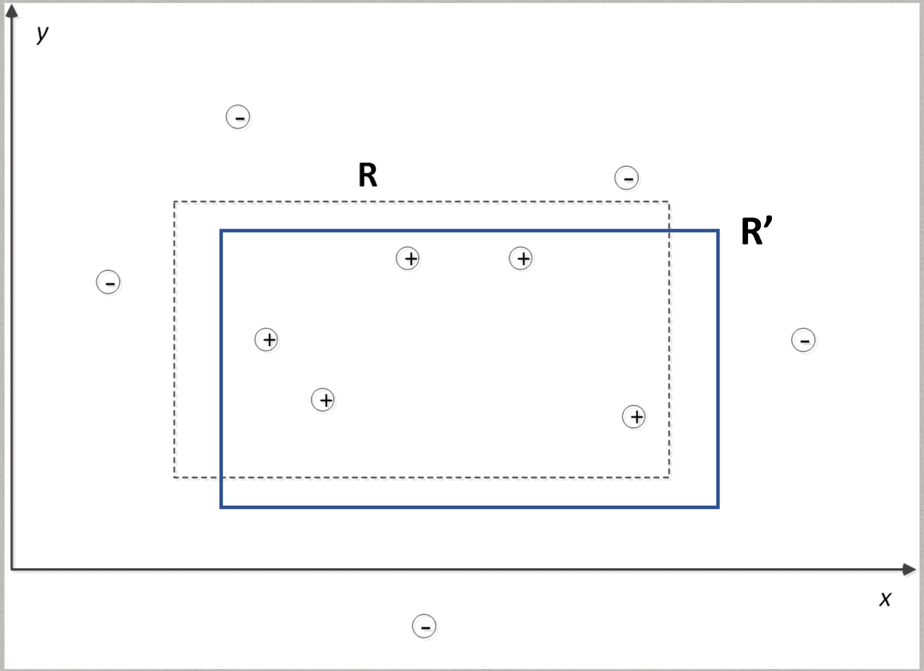
A Rectangle Learning Game

- The learner receives information about R through the following process (repeat n times):
  - a random point p is chosen in the plane according to some fixed probability distribution $\mathbb{D}$.
  - The learner is given the point p together with a label indicating whether p is contained in R (a positive example) or not contained in R (a negative example).



A Rectangle Learning Game

- The goal of the learner is to use as few examples as possible, and as little computation as possible, to pick a ***hypothesis*** rectangle R' that is a close approximation to R.
- The learner will choose R' from those not only separate given points but will also perform (predict) well on unseen data points (good **generalization**)



Key components of a Machine Learning Problem

- A true model (unknown) relationship between x and y
- A set of data to learn from (a random sample governed by the true model) dataset
- A set of hypothesis models
- An objective function to measure the learning error loss function

目标函数

The iris dataset

In UCI Machine Learning Repository

https://archive.ics.uci.edu/ml/index.php

Input (x)

Feature/predictor/attribute

Output (y)

	Feature #1	Feature #2	Feature #3	Feature #4	Target value

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	**5.1**	**3.5**	**1.4**	**0.2**	**0**
1	**4.9**	**3**	**1.4**	**0.2**	**0**
2	**4.7**	**3.2**	**1.3**	**0.2**	**0**
3	**4.6**	**3.1**	**1.5**	**0.2**	**0**
4	**5**	**3.6**	**1.4**	**0.2**	**0**
…					
50	**7**	**3.2**	**4.7**	**1.4**	**1**
51	**6.4**	**3.2**	**4.5**	**1.5**	**1**
52	**6.9**	**3.1**	**4.9**	**1.5**	**1**
53	**5.5**	**2.3**	**4**	**1.3**	**1**
54	**6.5**	**2.8**	**4.6**	**1.5**	**1**
…					
144	**6.7**	**3.3**	**5.7**	**2.5**	**2**
145	**6.7**	**3**	**5.2**	**2.3**	**2**
146	**6.3**	**2.5**	**5**	**1.9**	**2**
147	**6.5**	**3**	**5.2**	**2**	**2**
148	**6.2**	**3.4**	**5.4**	**2.3**	**2**
149	**5.9**	**3**	**5.1**	**1.8**	**2**

Instance/example/sample/observation

Iris Versicolor **Iris Setosa** **Iris Virginica**

Formalize the Machine Learning Problems

- ***The unknown truth***: In machine learning problems, our goal is to extract an ***unknown*** relationship $y = f(\boldsymbol{x})$ from data.
  - where the output $y \in \mathcal{R}$ (***target value***) is some quantity that can be predicted from an ***input*** $\boldsymbol{x} \in \mathcal{R}^d$,
  - where, $\boldsymbol{x} = [x_1, x_2, \dots, x_d]^T$ is the ***feature vector*** and d is the dimension of the feature vector.
  - 特性 实例
  - $x_i, i = 1,2, \dots, d$ are the ***features*** (attributes) of the ***instances*** (examples, objects).
- ***The given data set:*** We assume that we have access to a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, where $(\boldsymbol{x}_i, y_i)$ is the i^{th} point in the data set and is an example or instance (probably noisy) of the unknown mapping f to be learned.

$$\boldsymbol{x}_i = [x_{i1}, x_{i2}, \dots, x_{id}]^T$$

- ***The hypothesis model set:*** Since learning arbitrary function is intractable, we restrict 棘手的 ourselves to some ***hypothesis class*** H of allowable functions. Specifically, we typically employ a parametric model:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = g(\boldsymbol{x}, \boldsymbol{w})$$

Where $\boldsymbol{w} = [w_1, w_2, \dots, w_d]^T \in \mathcal{R}^d$, whose elements are known as parameters or ***weights***.

The ***hypothesis class*** is then the set of all functions induced by the possible choice of the parameter $\boldsymbol{w}$:

$$H = \{h_{\boldsymbol{w}} | \boldsymbol{w} \in \mathcal{R}^d\}$$

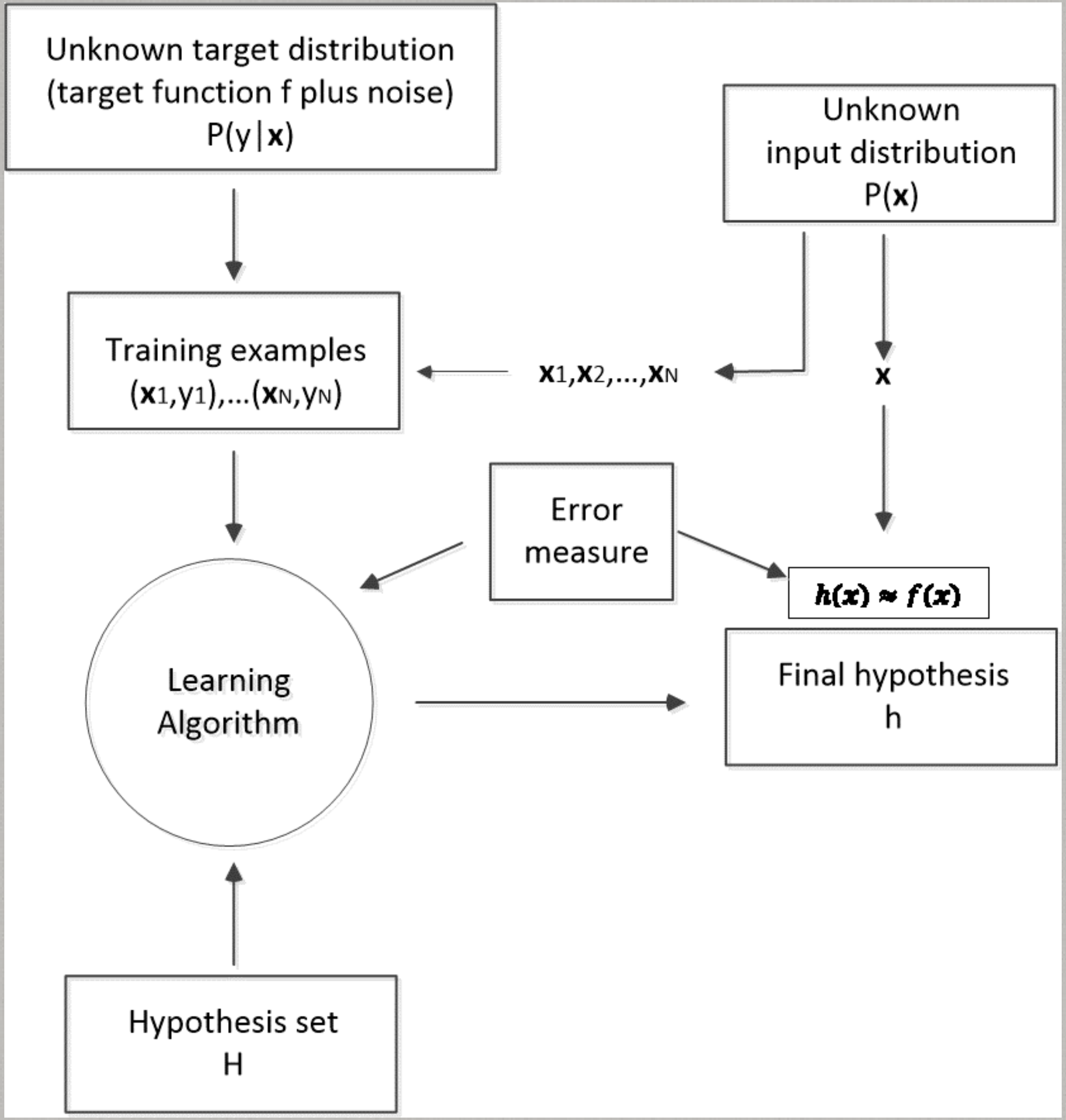
- After designating 指定 a ***loss (cost) function*** L, which measures how poorly the prediction $\hat{y}$ of the hypothesis matches the true output y, we can proceed to search for the parameter that best fit the data by minimizing this function:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{arg\,min}\{L(\boldsymbol{w})\}$$

