安徽理工大学
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

• 人工智能专业 学科基础教育必修模块

2025

Python与机器学习

Python and Machine Learning

Chapter 8: Linear Models

• Lecturer ： 孟亦凡

• E-mail：myf@aust.edu.cn

Chapter 8: Linear Models

CONTENTS

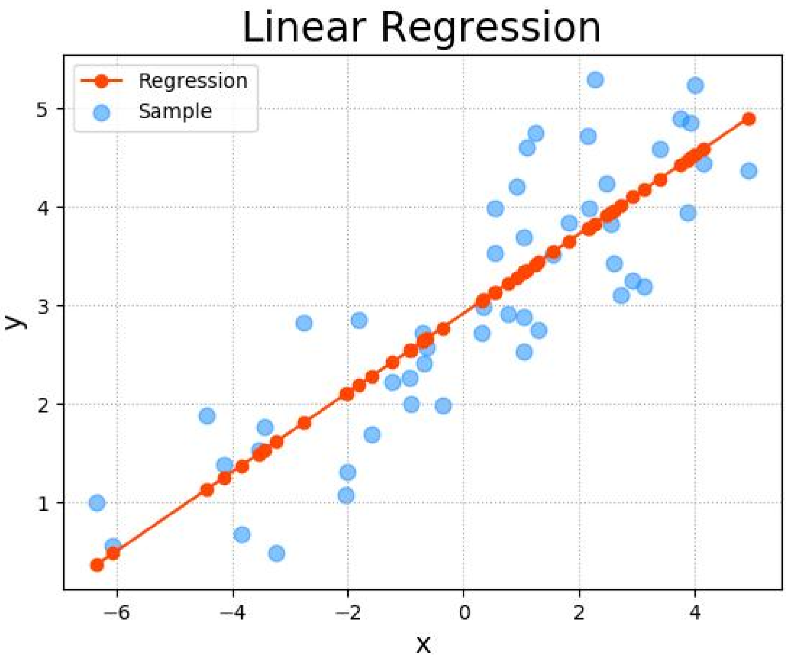
CONTENTS

8.1 Linear Regression
线性回归

线性回归是一种通过对属性进行线性组合来做出预测的模型。
Linear regression is a model that makes predictions through linear combination of attributes.

➢ What is Linear Regression?

- **Linear regression** is a model that makes predictions through linear combination of attributes.
- Goal: Find a straight line（直线）, plane（平面）, or higher-dimensional hyperplane（高维超平面） that minimizes the error between predicted and actual values.
- Core idea: Use the simplest linear relationship to approximate complex real-world patterns.（用最简单的线性关系逼近复杂世界的规律。）



➢ Basic form

- **General Form of Linear Models**

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}_1\boldsymbol{x}_1 + \boldsymbol{w}_2\boldsymbol{x}_2 + \cdots + \boldsymbol{w}_d\boldsymbol{x}_d + \boldsymbol{b}$$

$x = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_d)$is an example described by attributes, which $\boldsymbol{x}_i$ is the value of on the i-th attribute.

- **Vector form**

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b}$$

Where$w = (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots \boldsymbol{w}_d)$are the weights / 权重 (importance of each feature)

b is the bias / 偏置 (overall offset).

➢ Advantages of Linear Models

- **Simplicity | 形式简单**
  - **Easy to understand, implement, and compute.(易于理解、实现和计算)**
- **Interpretability | 可解释性强**
  - **The weight w_i directly shows the importance and direction of influence of feature x_i.**
- **Foundation | 基础性强**
  - **The cornerstone for many nonlinear models (e.g., neural networks).**

➢ A Practical Example

- Task: Predict if a watermelon is good.
  - Model: f(x) = 0.2 × color + 0.5 × stem（根蒂） + 0.3 × sound + 0.1

- Interpretation | 解读:
- Stem (0.5) has the largest weight → Most important feature.

- Sound (0.3) > Color (0.2) → More significant than color.