$y = f(\vec{x})$

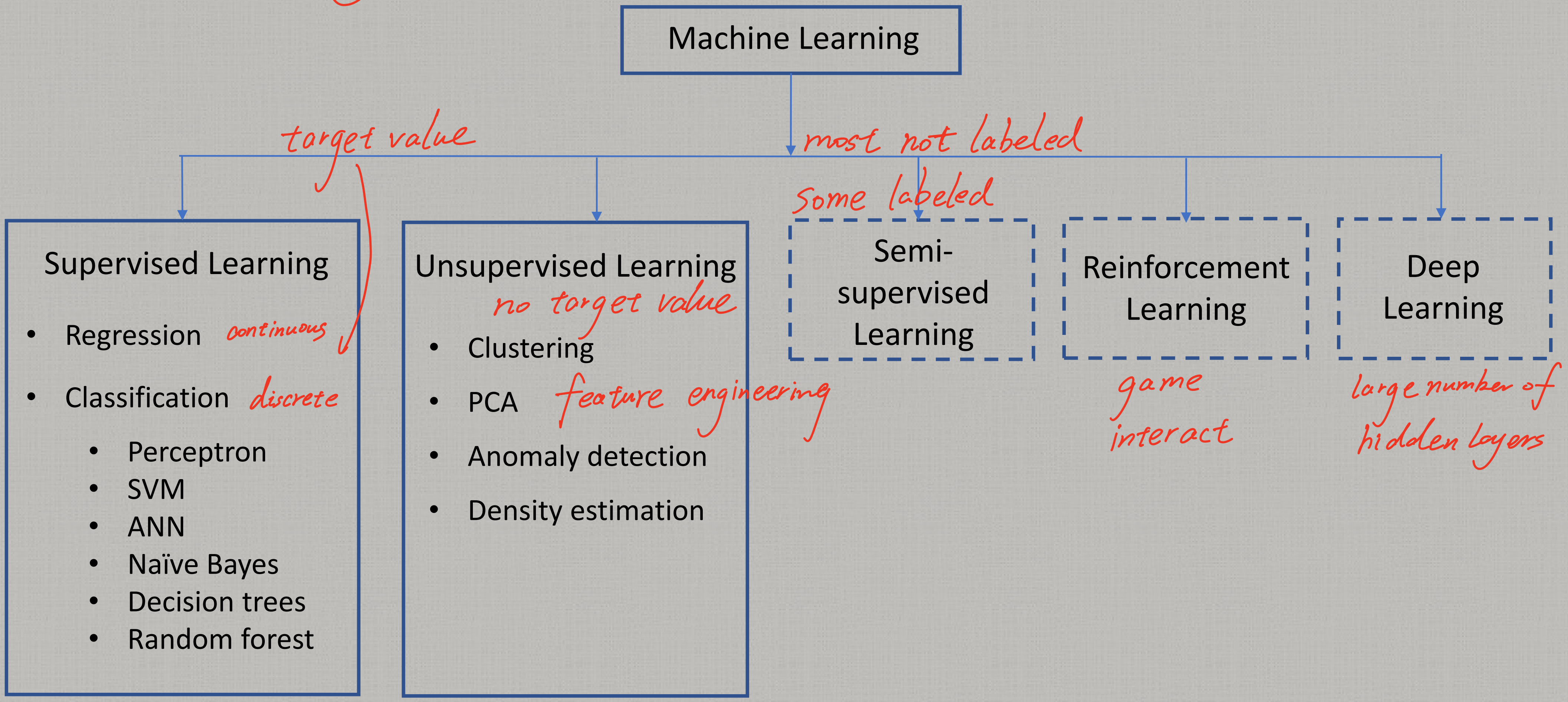
$\hat{y} = h_w(\vec{x})$

Machine Learning Problems



Unknown target distribution (target function f plus noise) P(y|**x**)
Unknown input distribution P(**x**)
Training examples (**x**1,y1),...(**x**N,yN)
x1,**x**2,...,**x**N
x
Error measure
h(x) ≈ f(x)
Learning Algorithm
Final hypothesis h
Hypothesis set H

Types of Machine Learning Problems



Machine Learning

target value

most not labeled

some labeled

Supervised Learning

- Regression continuous
- Classification discrete
    - Perceptron
    - SVM
    - ANN
    - Naïve Bayes
    - Decision trees
    - Random forest

Unsupervised Learning

no target value

- Clustering
- PCA feature engineering
- Anomaly detection
- Density estimation

Semi-supervised Learning

Reinforcement Learning

game interact

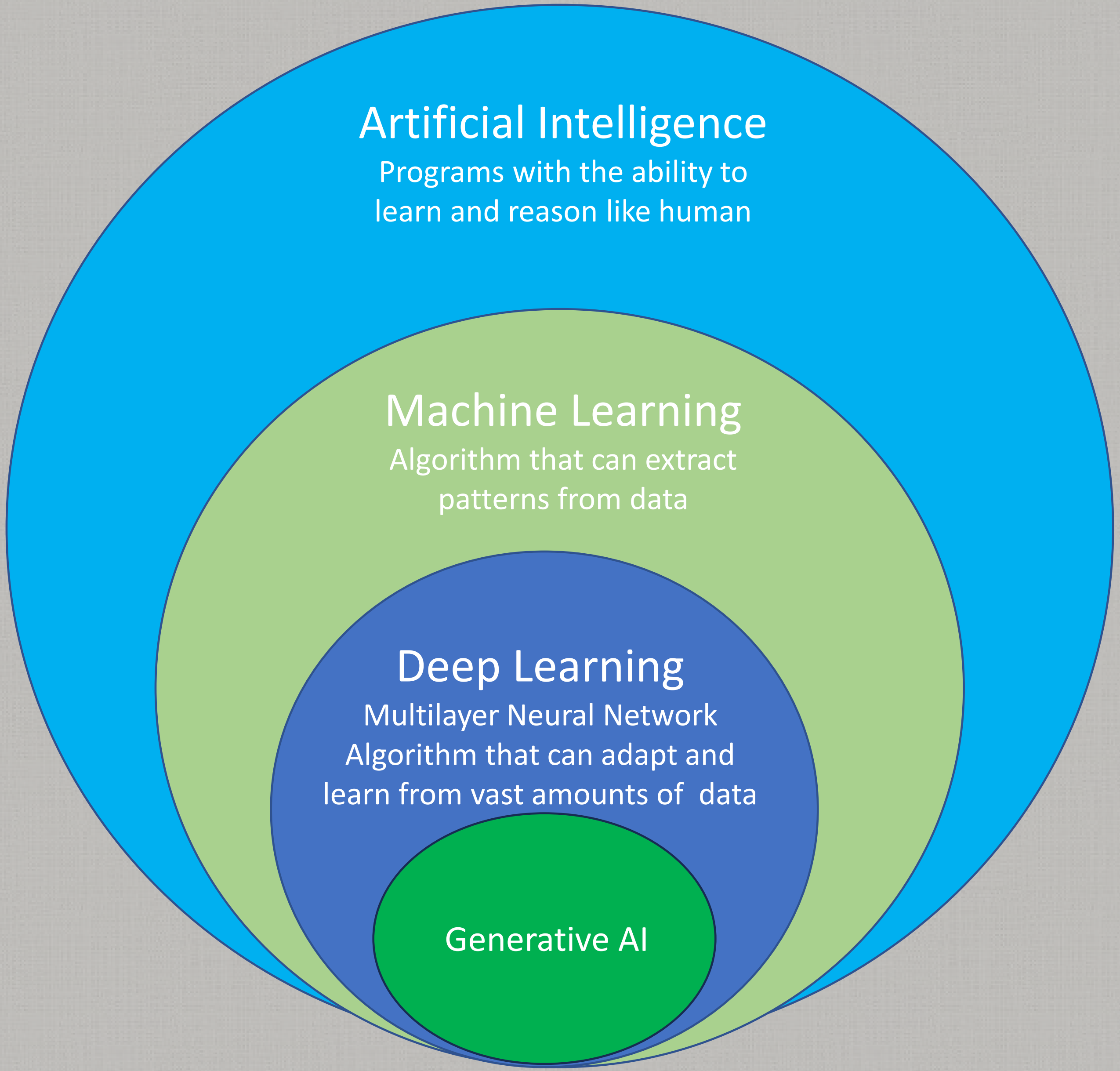
Deep Learning

large number of hidden layers

- ***Supervised learning***: the training data comprises examples of the input feature vectors along with their corresponding target values (labels).
  - When the target values are only from a finite number of discrete numbers, the problem is called ***classification***
  - When the target values are one or more continuous variables, the problem is called **regression**
- ***Unsupervised learning***: the training data consists of a set of input feature vectors without any corresponding target values.
  - ***Clustering***: discover groups of similar examples within the data
  - ***Anomaly detection***
  - ***Density estimate***: to determine the distribution of data within the input space
  - ***Visualization***: project the data from a high dimensional space down to the two or three dimensions (e.g., ***PCA***) for the purpose of visualization

- ***Semi-supervised learning:*** only a portion of the training data set are labeled, a large amount of the data are unlabeled. Unlabeled data can be used to improved the learning performance
- ***Reinforcement learning:*** there is no desired target values in the training data set. The algorithm interact with a dynamic environment that provides feedback in terms of rewards and punishments. The algorithm discover the optimal output through a process of trial and error.
- ***Deep learning*** is a class of artificial neural networks algorithm that uses multiple layers to progressively extract higher level features from the raw input and eventually make decisions.
  - Typical deep neural network types include *convolutional neural networks* (CNN), *deep belief network* and *recurrent neural networks* (RNN), etc.
  - Deep learning has been used in computer vision, speech recognition, natural language processing, machine translation, drug design, medical image analysis, and board game programs, etc.

Artificial Intelligence, Machine Learning and Deep Learning


Artificial Intelligence
Programs with the ability to learn and reason like human
Machine Learning
Algorithm that can extract patterns from data
Deep Learning
Multilayer Neural Network
Algorithm that can adapt and learn from vast amounts of data
Generative AI

Review of Vector Calculus and Linear Algebra

Consider two column vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}^n$,a matrix $\boldsymbol{A} \in \mathcal{R}^{m\times n}$, and a real number $\alpha \in \mathcal{R}$:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \boldsymbol{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Then,

$$\boldsymbol{x} + \boldsymbol{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

$$\alpha\boldsymbol{x} = \begin{bmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_n \end{bmatrix}$$

Consider two column vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}^n$,a matrix $\boldsymbol{A} \in \mathcal{R}^{m\times n}$, and a real number $\alpha \in \mathcal{R}$:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \boldsymbol{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Then,

$$\boldsymbol{x}^T = [x_1 \quad x_2 \quad \dots \quad x_n]$$

$$\boldsymbol{A}^T_{n\times m} = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}$$

Consider two column vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{R}^n$,a matrix $\boldsymbol{A} \in \mathcal{R}^{m\times n}$, and a real number $\alpha \in \mathcal{R}$:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \boldsymbol{A}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Then,

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y} = [x_1 \quad x_2 \quad \cdots \quad x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^n x_i y_i \quad \textbf{(\textit{inner product})}$$

$$\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n a_{1i} x_i \\ \sum_{i=1}^n a_{2i} x_i \\ \vdots \\ \sum_{i=1}^n a_{mi} x_i \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \quad \textbf{(\textit{linear transformation})}$$

- A ***norm*** of a real vector is a function $\|\cdot\|: \mathcal{R}^n \to \mathcal{R}$ that satisfies:
  - $\|\boldsymbol{x}\| \geq 0, \textit{with equality if and only if } \boldsymbol{x} = \boldsymbol{0}$
  - $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$
  - $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$ (***the triangle inequality***)
- We will typically be concerned with a few specific norms on $\mathcal{R}^n$:
  - 1-norm: $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ (*Taxicab norm*, ***L1 norm***)
  - 2-norm: $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$ (***Euclidean norm, L2 norm***)
  - p-norm: $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$ $(p \geq 1)$
  - Infinity-norm: $\|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$

- We will typically be concerned with a few specific norms on $\mathcal{R}^n$:
  - 1-norm: $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ (*Taxicab norm*, ***L1 norm***)
  - 2-norm: $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$ (***Euclidean norm, L2 norm***)
  - p-norm: $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}$ $(p \geq 1)$
  - Infinity-norm: $\|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$
- Note that 1-norm and 2-norm are special cases of the p-norm, and infinite-norm is the limit of the p-norm as $p \to \infty$
- ✓ A ***distance*** metric between two vectors $\boldsymbol{x}$ *and* $\boldsymbol{y}$ is then defined as: $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$

行列式

Determinant: (Laplace expansion)

- For a square matrix $\boldsymbol{A} \in \mathcal{R}^{n \times n}$, the determinant $|\boldsymbol{A}|$ is a scalar function of $\boldsymbol{A}$.

$$|\boldsymbol{A}| = \sum_{j=1}^{n} a_{kj} C_{kj}$$

  - Where, k is an arbitrary row, and C_{kj} is the ***cofactor*** of the element a_{kj}.

$$C_{kj} = (-1)^{k+j} M_{kj}$$

  - Where, M_{kj} is the ***minor*** associated with the element a_{kj} which is the ***determinant*** of the $(n-1) \times (n-1)$ matrix formed from $\boldsymbol{A}$ by crossing out the kth row and the jth column.

- Same method can be applied using a column instead of a row.

Determinant: (Laplace expansion)

- ***Example:*** calculate $|A|$ of $A = \begin{bmatrix} 2 & 4 & 1 \\ 0 & 1 & 0 \\ 2 & 2 & 3 \end{bmatrix}$ using *Laplace Expansion*

Properties of Determinant: For a square matrix $\boldsymbol{A} \in \mathcal{R}^{n\times n}$

- $|\boldsymbol{A}^T| = |\boldsymbol{A}|$
- $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}|$
- $|\boldsymbol{A}^{-1}| = |\boldsymbol{A}|^{-1}$
- $|\alpha\boldsymbol{A}| = \alpha^n|\boldsymbol{A}|$
- $|\boldsymbol{A}| = \prod_i \lambda_i(\boldsymbol{A})$, λ_i are the eigenvalues of matrix $\boldsymbol{A}$

特征值之和等于矩阵的迹

特征值之积等于矩阵行列式

矩阵的逆

inverse matrix

$A \in R^{n \times n} \Longrightarrow \underline{A^{-1}A = AA^{-1} = I}$

$(A^{-1})^{-1} = A$

$(\lambda A)^{-1} = \frac{1}{\lambda} A^{-1}$

$(AB)^{-1} = B^{-1}A^{-1}$

$(A^T)^{-1} = (A^{-1})^T$

$|A^{-1}| = \frac{1}{|A|}$

矩阵转置

transpose

$(A^T)^T = A$

$(A+B)^T = A^T + B^T$

$(cA)^T = c \cdot A^T$

$|A^T| = |A|$

$\vec{a} \cdot \vec{b} = \vec{a}^T \vec{b}$

矩阵乘法

$(AB)C = A(BC)$

$\lambda(AB) = (\lambda A)B = A(\lambda B)$

$A(B+C) = AB + AC$

$(B+C)A = BA + CA$

Eigenvalue and eigenvector: 特征值和特征向量

- For a square matrix $\boldsymbol{A} \in \mathcal{R}^{n \times n}$, we say that a nonzero vector $\boldsymbol{x} \in \mathcal{R}^n$ is an eigenvector of $\boldsymbol{A}$ corresponding to eigenvalue λ if $\boldsymbol{Ax} = \lambda \boldsymbol{x}$. Then,
  - for any $\gamma \in \mathcal{R}$, x is an eigenvector of $\boldsymbol{A} + \gamma I$ with eigenvalue $\lambda + \gamma$
  - If $\boldsymbol{A}$ is invertible 可逆的, then $\boldsymbol{x}$ is an eigenvector of $\boldsymbol{A}^{-1}$ with eigenvalue λ^{-1}
  - $\boldsymbol{A}^k \boldsymbol{x} = \lambda^k \boldsymbol{x}$ for any $k \in \mathcal{Z}$ (where $\boldsymbol{A}^0 = \boldsymbol{I}$ by definition)

Example:

- Find the eigenvalue of $\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$

$A\vec{x} = \lambda\vec{x}$

$|A - \lambda I| = 0 \Rightarrow \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = 0 \Rightarrow (2-\lambda)^2 - 1 = 0$

$$|\boldsymbol{A} - \lambda I| = 0 \Rightarrow \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = 0 \Rightarrow (2-\lambda)^2 - 1 = 0$$

$$\Rightarrow \lambda^2 - 4\lambda + 3 = 0 \Rightarrow \lambda_1 = 1, \lambda_2 = 3$$

$\Rightarrow \lambda^2 - 4\lambda + 3 = 0 \Rightarrow (\lambda - 3)(\lambda - 1) = 0 \Rightarrow \lambda_1 = 1, \lambda_2 = 3$

- ***Orthogonal matrices:*** 正交矩阵
  - Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are said to be ***orthogonal*** if $\langle \boldsymbol{x}, \boldsymbol{y}\rangle = \mathbf{0}$
  - A square matrix $\mathbf{Q} \in \mathcal{R}^{n\times n}$ is said to be ***orthogonal*** if its columns are pairwise orthogonal unit (***orthonormal***) vectors. This definition implies that

$$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I} \Rightarrow \boldsymbol{Q}^T = \boldsymbol{Q}^{-1}$$

  - Orthogonal matrices preserve inner products:

$$(\boldsymbol{Q}\boldsymbol{x})^T(\boldsymbol{Q}\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{y}$$

  - They also preserve 2-norm:

$$\|\boldsymbol{Q}\boldsymbol{x}\|_2 = \sqrt{(\boldsymbol{Q}\boldsymbol{x})^T(\boldsymbol{Q}\boldsymbol{x})} = \sqrt{\boldsymbol{x}^T\boldsymbol{x}} = \|\boldsymbol{x}\|_2$$

$n\times n,\ n\times 1 \to n\times 1$

- ***Eigendecomposition:*** 特征分解

Given a squared matrix $\boldsymbol{A} \in \mathcal{R}^{n\times n}$ with n linearly independent eigenvectors (diagnosable), then,

$$\boldsymbol{A} = \boldsymbol{Q\Lambda}\mathbf{Q}^{-1}$$

Where,

- $\boldsymbol{Q} = [\boldsymbol{q_1} \quad \cdots \quad \boldsymbol{q_n}]$ is a squared matrix with $\boldsymbol{q_1}$ … , $\boldsymbol{q_n}$, as its columns
- $\boldsymbol{q_1}$ … , $\boldsymbol{q_n}$ are the eigenvectors, $\lambda_1, \dots, \lambda_n$ are the eigenvalues.
- $\boldsymbol{\Lambda} = diag(\lambda_1, \dots, \lambda_n)$

↑
normalized eigenvectors

- ***Symmetric matrices:***
  - A squared matrix $\boldsymbol{A}$ is said to be ***symmetric*** if it is equal to its own transpose ($\boldsymbol{A} = \boldsymbol{A}^T$)
  - **Eigendecomposition:** Given a symmetric matrix $\boldsymbol{A} \in \mathcal{R}^{n \times n}$, then,

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$

Where,

- $\boldsymbol{Q} = [\boldsymbol{q_1} \quad \cdots \quad \boldsymbol{q_n}]$ is an orthogonal matrix with $\boldsymbol{q_1} \dots , \boldsymbol{q_n}$, as its columns
- $\boldsymbol{q_1} \dots , \boldsymbol{q_n}$ are the orthonormal basis of eigenvectors, $\lambda_1, \dots , \lambda_n$ are the eigenvalues.
- $\boldsymbol{\Lambda} = diag(\lambda_1, \dots , \lambda_n)$

- ***Eigendecomposition: Example***

Let's find the eigendecomposition of the squared matrix:

$$\boldsymbol{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$|A - \lambda I| = \begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = (2-\lambda)^2 - 1 = 4 - 4\lambda + \lambda^2 - 1$$

$$= \lambda^2 - 4\lambda + 3 = (\lambda - 3)(\lambda - 1) = 0$$

$$\Longrightarrow \lambda_1 = 1, \quad \lambda_2 = 3$$

We already find the eigenvalues of this matrix as $\lambda_1 = 1, \lambda_2 = 3$.

To find the eigenvector corresponding to eigenvalue $\lambda_1 = 1$, we substitute $\lambda_1 = 1$ to $A\boldsymbol{x} = \lambda\boldsymbol{x}$,

$$A\vec{x} = \underset{\overset{\shortparallel}{1}}{\lambda_1}\vec{x}$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Which is equivalence to:

$$2x_1 + x_2 = x_1 \Rightarrow x_1 = -x_2$$

$$x_1 + 2x_2 = x_2 \Rightarrow x_1 = -x_2$$

Hence, a possible eigenvector corresponding to the eigenvalue $\lambda_1 = 1$ is:

$$\boldsymbol{x_1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

The normalized eigenvector is: ✓

$$\boldsymbol{x_1} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$$

To find the eigenvector corresponding to eigenvalue $\lambda_2 = 3$, we substitute $\lambda_2 = 3$ to $A\boldsymbol{x} = \lambda\boldsymbol{x}$,

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 3 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Which is equivalence to:

$$2x_1 + x_2 = 3x_1 \Rightarrow x_1 = x_2$$

$$x_1 + 2x_2 = 3x_2 \Rightarrow x_1 = x_2$$

Hence, a possible eigenvector corresponding to the eigenvalue $\lambda_2 = 3$ is:

$$\boldsymbol{x_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

The normalized eigenvector is: ✓

$$\boldsymbol{x_2} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

Since matrix A is symmetric, according to the eigendecomposition result, we have,

$$\boldsymbol{A} = \boldsymbol{Q\Lambda Q^T}$$

i.e.,

$$(\vec{x}_1, \vec{x}_2) \quad \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \quad (\vec{x}_1, \vec{x}_2)^T$$

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

- ***Singular value decomposition:*** 奇异值分解
  - The ***singular value decomposition*** (SVD) is a factorization of a real or complex matrix. It is the generalization of the eigendecomposition of a squared matrix to any $m \times n$ matrix.
  - Specifically, every matrix $A \in \mathcal{R}^{m \times n}$ can be decomposed as:

$$\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$$

Where,

- $\boldsymbol{U} \in \mathcal{R}^{m \times m}$ and $\boldsymbol{V} \in \mathcal{R}^{n \times n}$ are orthogonal matrices and $\boldsymbol{\Sigma} \in \mathcal{R}^{m \times n}$ is a diagonal matrix with non-negative real numbers on the diagonal;
- 对角线元素
- The diagonal entries σ_i of $\boldsymbol{\Sigma}$ are the singular values of $\boldsymbol{A}$;
- The columns of $\boldsymbol{U}$ are called the left-singular vectors of $\boldsymbol{A}$;
- The columns of $\boldsymbol{V}$ are called the right-singular vectors of $\boldsymbol{A}$;

- Observe that the SVD factors provide eigendecompostions for $\boldsymbol{A^T A}$ *and* $\boldsymbol{AA^T}$:

$$\boldsymbol{A^T A = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T}$$
$$\boldsymbol{AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma^T \Sigma U^T}$$

Thus,

- The columns of $\boldsymbol{V}$ (***the right-singular vectors*** of $\boldsymbol{A}$) are eigenvectors of $\boldsymbol{A^T A}$
- The columns of $\boldsymbol{U}$ (***the left-singular vectors*** of $\boldsymbol{A}$) are eigenvectors of $\boldsymbol{AA^T}$
- The singular values σ_i of $\boldsymbol{A}$ are the square roots of the eigenvalues of $\boldsymbol{A^T A}$ (or equivalently, of $\boldsymbol{AA^T}$)

平方根

- ***Example: Eigendecomposition of the following matrix (linear algebra package in Numpy)***

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & -2 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.9258 & 0.1892 & -0.5066 \\ -0.3304 & 0.7660 & 0.3506 \\ -0.1834 & -0.6134 & 0.7877 \end{bmatrix} \begin{bmatrix} 0.8019 & 0 & 0 \\ 0 & -2.2470 & 0 \\ 0 & 0 & -0.5550 \end{bmatrix} \begin{bmatrix} 1.3182 & 0.2611 & 0.7315 \\ 0.3155 & 1.0246 & -0.2530 \\ 0.5529 & 0.8598 & 1.2425 \end{bmatrix}$$

- This matrix is not symmetric. The positive definiteness is not defined. The eigendecomposition is in the form:

$$\boldsymbol{A = Q\Lambda Q^{-1}}$$

- ***Quadratic form:*** 二次形
  - Let $\boldsymbol{A} \in \mathcal{R}^{n\times n}$ be a ***symmetric*** matrix. The expression $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ is called a *quadratic form*
  - ***For example,***

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}$$

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a_{11}x_1^{\textcircled{2}} + 2a_{12}x_1x_2 + a_{22}x_2^{\textcircled{2}}$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + 2a_{12}x_1x_2 + 2a_{23}x_2x_3 + 2a_{13}x_1x_3$$

  - When $\boldsymbol{A} = \boldsymbol{I}$, these two quadratic forms become $\boldsymbol{x}^T\boldsymbol{x} = \|\boldsymbol{x}\|_2^2$

- ***Positive (semi-)definite matrices:***

对称 半正定 表明

- A symmetric matrix $\boldsymbol{A}$ is ***positive semi-definite*** if for all $\boldsymbol{x} \in \mathcal{R}^n$, $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \geq 0$. denoted as $\boldsymbol{A} \succcurlyeq 0$.