➢ Handling Discrete Features（处理离散特征）

- Features are not always numbers. How do we handle them?
- 1. Ordered Features (e.g., Size: Small, Medium, Large) | 有序特征（例如，尺寸：小、中、大）

- Method: Map to continuous values.（映射为连续值）
- Example: Small→0.0, Medium→0.5, Large→1.0

➢ Handling Discrete Features（处理离散特征）

- Features are not always numbers. How do we handle them?
- 2. Unordered Features (e.g., Type: A, B, C) | 无序特征（例如，品种： A, B, C)
- Method: One-Hot Encoding.（独热编码）
- Example: Type A → [1, 0, 0], Type B → [0, 1, 0], Type C → [0, 0, 1]

➢ The Goal of Linear Regression

- Given dataset $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$, each sample has single attribute.
- We want to find the best w and b so that:

$$f(x_i) = w\boldsymbol{x} + \boldsymbol{b}$$

$f(x_i)$ is as close as possible to y_i for all i.

linear regression（线性回归）：

The aim of the linear model is to learn a linear model that predicts the actual value output labels as accurately as possible.

➢ Parameter Estimation: Least Squares Method(参数估计：最小二乘法)

- **Core Idea: Minimize the Sum of Squared Errors (SSE).(最小化误差平方和 (SSE))**

$$
\begin{aligned}
(w^*, b^*) &= \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} \left(f\left(x_i\right) - y_i\right)^2 \\
&= \underset{(w,b)}{\arg\min} \sum_{i=1}^{m} \left(y_i - w x_i - b\right)^2
\end{aligned}
$$

找到 **(w, b)** 使得预测值与真实值之间的误差最小化。

在线性回归中最小二乘法就是试图

找一条直线使所有样本到直线的欧氏距离之和最小.

➢ Parameter Estimation: Least Squares Method(参数估计：最小二乘法)

- **Minimum Squared Error**

$$E_{(w,b)} = \sum_{i=1}^{m} (y_i - wx_i - b)^2$$

- **The derivatives of w and b are obtained respectively as follows:**

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m} (y_i - b)\, x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m} (y_i - wx_i)\right)$$

➢ Parameter Estimation: Least Squares Method(参数估计：最小二乘法)

- **Getting a closed-form（闭式） solution**

$$w=\frac{\sum_{i=1}^{m} y_i\left(x_i-\bar{x}\right)}{\sum_{i=1}^{m} x_i^2-\frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2}$$

$$b=\frac{1}{m} \sum_{i=1}^{m}\left(y_i-w x_i\right)$$

- **where** $\bar{x}=\frac{1}{m} \sum_{i=1}^{m} x_i$ **is the mean of** x_i

8.2 Multivariate Linear Regression
多元线性回归

当每个样本由多个属性描述时，此时寻求的线性模型称为多元线性回归。
When each sample is described by multiple attributes, the linear model is called multiple linear regression.

➢ Multivariate Linear Regression

- **Given dataset**

$$D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$$

$$\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{id}) \quad y_i \in \mathbb{R}$$

- Objective of multivariate linear regression

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b \quad \text{such that} \quad f(\boldsymbol{x}_i) \simeq y_i$$

➢ Multivariate Linear Regression

- Transforming *w and b* into vector form $\hat{w} = (w; b)$, the dataset is represented as:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix}$$

$$\boldsymbol{y} = (y_1; y_2; \ldots; y_m)$$

➢ Multivariate linear regression - Least square method

- **Core Idea: Minimize the Sum of Squared Errors (SSE).**

$$\hat{\boldsymbol{w}}^* = \underset{\hat{w}}{\arg\min}\,(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}}(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})$$

- Let $E_{\hat{\boldsymbol{w}}} = (\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})^{\mathrm{T}}(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}})$, the derivative of $\widehat{w}$ yields:

$$\frac{\partial E_{\hat{\boldsymbol{w}}}}{\partial \hat{\boldsymbol{w}}} = 2\mathbf{X}^{\mathrm{T}}(\mathbf{X}\hat{\boldsymbol{w}} - \boldsymbol{y})$$

- Letting the above equation equal zero yields the closed-form solution for the $\widehat{w}$ optimal solution :

$$\mathbf{X}^T\mathbf{X}\hat{w} = \mathbf{X}^T\boldsymbol{y}$$

➢ Multivariate linear regression - Least square method

- **If $X^T X$ is a full-rank matrix (满秩矩阵) or a positive definite matrix （正定矩阵）, then**

$$\hat{\boldsymbol{w}}^* = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

- **where $(X^T X)^{-1}$ is the inverse matrix (逆矩阵) of $X^T X$, the linear regression model is denoted as:**

$$f\left(\hat{\boldsymbol{x}}_i\right) = \hat{\boldsymbol{x}}_i^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

➢ Multivariate linear regression - Least square method

- **If $X^T X$ is not a full-rank matrix, multiple solutions for $\widehat{w}$ can be derived.**
- **The selection of which solution to output is determined by the learning algorithm's inductive preference, with regularization being a common practice.**（选择哪一解作为输出,将由学习算法的归纳偏好决定，常见的做法是引入正则化 **(regularization)** .）

➢ Limitations of the Least Squares Method （最小二乘法的局限性）

- **Matrix Inversion Problem | 矩阵求逆问题**
- **Fails if X^TX is not invertible (e.g., too many features, correlated features).**
- **Computational Cost | 计算成本高**
- **Calculating the inverse has a complexity of $O(n^3)$, slow for large n.**
- 计算矩阵逆的复杂度为 **$O(n^3)$**，对于大的 **n** 很慢。
- **Sensitivity to Outliers | 对异常值敏感**
- **Squared error heavily penalizes large errors, making the model swayed by outliers.**
- 平方误差会严重惩罚大误差，使模型容易被异常值影响。

8.2 Polynomial Regression

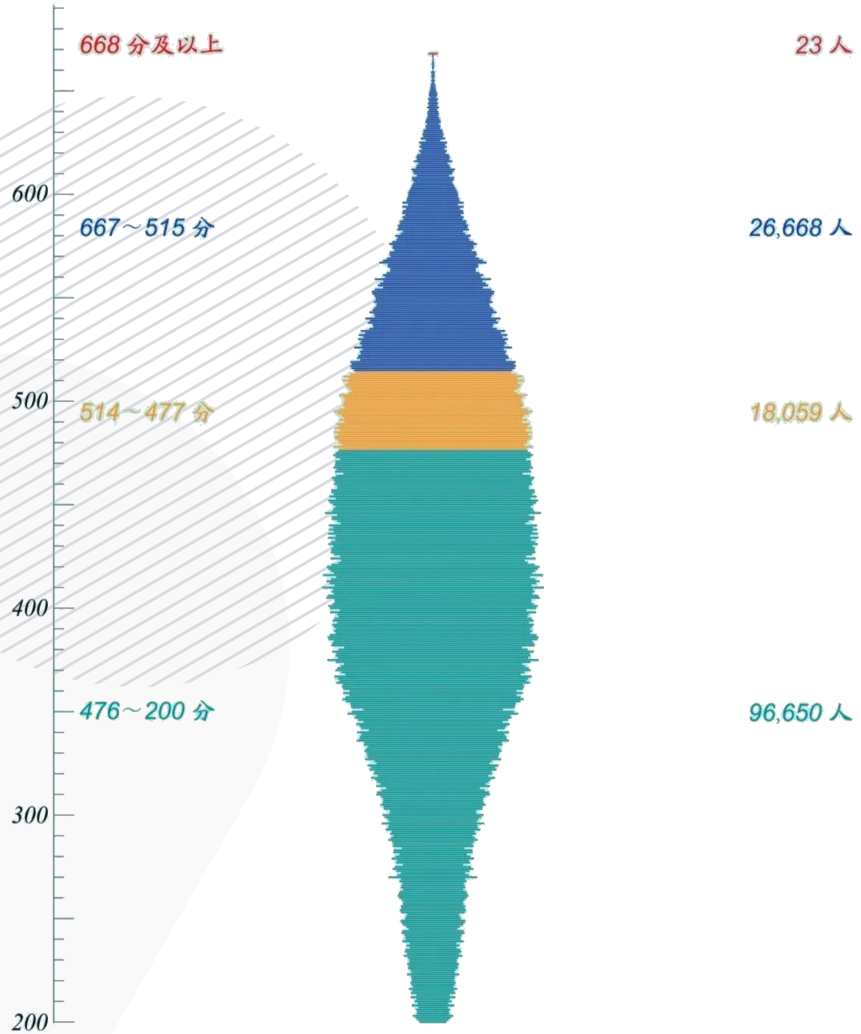
安徽理工大學
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