正定

- A symmetric matrix $\boldsymbol{A}$ is ***positive definite*** if for all nonzero $\boldsymbol{x} \in \mathcal{R}^n$, $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} > 0$. denoted as $\boldsymbol{A} \succ 0$.

命题

- ***Proposition:*** A symmetric matrix is ***positive semi-definite*** *if and only if* all of its eigenvalues are nonnegative, and ***positive definite*** *if and only if* all of its eigenvalues are positive.

Hf	半正定	凸函数
	正定	严格凸函数
	半负定	凹函数
	负定	严格凹函数

✓ Optimization

2024/9/3

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

Subject to: $g_i(\boldsymbol{x}) \leq 0, i = 1,2, \dots, m$

$$h_j(\boldsymbol{x}) = 0, j = 1,2, \dots, p$$

- ✓ Suppose $f(\boldsymbol{x}): \mathcal{R}^n \to \mathcal{R}$. ***Optimization*** is about finding ***extrema*** (minima or maxima) ($\boldsymbol{x}^*$). It is necessary to consider the set of inputs over which we are optimizing. This set $\boldsymbol{\mathcal{X}} \subseteq \mathcal{R}^n$ is called the ***feasible set***.
- If $\boldsymbol{\mathcal{X}}$ is the entire domain of the function being optimized, the problem is ***unconstrained*** optimization. Otherwise, the problem is ***constrained*** and maybe much harder to solve, depending on the nature of the feasible set.

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

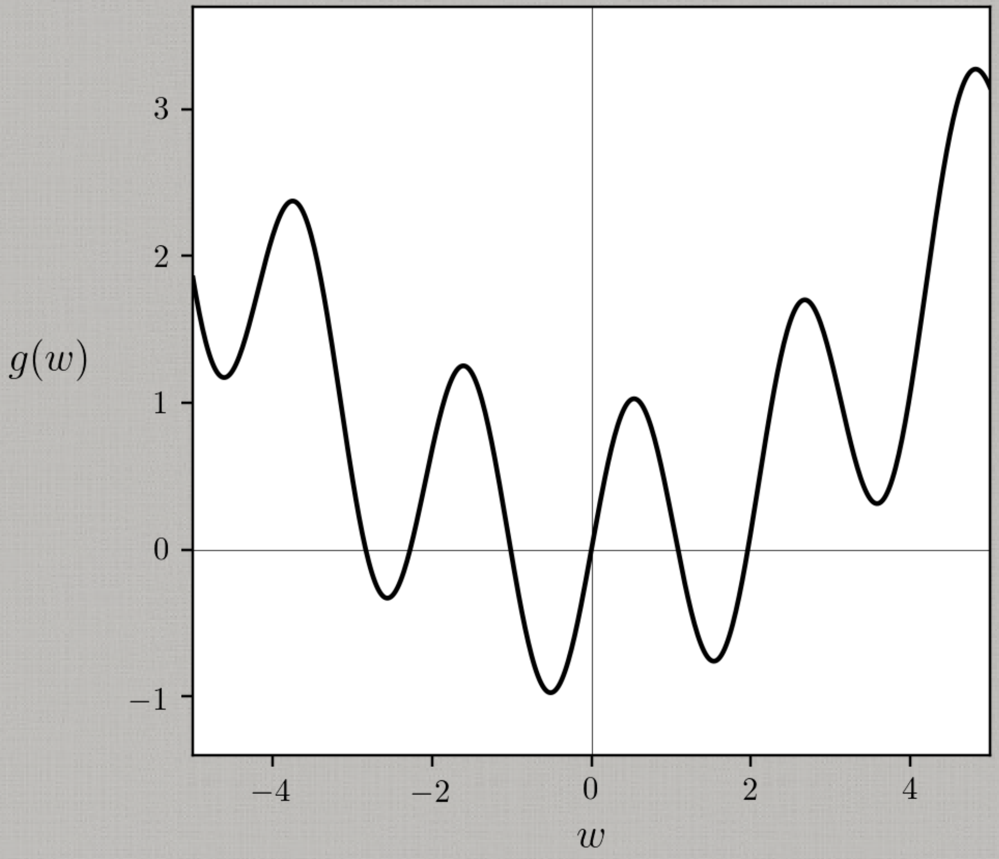
$$\text{Subject to: } g_i(\boldsymbol{x}) \leq 0, i = 1,2,\ldots,m$$

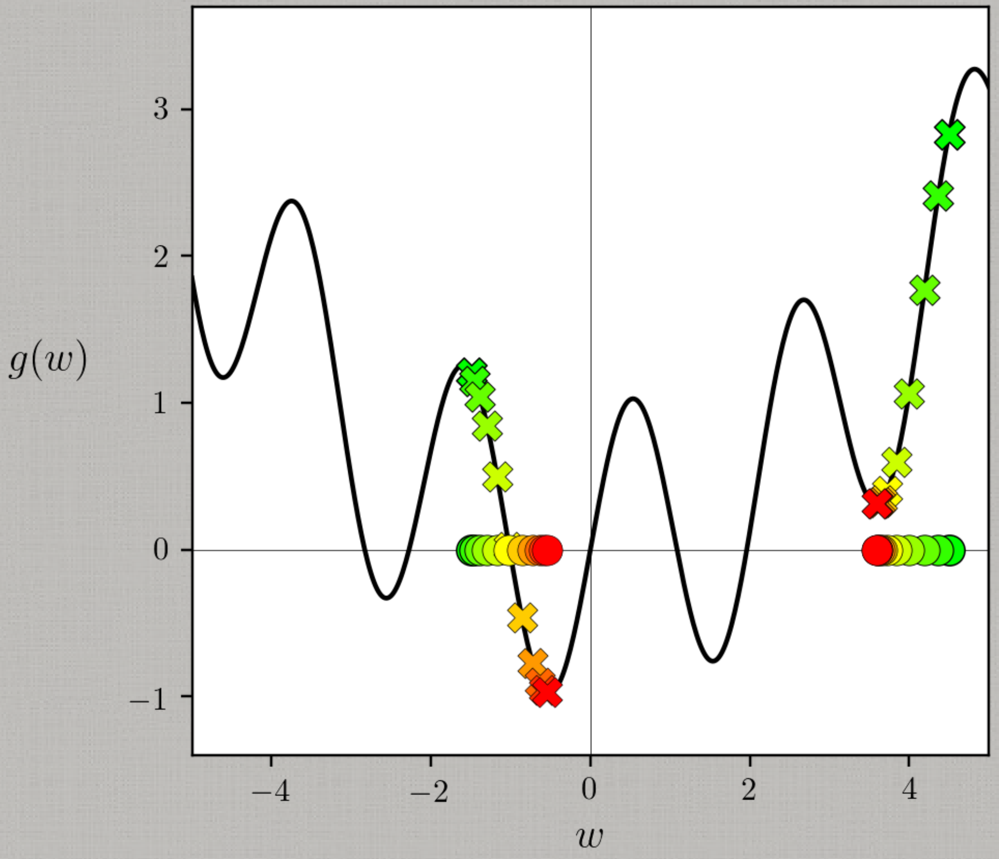
$$h_j(\boldsymbol{x}) = 0, j = 1,2,\ldots,p$$

- ✓ A point $\boldsymbol{x}$ is said to be a ***local minimum*** (resp. local maximum) of f in $\boldsymbol{\mathcal{X}}$ if $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ (resp. $f(\boldsymbol{x}) \geq f(\boldsymbol{y})$) for all $\boldsymbol{y}$ in some neighborhood $\boldsymbol{\mathcal{N}} \subseteq \boldsymbol{\mathcal{X}}$ about $\boldsymbol{x}$. Furthermore, if $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ for all $\boldsymbol{y} \in \boldsymbol{\mathcal{X}}$, then $\boldsymbol{x}$ is a ***global minimum*** (similarly for global maximum).

- Observe that maximizing a function f is equivalent to minimizing $-f$, so optimization problems are typically phrased in terms of ***minimizing*** without loss of generality.

$$g(w) = \sin(3w) + 0.1w^2$$





- ***Gradients:*** generalize derivatives to scalar functions of several variables.
  - The ***gradient*** of $f:\mathcal{R}^n \to \mathcal{R}$ denoted ∇f, is given by

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

∇ is called “Del” or “Nabla”

  - The gradient has the following very important property: $\nabla f(\boldsymbol{x})$ points in the direction of steepest ***ascent*** of f from $\boldsymbol{x}$.

上升

- ***Gradients:*** generalize derivatives to scalar functions of several variables.
  - The ***gradient*** of $f : \mathcal{R}^n \to \mathcal{R}$ denoted ∇f, is given by

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

- ***Example:*** Let $f(x_1, x_2) = 5e^{2x_1} + 10x_2^2$. Find $\nabla f(0,1)$.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 10e^{2x_1} \\ 20x_2 \end{bmatrix} \Rightarrow \nabla f(0,1) = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

- ***The Hessian:***
  - The ***Hessian*** of $f: \mathcal{R}^n \to \mathcal{R}$ is a matrix of second order partial derivatives:

$$Hf \quad \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}, \qquad [\nabla^2 f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

  - When $n = 2, 3$, we have:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}, \qquad \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix}$$

- ***The Hessian:***

  - The ***Hessian*** of $f: \mathcal{R}^n \to \mathcal{R}$ is a matrix of second order partial derivatives:

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}, \qquad [\nabla^2 f]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

  - ***Example:*** Let $f(x_1, x_2) = 5x_2 e^{2x_1} + 10x_2^2$. Find $\nabla^2 f$

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 20x_2 e^{2x_1} & 10e^{2x_1} \\ 10e^{2x_1} & 20 \end{bmatrix}$$

- **Proposition:** if $\boldsymbol{x}^*$ is a local *extrema* (*minimum or maximum*) of f and f is continuously differentiable in a neighborhood of $\boldsymbol{x}^*$, then $\nabla f(\boldsymbol{x}^*) = 0$

- **Proposition:** if $\boldsymbol{x}^*$ is a local *minimum* of f and f is twice continuously differentiable in a neighborhood of $\boldsymbol{x}^*$, then $\nabla^2 f(\boldsymbol{x}^*)$ is ***positive semi-definite***. $x^T \nabla^2 f(x^*) x \geq 0$. $\nabla^2 f(x^*) \geq 0$

- **Proposition:** Suppose f is twice continuously differentiable with $\nabla^2 f$ positive semi-definite in a neighborhood of $\boldsymbol{x}^*$, and that $\nabla f(\boldsymbol{x}^*) = 0$. Then $\boldsymbol{x}^*$ is a local minimum of f. Furthermore, if $\nabla^2 f(\boldsymbol{x}^*)$ is ***positive definite***, then $\boldsymbol{x}^*$ is a ***strict*** local minimum.