➢ Limitations of Linear Regression（线性回归的局限性）

- Real-world relationships are often nonlinear. A straight line may not fit well.
- A curve fits better than a straight line.一条曲线比直线拟合得更好。

安徽 2025 年高考（历史类）一分一段 纺锤图

满分：750 分



➢ Add Polynomial Features | 核心思想：增加多项式特征

- The quadratic polynomial regression model is

$$h(\boldsymbol{x};\boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2$$

- Polynomial regression problems can be transformed into multiple linear regression problems through variable transformation.

令 $z_1 = x_1$，$z_2 = x_2$，$z_3 = x_1^2$，$z_4 = x_2^2$，$z_5 = x_1 x_2$，

- Then the quadratic polynomial regression model is transformed into a quintic linear regression model:

$$h(\boldsymbol{x};\boldsymbol{\theta}) = \theta_0 + \theta_1 z_1 + \theta_2 z_2 + \theta_3 z_3 + \theta_4 z_4 + \theta_5 z_5$$

- 它在参数 (w) 上仍然是“线性”的！我们可以使用线性回归技术。

8.3 Gradient Descent Method
梯度下降法

想象你在一座山上，想要到达最低的山谷（代价函数的最小值）。
Imagine you're on a mountain and want to reach the lowest valley (minimum of a cost function).

➢ When Least Squares Fails （当最小二乘法失效时）

- Problem: Computing $\hat{w} = (X^TX)^{-1} X^TY$ can be:
- Fails if X^TX is not invertible (e.g., too many features, correlated features).
- Com
- Calc
- 计算矩阵逆的复杂度为 $O(n^3)$，对于大的 n 很慢。
- Impossible online | 无法在线学习 (requires all data at once | 需要所有数据一次性输入)

We need an iterative, approximate approach.

我们需要一种迭代的、近似的方法。

➢ The Intuition: Finding the Valley | 直观理解：寻找山谷

- **Imagine you're on a mountain and want to reach the lowest valley (minimum of a cost function).**想象你在一座山上，想要到达最低的山谷。
- **Strategy:**
  - **Look around to find the steepest downhill direction (the negative gradient).**环顾四周，找到最陡的下山方向（负梯度方向）。
  - **Take a step in that direction (determined by the learning rate).**朝那个方向迈出一步（步长由学习率决定）。
  - **Repeat until you reach the bottom (convergence).**
  - 重复直到到达谷底（收敛）。

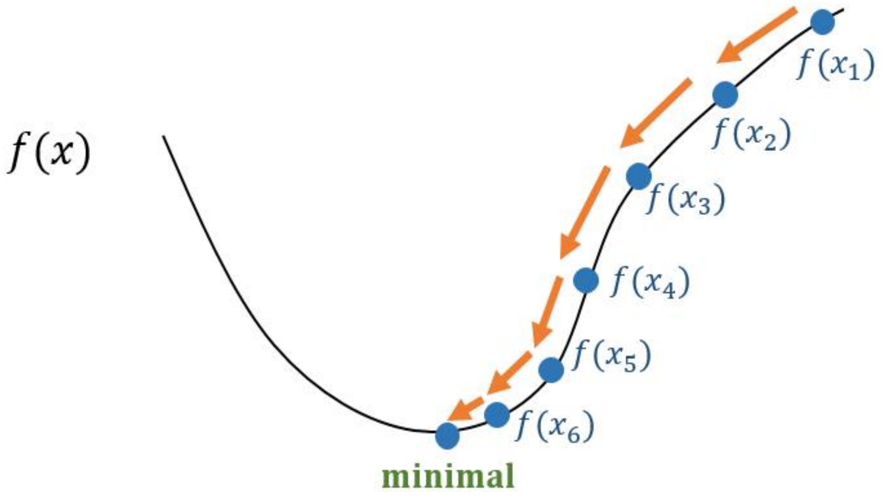
➢ Fundamental Principle of Gradient Descent (梯度下降法的核心原理)

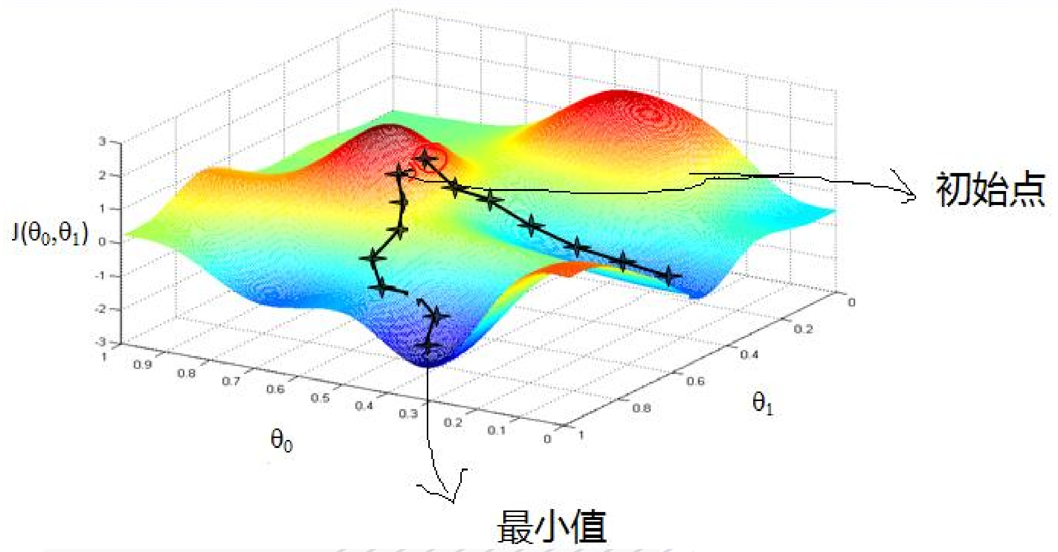
- **The fundamental principle of gradient descent is to iteratively update model parameters in the direction opposite to the gradient vector of the function.** (通过反向调整模型参数，沿着函数梯度向量的方向进行迭代优化。)
- **This approach enables the fastest possible descent in the function value, allowing the model to rapidly approach the function's minimum point until convergence. Ultimately, it achieves a minimized cost function and optimal model parameter values.**(该方法能实现函数值的极速下降，使模型快速逼近函数极小值直至收敛。最终可获得最小化代价函数和最优模型参数值。)

➢ The Intuition: Finding the Valley | 直观理解：寻找山谷

- **We want to minimize a cost function J(w) (e.g., Mean Squared Error).** 最小化一个代价函数 **J(w)**（例如，均方误差）。
- **Gradient (**梯度**): The vector of partial derivatives of J with respect to each parameter w_j.** 对每个参数 w_j 的偏导数组成的向量。
- **Update Rule (for each parameter j):**
- $w_j := w_j - \alpha * (\partial J(w)/\partial w_j)$
- **Where |** 其中：
- **α (alpha) is the learning rate or step size**(学习率或步长)
- **($\partial J(w)/\partial w_j$) is the gradient of J with respect to w_j | J** 对 w_j 的梯度

➢ The Intuition: Finding the Valley | 直观理解：寻找山谷





➢ The Three Variants:Batch Gradient Descent (BGD) | 批量梯度下降

- **Uses all training examples to compute the gradient in each iteration.**
- 每次迭代使用所有训练样本计算梯度。

参数更新

$$w_j := w_j - \overset{\text{学习率}}{\alpha}\frac{1}{m}\sum_{i=1}^{m}\overbrace{\left(\left(h\left(x^{(i)}\right) - y^{(i)}\right)\cdot x_j^{(i)}\right)}^{\text{梯度}}$$