$x^T \nabla^2 f(x^*) x > 0$, $\nabla^2 f(x^*) > 0$ 正定的

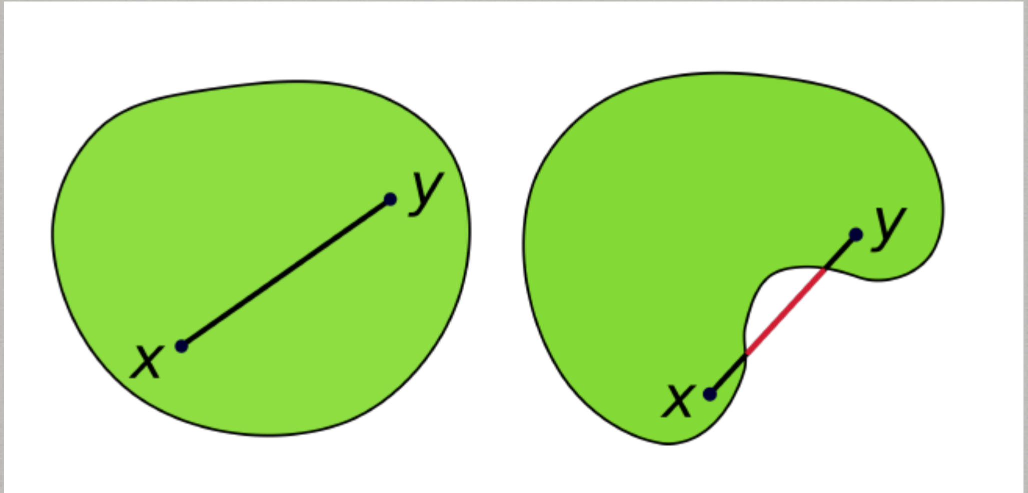
convex 凸 concave 凹

Convexity: Convexity of a function related closely to its minima: do they exist, are they unique?

Convex Sets: 凸集

A set $\boldsymbol{Z} \subseteq \mathcal{R}^{\boldsymbol{d}}$ is convex if $[t\boldsymbol{x} + (1-t)\boldsymbol{y}] \in \boldsymbol{Z}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{Z}$ and all $t \in [0,1]$.

Geometrically, this means that all the points on the line segment between any two points in $\boldsymbol{\mathcal{X}}$ are also in $\boldsymbol{\mathcal{X}}$.



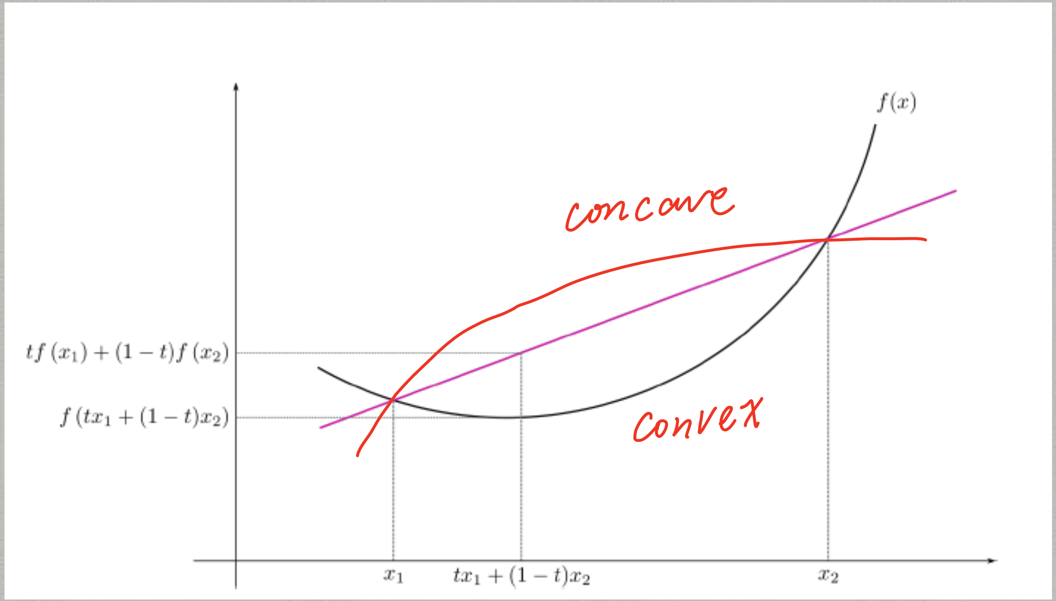
(a) A convex set (b) A non-convex set

Convex functions: 凸函数

- A function $f: \mathcal{R}^d \to \mathcal{R}$ is ***convex*** if $f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$ for all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \underset{\text{domain}}{dom}(f)$ and all $t \in [0,1]$.
- We say that f is ***strictly convex*** if the inequality holds strictly for all $t \in [0,1]$ and $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$.
- Geometrically, ***convexity means that the line segment between two points on the graph of f lies on or above the graph itself.***

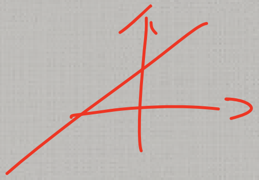
几何法

$f''(x) > 0,$

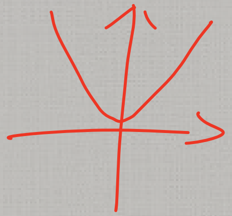


- Examples of ***convex functions***:

$$f(x) = ax + b$$

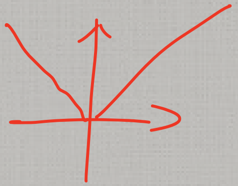


$$f(x) = x^2$$

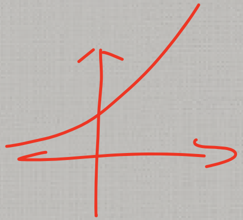


$$f(x) = |x|$$

$f'(x)\nearrow$

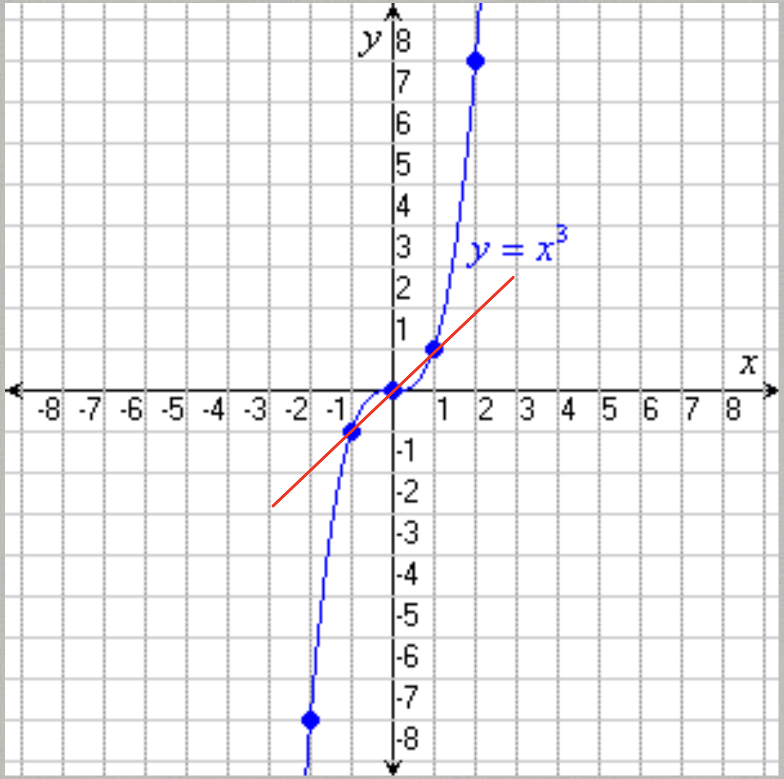


$$f(x) = e^x$$



- Every norm is a convex function

- How about x^3? x^3 is neither convex nor concave



$f'(x) \downarrow \to 0$

$f'(x) \uparrow \to \infty$

Consequences of convexity:

- **Proposition:** Let $\mathcal{X}$ be a convex set. If f is convex, then any ***local minimum*** of f in $\mathcal{X}$ is also a ***global minimum***.

- **Proposition:** Let $\mathcal{X}$ be a convex set. If f is strictly convex, then there exists at most one local minimum of f in $\mathcal{X}$. Consequently, if it exists, it is the ***unique global minimum*** of f in $\mathcal{X}$.

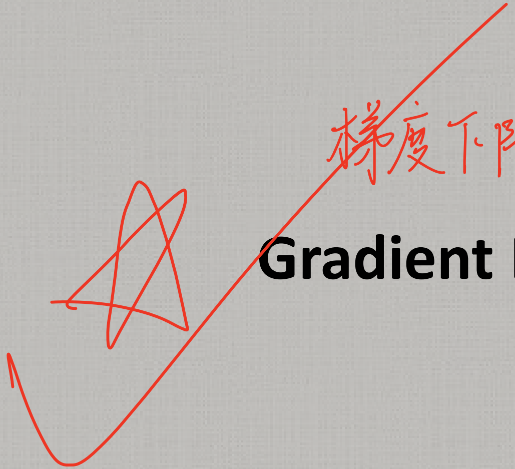
- **Convex functions:**
  - **Proposition:** Suppose f is twice differentiable. Then,
    - f is convex *if and only if* $\nabla^2 f(\boldsymbol{x}) \succcurlyeq 0$ for all $\boldsymbol{x} \in dom(f)$. $f''(x) \geq 0$
    - If $\nabla^2 f(\boldsymbol{x}) \succ 0$ for all $\boldsymbol{x} \in dom(f)$, then f is strictly convex. $f''(x) > 0$
  - **Proposition:** norms are convex

- **Proposition:** if f is convex and $\alpha \geq 0$, then αf is convex.
- **Proposition:** if $f \; and \; g$ are convex, then $f + g$ is convex.
- **Proposition:** if $f_1, \dots, f_n$ are convex and $\alpha_1, \dots, \alpha_n \geq 0$, then, $\sum_{i=1}^{n} \alpha_i f_i$ is convex.
- **Proposition:** if f is convex, then $g(\boldsymbol{x}) \equiv f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$ is convex for any appropriately-sized $\boldsymbol{A} \; and \; \boldsymbol{b}$

$$\|\boldsymbol{x}\|_2^2? \qquad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2?$$

- **Proposition:** if $f \; and \; g$ are convex, then $h(\boldsymbol{x}) \equiv max\{f(\boldsymbol{x}), g(\boldsymbol{x})\}$ is convex

梯度下降



Gradient Descent Search Method to Find Local Minima

- Given an objective function $g(\boldsymbol{w})$, to find the value of the parameter $\boldsymbol{w}$ that minimizes the objective function, we start from a initial point $\boldsymbol{w}^0$, the local optimization method refine this initial point sequentially, pulling it downhill towards points that are lower and lower on the objective function, eventually reaching a minimum.