(同步更新w_j， (j=0,1,...,n))

➢ The Three Variants:Batch Gradient Descent (BGD) | 批量梯度下降

- **Pros: Stable convergence to global minimum (for convex functions).**
- 优点： 稳定收敛到全局最小值（对于凸函数）。

- **Cons: Very slow for large datasets.**
- 缺点： 对于大型数据集非常慢。

➢ The Three Variants:Stochastic Gradient Descent (SGD) | 随机梯度下降

- **Uses one random training example to compute the gradient in each iteration.**
- 每次迭代使用一个随机训练样本计算梯度。

- **Pros: Very fast, can escape local minima.**
- 优点： 非常快，可以逃离局部最小值。

参数更新

$$w_j := w_j - \alpha\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

(同步更新w_j，(j=0,1,...,n))

- **Cons: Noisy updates, may not converge to exact minimum.**
- 缺点： 更新有噪声，可能不会精确收敛到最小值。

➢ The Three Variants:Mini-batch Gradient Descent (MBGD) | 小批量梯度下降

- **Uses a small random subset (batch) of examples to compute the gradient.**
- 使用一个小的随机样本子集（批量）计算梯度。
- **Pros: Balance of speed and stability. Most common in practice!**
- 优点： 速度和稳定性的平衡。实践中最常用！

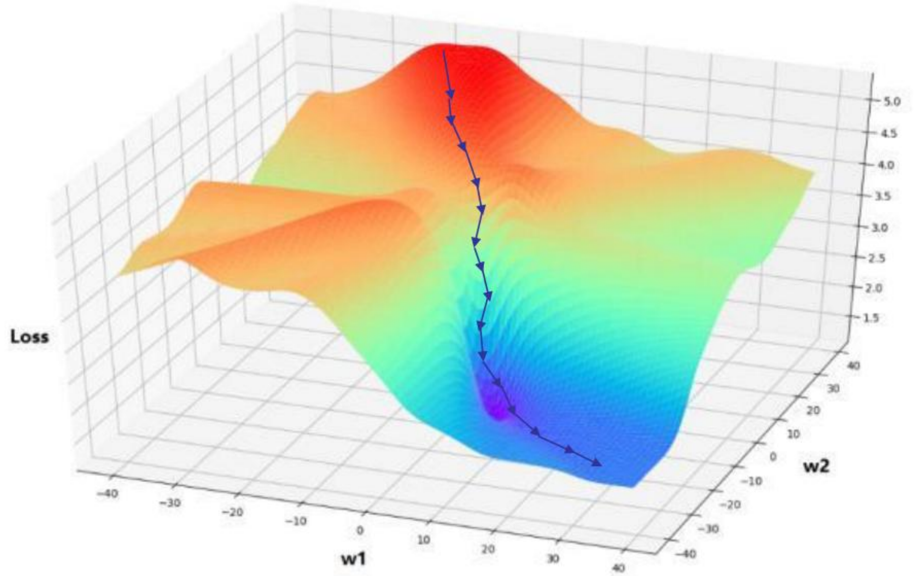
参数更新

$$w_j := w_j - \alpha \frac{1}{b} \sum_{k=i}^{i+b-1} \left(h\left(x^{(k)}\right) - y^{(k)}\right) x_j^{(k)}$$

(同步更新w_j，(j=0,1,...,n))

b=1（随机梯度下降,SGD）
b=m（批量梯度下降,BGD）
b=*batch_size*，通常是2的指数倍，常见有32,64,128等。（小批量梯度下降,MBGD）

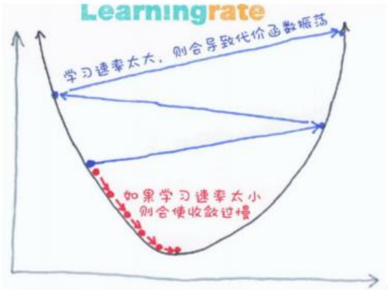
8.3 Gradient Descent Method

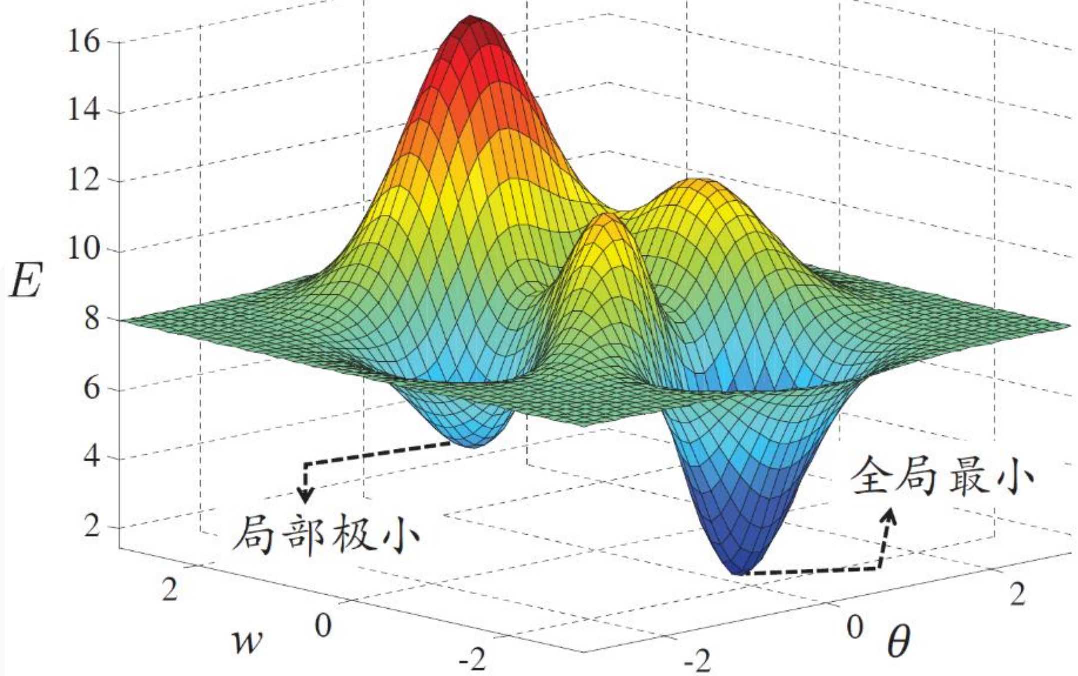


学习率

α

步长





8.3 Gradient Descent Method

Type 类型	Gradient Computation 梯度计算	Speed 速度	Stability 稳定性	Use Case 适用场景
Batch 批量	All data 所有数据	Slow 慢	Very Stable 非常稳定	Small datasets 小数据集
Stochastic 随机	One example 一个样本	Very Fast 非常快	Unstable 不稳定	Online learning 在线学习
Mini-batch 小批量	Small batch 小批量	Fast 快	Stable 稳定	Most ML problems 大多数ML问题

➢ Summary

- **Gradient Descent is an iterative optimization algorithm for finding local minima.**梯度下降 是一种迭代优化算法，用于寻找局部最小值。
- **It's essential for large-scale machine learning where closed-form solutions are impractical.**在大规模机器学习中必不可少，因为闭式解不实用。
- **The learning rate α is critical and must be carefully chosen/tuned.**学习率 α 至关重要，必须仔细选择/调整。
- **It enables online learning and can handle non-convex functions (like neural networks).**它支持在线学习，并且可以处理非凸函数（如神经网络）。

安徽理工大學
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

➢ Summary

- **Gradient Descent is an iterative optimization algorithm for finding local minima.** 梯度下降 是一种迭代优化算法，用于寻找局部最小值。
- **It's essential for large-scale machine learning where closed-form solutions are impractical.** 在大规模机器学习中必不可少，因为闭式解不实用。
- **The learning rate α is critical and must be carefully chosen/tuned.** 学习率 α 至关重要，必须仔细选择/调整。
- **It enables online learning and can handle non-convex functions (like neural networks).** 它支持在线学习，并且可以处理非凸函数（如神经网络）。

8.4 Logit Regression 、Logistic Regression

对数几率回归、逻辑回归

我们能将线性回归用于分类任务吗?