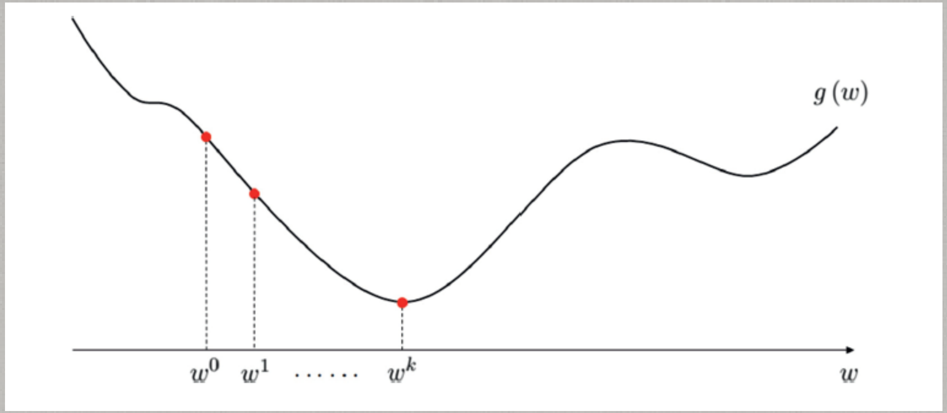
- This process yields a sequence of $K+1$ points (***minimizing sequence***):

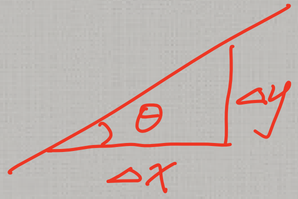
$$\boldsymbol{w}^0, \boldsymbol{w}^1, \dots, \boldsymbol{w}^K$$

So that,

$$g(\boldsymbol{w}^0) > g(\boldsymbol{w}^1) > \cdots > g(\boldsymbol{w}^K)$$

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, g(\boldsymbol{w})$$





$y = kx$

$k = \frac{\Delta y}{\Delta x} = \tan\theta$

Gradient descent search in one dimension

- According to ***Taylor series expansion*** of $g(w)$, we have,

$(w+\Delta w - w)$

$$g(w+\Delta w) \approx g(w) + g'(w)\Delta w$$

代替

- Substitute Δw with $-\eta g'(w)$, where $\eta > 0$ is a constant (called ***learning rate***), we have,

$$g\big(w-\eta g'(w)\big) \approx g(w) - \eta\big(g'(w)\big)^2$$

>0

- Hence, we have, when $g'(w) \neq 0$,

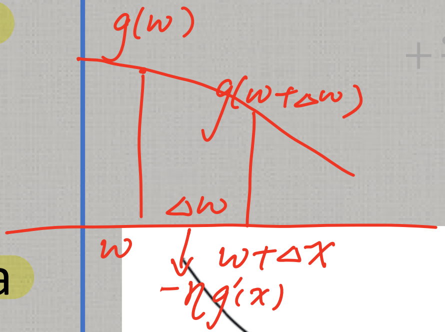
$$g\big(w-\eta g'(w)\big) < g(w)$$

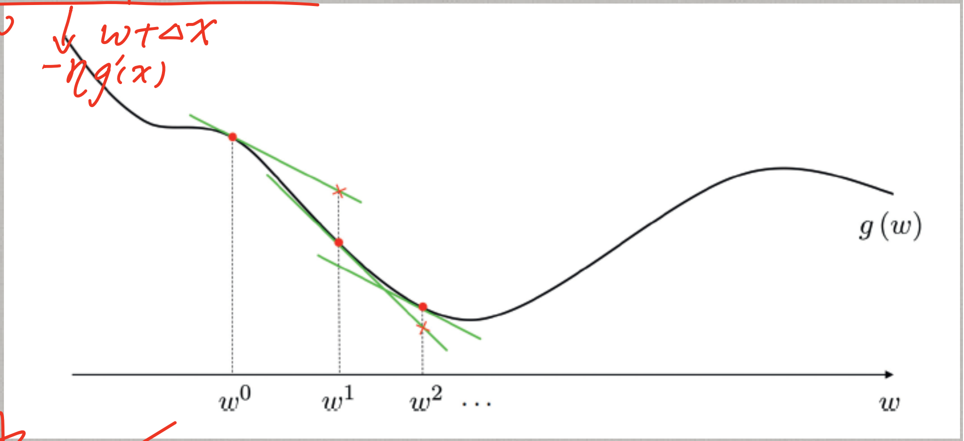
- i.e., If w is updated as: $w' = w^0 - \eta g'(w^0)$

$$w^k = w^{k-1} - \eta g'\big(w^{k-1}\big)$$

the minimizing sequence can be generated.

$$g(w+\Delta w) = g(w) + g'(w)\Delta w + \frac{g''(w)}{2!}(\Delta w)^2 + \frac{g'''(w)}{3!}(\Delta w)^3 + \cdots$$





$$f(x) = \frac{f(x_0)}{0!} + \frac{f'(x_0)}{1!}(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + R_n(x)$$

$\Downarrow$ Peano

$o[(x-x_0)^n]$

☆ ✓ $f(x) \approx f(x_0) + \nabla f(x_0)(x-x_0) + \frac{1}{2!}(x-x_0)^T Hf(x_0)(x-x_0) + \cdots$

Gradient descent search in multi-dimension case:

- In higher dimensional case, the Taylor series of the objective function becomes:

$$g(\boldsymbol{w}+\Delta\boldsymbol{w}) \approx g(\boldsymbol{w}) + \nabla g(\boldsymbol{w})\Delta\boldsymbol{w}$$

Where,

$$\nabla g(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \vdots \\ \frac{\partial g}{\partial w_n} \end{bmatrix}$$

is the gradient of the objective function.

$\eta > 0$

substitute Δw with $-\eta \nabla g(w)$

$g(w - \eta \nabla g(w)) \approx g(w) - \eta \underbrace{|\nabla g(w)|^2}_{>0}$

$g(w - \eta \nabla g(w)) < g(w)$

$w^1 = w^0 - \eta \nabla g(w^0)$

- The gradient descent update of the variable becomes:

$$\boldsymbol{w}^k = \boldsymbol{w}^{k-1} - \eta\nabla g(\boldsymbol{w}^{k-1})$$

Example of Gradient descent Search in one-dimension:

- Consider the following objective function:

$$g(w) = (w)^2 + 2w + 2$$

Find the value of w that minimizes $g(w)$.

- Consider the following objective function:

$$g(w) = (w)^2 + 2w + 2$$

To find the value of w that minimizes $g(w)$, we take the derivative of $g(w)$ and let it equal to zero:

$g'(w) = 0$

$$g'(w) = 2w + 2 = 0 \Rightarrow w = -1$$

$$g''(w) = 2 > 0$$

Hence, we have, $g(w)$ is minimized at $w^* = -1$ and $g(w^*) = 1$

- Consider the following objective function:

$$g(w) = (w)^2 + 2w + 2$$

Let's solve this problem using the gradient descent search method with:

$$w^0 = 0, \eta = 0.1$$

Where, $\overset{w^k}{w^0} = 0$ is the value of w at step 0 ($k = 0$), i.e., the initial value of w.

$$g(w+\Delta w) \approx g(w) + g'(w)\Delta w$$

substitute Δw with $-\eta g'(w)$, $\eta > 0$

$$g(w - \eta g'(w)) \approx g(w) - \eta \underbrace{(g'(w))^2}_{>0}$$

$$g(w - \eta g'(w)) < g(w)$$

$$w^k = w^{k-1} - \eta g'(w^{k-1}) \longleftarrow w^1 = w^0 - \eta g'(w^0)$$

In 1D case, the gradient becomes the derivative of the objective function:

$$g'(w) = 2w + 2$$

- $(k = 1)$ At $w^0 = 0$, we have,

$w^{0^2} + 2w^0 + 2$

$$g'(w^0) = 2w^0 + 2 = 2; \quad g(w^0) = (w^0 + 1)^2 + 1 = 2$$

$g'(0) = 2$

Update w:

$$w^1 = w^0 - \eta g'(w^0) = 0 - 0.1 \times 2 = -0.2$$

$0 \longrightarrow -0.2$

2

- $(k = 2)$ At $w^1 = -0.2$, we have,

$$g'(w^1) = 2w^1 + 2 = 1.6$$

$$g(w^1) = (-0.2 + 1)^2 + 1 = 1.64$$

Update w:

$$w^2 = w^1 - \eta g'(w^1) = -0.2 - 0.1 \times 1.6 = -0.36$$

$-0.2 \longrightarrow -0.36$

1.64

- $(k = 3)$ At $w^2 = -0.36$, we have,

$$g'(w^2) = 2w^2 + 2 = 1.28$$

$$g(w^2) = (w^2 + 1)^2 + 1 = 1.41$$

Update w:

$$w^3 = w^3 - \eta g'(w^2) = -0.36 - 0.1 \times 1.28 = -0.49$$

$-0.36 \longrightarrow -0.49$

1.41

- $(k = 4)$ At $w^3 = -0.49$, we have,

$$g'(w^3) = 2w^3 + 2 = 1.02$$

$$g(w^3) = (w^3 + 1)^2 + 1 = 1.26$$

Update w:

$$w^4 = w^3 - \eta g'(w^3) = -0.49 - 0.1 \times 1.02 = -0.59$$

$-0.49 \longrightarrow -0.59$

1.26

- $(k = 5)$ At $w^4 = -0.59$, we have,

$$g'(w^4) = 2w^4 + 2 = 0.82$$

$$g(w^4) = (w^4 + 1)^2 + 1 = 1.17$$

Update w:

$$w^5 = w^4 - \eta g'(w^4) = -0.59 - 0.1 \times 0.82 = -0.67$$

$-0.59 \longrightarrow -0.67$

1.17

- $(k = 6)$ At $w^5 = -0.67$, we have,

$$g'(w^5) = 2w^5 + 2 = 0.66$$

$$g(w^5) = (w^5 + 1)^2 + 1 = 1.11$$

Update w:

$$w^6 = w^5 - \eta g'(w^5) = -0.67 - 0.1 \times 0.66 = -0.74$$

$-0.67 \longrightarrow -0.74$

1.11

- ($k = 7$) At $w^6 = -0.74$, we have,

$$g'(w^6) = 2w^6 + 2 = 0.52$$

$$g(w^6) = (w^6 + 1)^2 + 1 = 1.07$$

Update w:

$$w^7 = w^6 - \eta g'(w^6) = -0.74 - 0.1 \times 0.52 = -0.79$$

$-0.74 \longrightarrow -0.79$

1.07

- ($k = 8$) At $w^7 = -0.79$, we have,

$$g'(w^7) = 2w^7 + 2 = 0.42$$

$$g(w^7) = (w^7 + 1)^2 + 1 = 1.04$$

Update w:

$$w^8 = w^7 - \eta g'(w^7) = -0.79 - 0.1 \times 0.42 = -0.83$$

$-0.79 \longrightarrow -0.83$

1.04

- $(k = 8)$ At $w^8 = -0.83$, we have,

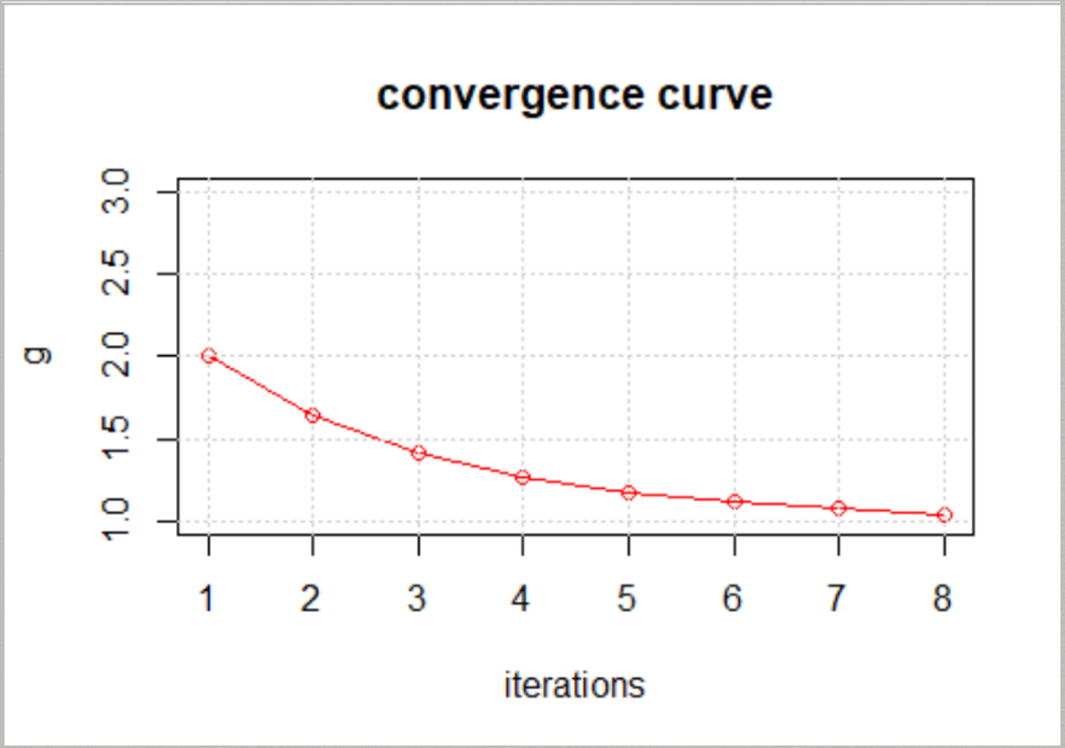
$$g'(w^8) = 2w^8 + 2 = 0.34$$

$$g(w^8) = (w^8 + 1)^2 + 1 = 1.03$$

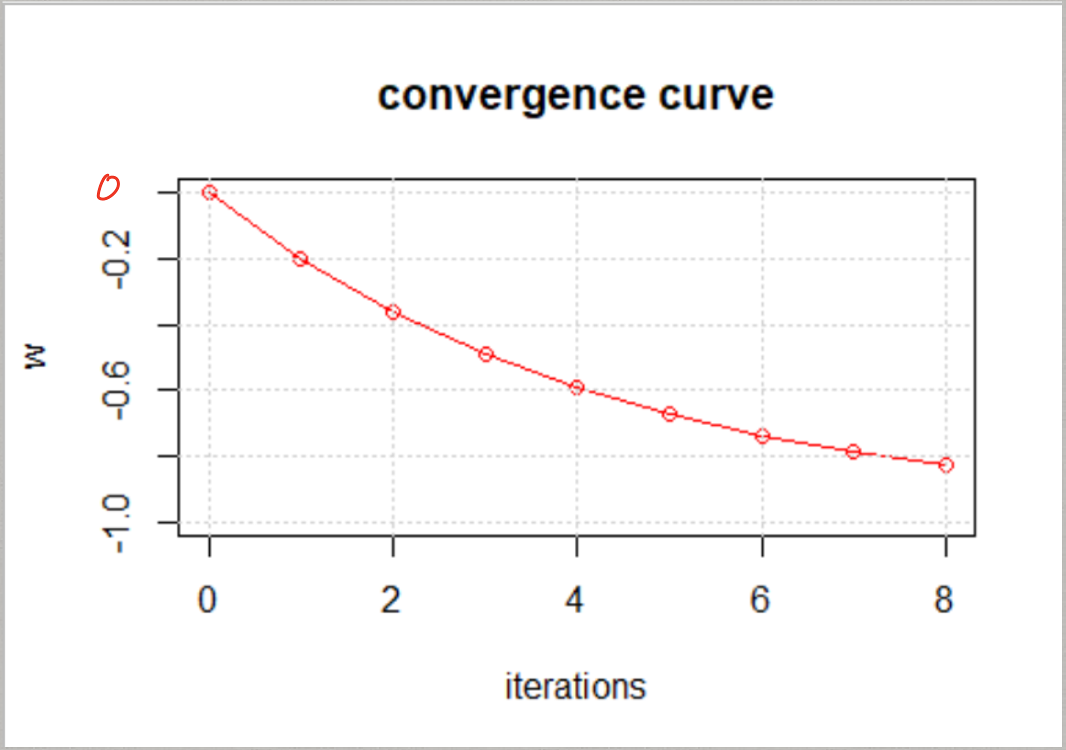
$$\vdots$$

$$\begin{array}{l} -0.83 \longrightarrow \cdots \\ 1.03 \end{array}$$

收敛曲线



convergence curve
g
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5
6
7
8
iterations



convergence curve
w
0
-0.2
-0.6
-1.0
0
2
4
6
8
iterations

✓ Example of Gradient descent Search in 2D:

目标函数

- Consider the following objective function:

$$g(\boldsymbol{w}) = 0.5w_1^2 + w_2^2, \qquad \boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Find the value of $\boldsymbol{w}$ that minimizes $g(\boldsymbol{w})$.

$g(\vec{w} + \Delta\vec{w}) \approx g(\vec{w}) + \nabla g(\vec{w}) \Delta\vec{w}$

substitute $\Delta\vec{w}$ with $-\eta \nabla g(\vec{w})$, $\eta > 0$

$g(\vec{w} - \eta \nabla g(\vec{w})) \approx g(\vec{w}) - \eta \underbrace{|\nabla g(\vec{w})|^2}_{>0}$

$g(\vec{w} - \eta \nabla g(\vec{w})) < g(\vec{w})$

$w' = w^0 - \eta \nabla g(w^0)$

- It is easy to find out that the objective function is minimized at:

$$\mathbf{w}^* = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ and \ g(\mathbf{w}^*) = 0$$

$\nabla g(w) = 0 \Rightarrow w = \binom{0}{0}$

- The gradient of the objective function is:

$$\nabla g(\mathbf{w}) = \begin{bmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \end{bmatrix} = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix}$$

- The gradient descent update of the variable becomes:

$$\mathbf{w}^k = \mathbf{w}^{k-1} - \eta \nabla g\left(\mathbf{w}^{k-1}\right)$$

Let's carry out the gradient descent search with:

$$\boldsymbol{w}^0 = \begin{bmatrix}1\\1\end{bmatrix} \; and \; \eta = 0.2$$

- $(k = 1)$, we have,

$$\nabla g(\boldsymbol{w}^0) = \begin{bmatrix}w_1\\2w_2\end{bmatrix} = \begin{bmatrix}1\\2\end{bmatrix}, \qquad g(\boldsymbol{w}^0) = 1.5$$

$w^k = w^{k-1} - \eta \nabla g(w^{k-1})$

$$\boldsymbol{w}^1 = \boldsymbol{w}^0 - \eta\nabla g(\boldsymbol{w}^0) = \begin{bmatrix}1\\1\end{bmatrix} - 0.2 \times \begin{bmatrix}1\\2\end{bmatrix} = \begin{bmatrix}0.8\\0.6\end{bmatrix}$$

$$\begin{pmatrix}1\\1\end{pmatrix} \longrightarrow \begin{pmatrix}0.8\\0.6\end{pmatrix}$$

1.5

- $(k = 2)$, at $\boldsymbol{w}^1 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}$ we have,

$$\begin{pmatrix} 0.8 \\ 0.6 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.64 \\ 0.36 \end{pmatrix}$$

0.68

$$\nabla g(\boldsymbol{w}^1) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 1.2 \end{bmatrix}, \qquad g(\boldsymbol{w}^1) = 0.68$$

$$\boldsymbol{w}^2 = \boldsymbol{w}^1 - \eta \nabla g(\boldsymbol{w}^1) = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.8 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0.64 \\ 0.36 \end{bmatrix}$$

- $(k = 3)$, at $\boldsymbol{w}^2 = \begin{bmatrix} 0.64 \\ 0.36 \end{bmatrix}$ we have,

$$\begin{pmatrix} 0.64 \\ 0.36 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.512 \\ 0.216 \end{pmatrix}$$

0.3344

$$\nabla g(\boldsymbol{w}^2) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.64 \\ 0.72 \end{bmatrix}, \qquad g(\boldsymbol{w}^2) = 0.3344$$

$$\boldsymbol{w}^3 = \boldsymbol{w}^2 - \eta \nabla g(\boldsymbol{w}^2) = \begin{bmatrix} 0.64 \\ 0.36 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.64 \\ 0.72 \end{bmatrix} = \begin{bmatrix} 0.512 \\ 0.216 \end{bmatrix}$$

- $(k = 4)$, at $\boldsymbol{w}^3 = \begin{bmatrix} 0.512 \\ 0.216 \end{bmatrix}$, we have,

$\begin{pmatrix} 0.512 \\ 0.216 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.41 \\ 0.13 \end{pmatrix}$

0.178

$$\nabla g(\boldsymbol{w}^3) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.512 \\ 0.432 \end{bmatrix}, \qquad g(\boldsymbol{w}^3) = 0.178$$

$$\boldsymbol{w}^4 = \boldsymbol{w}^3 - \eta \nabla g(\boldsymbol{w}^3) = \begin{bmatrix} 0.512 \\ 0.216 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.512 \\ 0.432 \end{bmatrix} = \begin{bmatrix} 0.41 \\ 0.13 \end{bmatrix}$$

$\begin{pmatrix} 0.41 \\ 0.13 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.328 \\ 0.078 \end{pmatrix}$

0.101

- $(k = 5)$, at $\boldsymbol{w}^4 = \begin{bmatrix} 0.41 \\ 0.13 \end{bmatrix}$, we have,

$$\nabla g(\boldsymbol{w}^4) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.41 \\ 0.26 \end{bmatrix}, \qquad g(\boldsymbol{w}^4) = 0.101$$

$$\boldsymbol{w}^5 = \boldsymbol{w}^4 - \eta \nabla g(\boldsymbol{w}^4) = \begin{bmatrix} 0.41 \\ 0.13 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.41 \\ 0.26 \end{bmatrix} = \begin{bmatrix} 0.328 \\ 0.078 \end{bmatrix}$$

- $(k = 6)$, at $\boldsymbol{w}^5 = \begin{bmatrix} 0.328 \\ 0.078 \end{bmatrix}$, we have,

$$\begin{pmatrix} 0.328 \\ 0.078 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.262 \\ 0.047 \end{pmatrix}$$

0.06

$$\nabla g(\boldsymbol{w}^5) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.328 \\ 0.156 \end{bmatrix}, \qquad g(\boldsymbol{w}^5) = 0.06$$

$$\boldsymbol{w}^6 = \boldsymbol{w}^5 - \eta \nabla g(\boldsymbol{w}^5) = \begin{bmatrix} 0.328 \\ 0.078 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.328 \\ 0.156 \end{bmatrix} = \begin{bmatrix} 0.262 \\ 0.047 \end{bmatrix}$$

$$\begin{pmatrix} 0.262 \\ 0.047 \end{pmatrix} \longrightarrow \begin{pmatrix} 0.21 \\ 0.028 \end{pmatrix}$$

0.037

- $(k = 7)$, at $\boldsymbol{w}^6 = \begin{bmatrix} 0.262 \\ 0.047 \end{bmatrix}$, we have,

$$\nabla g(\boldsymbol{w}^6) = \begin{bmatrix} w_1 \\ 2w_2 \end{bmatrix} = \begin{bmatrix} 0.262 \\ 0.094 \end{bmatrix}, \qquad g(\boldsymbol{w}^6) = 0.037$$

$$\boldsymbol{w}^7 = \boldsymbol{w}^6 - \eta \nabla g(\boldsymbol{w}^6) = \begin{bmatrix} 0.262 \\ 0.047 \end{bmatrix} - 0.2 \times \begin{bmatrix} 0.262 \\ 0.094 \end{bmatrix} = \begin{bmatrix} 0.21 \\ 0.028 \end{bmatrix}$$

at $\boldsymbol{w}^7 = \begin{bmatrix} 0.21 \\ 0.028 \end{bmatrix}$, we have,

$$\begin{pmatrix} 0.21 \\ 0.028 \end{pmatrix} \longrightarrow \cdot\ \cdot$$

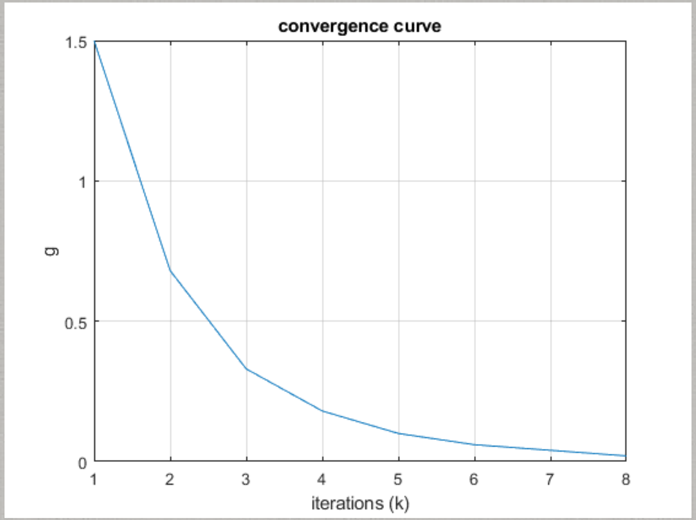
$$g(\boldsymbol{w}^7) = 0.023$$

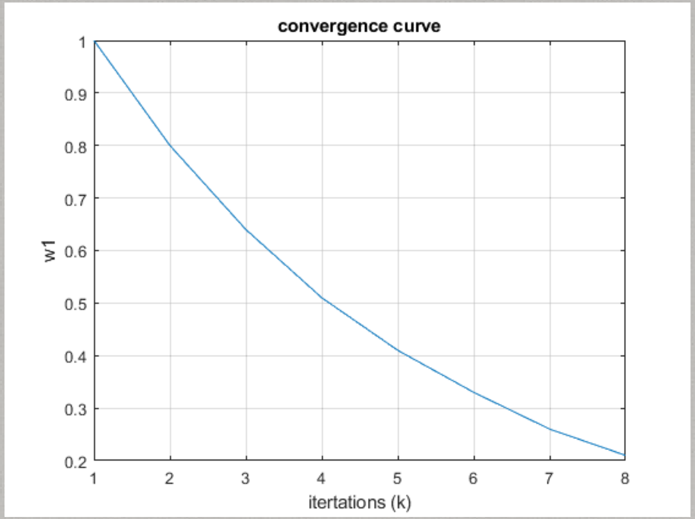
0.023

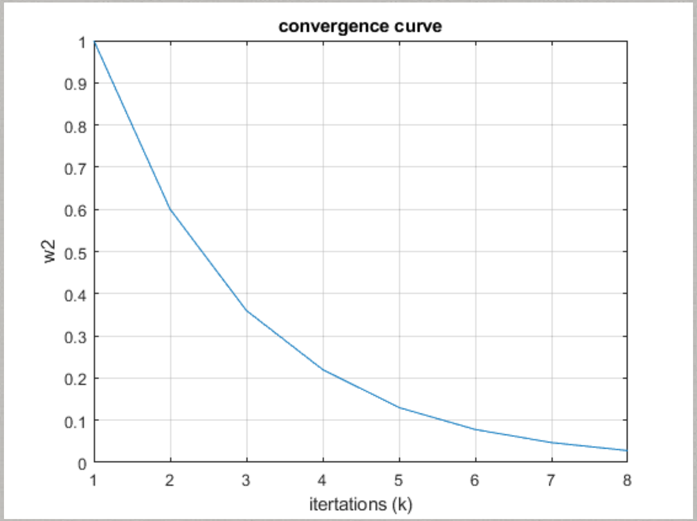
$$\vdots$$

when should I stop?

Set a threshold value







收敛准则 **Convergence Criterion of Gradient Descent Search Method**

停止

Technically, if the learning rate is chosen wisely, the algorithm will halt near stationary points of the objective function. Hence, we can come up with the following criterions to end the algorithm:

- When the magnitude of the gradient is sufficiently small, i.e., for any small number $\epsilon > 0$

$$\left\|\nabla g\left(\mathbf{w}^{k-1}\right)\right\|_2 \leq \epsilon$$

- When steps no longer make significant progress,

$$\left\|\mathbf{w}^{k} - \mathbf{w}^{k-1}\right\|_2 \leq \epsilon$$

- When corresponding evaluations no longer differ substantially, 本质上

$$\left|g\left(\mathbf{w}^{k}\right) - g\left(\mathbf{w}^{k-1}\right)\right| \leq \epsilon$$

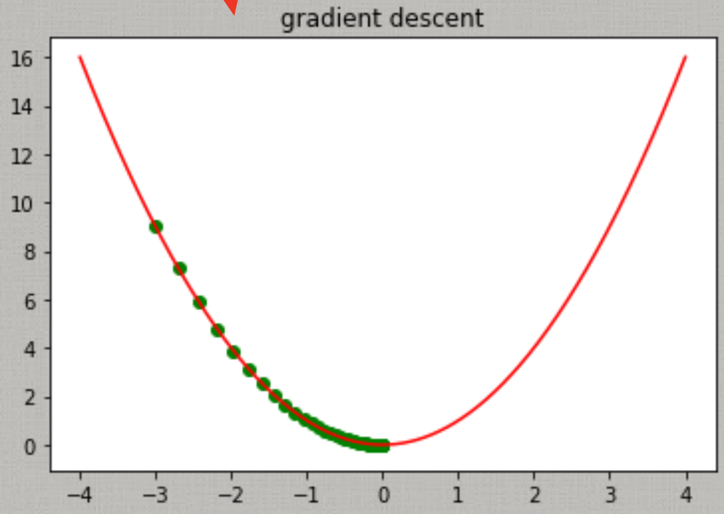
- Run the algorithm for a fixed number of maximum iterations.

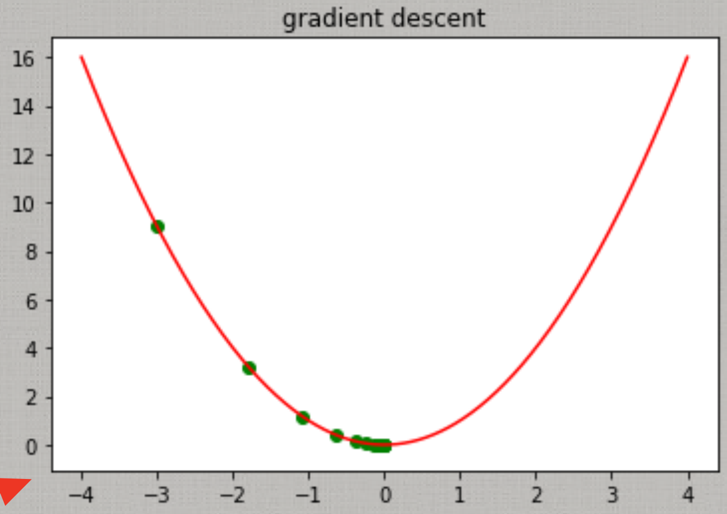
Learning rate for the Gradient Descent Search Method

- **Fixed value learning rate**

$$\eta = 10^{-n}, n = 1,2,3$$

0.1, 0.01, 0.001





- **Diminishing learning rate**

$k\uparrow \quad n\downarrow$

$$\eta = \frac{1}{k}, k \text{ is the number of iterations}$$

Learning rate for the Gradient Descent Search Method

- What is the consequence that the learning rate is too small?
- What is the consequence that the learning rate is too large?

Practice Example of Gradient descent Search in two-dimension:

- Consider the following objective function:

$$g(\boldsymbol{w}) = (w_1)^2 + 2w_1 + 2(w_2)^2 + 2 = (w_1 + 1)^2 + 2(w_2)^2 + 1$$

Use gradient descent search method to find the values of $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ that minimizes $g(\boldsymbol{w})$. You may assume different initial values of $\boldsymbol{w}$ and different learning rates.

$$\nabla g(\vec{w}) = \begin{pmatrix} \frac{\partial g}{\partial w_1} \\ \frac{\partial g}{\partial w_2} \end{pmatrix} = \begin{pmatrix} 2w_1 + 2 \\ 4w_2 \end{pmatrix} = 0 \implies \begin{cases} w_1 = -1 \\ w_2 = 0 \end{cases}$$

$$\vec{w}^* = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

$$g(\vec{w}^*) = g(-1, 0) = 1 - 2 + 0 + 2 = 1$$

Taylor Series of the objective function

$$g(\vec{w}+\Delta\vec{w}) \approx g(\vec{w}) + \nabla g(\vec{w})\Delta\vec{w}$$

substitute $\Delta\vec{w}$ with $-\eta\nabla g(\vec{w})$, $\eta>0$, constant

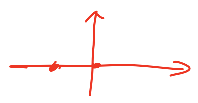
$$g(\vec{w}-\eta\nabla g(\vec{w})) \approx g(\vec{w}) - \eta\underbrace{\|\nabla g(\vec{w})\|_2^2}_{>0}$$

$$g(\vec{w}-\eta\nabla g(\vec{w})) < g(\vec{w})$$

$$\vec{w}^1 = \vec{w}^0 - \eta\nabla g(\vec{w}^0)$$

$$\vec{w}^k = \vec{w}^{k-1} - \eta\nabla g(\vec{w}^{k-1})$$

Let's carry out the gradient descent search with



$\vec{w}^0 = \begin{pmatrix}0\\0\end{pmatrix}$ and $\eta = 0.2$

$$\nabla g(\vec{w}) = \begin{pmatrix}2w_1+2\\4w_2\end{pmatrix}$$

$$g(w_1,w_2) = (w_1+1)^2+2w_2^2+1$$

$$g(\vec{w}^0) = (0+1)^2+0+1 = 2$$

$$\vec{w}^1 = \vec{w}^0 - \eta\nabla g(\vec{w}^0)$$

$$= \begin{pmatrix}0\\0\end{pmatrix} - 0.2\begin{pmatrix}2\\0\end{pmatrix} = \begin{pmatrix}-0.4\\0\end{pmatrix}$$

$$g(\vec{w}^1) = (-0.4+1)^2+0+1 = 1.36$$

$$\vec{w}^2 = \vec{w}^1 - \eta\nabla g(\vec{w}^1)$$

$$= \begin{pmatrix}-0.4\\0\end{pmatrix} - 0.2\begin{pmatrix}1.2\\0\end{pmatrix} = \begin{pmatrix}-0.4-0.24\\0\end{pmatrix} = \begin{pmatrix}-0.64\\0\end{pmatrix}$$

$$g(\vec{w}^2) = (-0.64+1)^2+0+1 = 1.1296$$

$$\vec{w}^3 = \vec{w}^2 - \eta\nabla g(\vec{w}^2)$$

$$= \begin{pmatrix}-0.64\\0\end{pmatrix} - 0.2\begin{pmatrix}-1.28+2\\0\end{pmatrix} = \begin{pmatrix}-0.64-0.144\\0\end{pmatrix} = \begin{pmatrix}-0.784\\0\end{pmatrix}$$

$$g(\vec{w}^3) = (-0.784+1)^2+0+1 = 1.046656$$

- - .