Can we use linear regression for classification tasks?

From Regression to Classification

- **Question: Can we use linear regression for classification tasks?**
- **Challenges | 挑战:**
  - **Linear regression outputs continuous values, but classification needs discrete labels**
  - 线性回归输出连续值，但分类需要离散标签
  - **Real-world probabilities are bounded between 0 and 1**
  - 真实世界的概率被限制在0和1之间

➢ From Regression to Classification

- **Ideal function: Unit step function 单位阶跃函数**

$$y = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$$

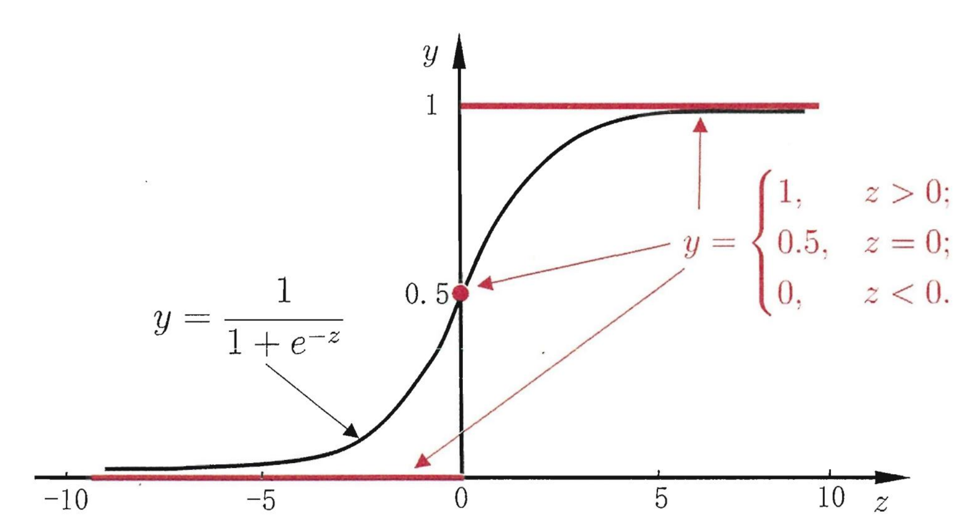
- 但单位阶跃函数不连续、不可微 → 难以优化

➤ Logistic Function

- **Logistic function (sigmoid):**
- 对数几率函数/逻辑函数（**sigmoid**）：

$$y = \frac{1}{1+e^{-z}}$$

- **Properties | 特性:**
  - **Monotonic and differentiable | 单调且可微**
  - **Bounded between 0 and 1 | 限制在0和1之间**
  - **Smooth curve | 平滑曲线**



Logistic Regression Model

- **Combine linear regression with logistic function:**

$$y = \frac{1}{1 + e^{-(W^T x + b)}}$$

$$\ln \frac{y}{1 - y} = W^T x + b$$

- **y**可视为**x**为正例的可能性
- 则**1-y**可视为反例的可能性

➢ Logistic Regression Model

- **Odds (几率): Ratio of probability of success to probability of failure.** 成功概率与失败概率的比值

$$\text{Odds} = \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} = \frac{p}{1-p}$$

$$\text{Log-odds} = \ln\left(\frac{p}{1-p}\right)$$

- **Log-odds (对数几率): Natural logarithm of odds** 几率的自然对数
  - **Maps probabilities from [0,1] to (-∞, +∞) |** 将概率从**[0,1]**映射到**(-∞, +∞)**
  - **Linear relationship with features |** 与特征呈线性关系
  - **Enables linear regression framework |** 可以使用线性回归框架

➢ Logistic Regression Model

- **Logistic**

$$\ln\frac{y}{1-y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b \xrightarrow{y \sim p(y=1|\boldsymbol{x})} \ln\frac{p(y=1\mid\boldsymbol{x})}{p(y=0\mid\boldsymbol{x})} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$$

- **Obviously , there are**

$$p(y=1\mid\boldsymbol{x}) = \frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$$

$$p(y=0\mid\boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$$

➢ Logistic Regression Model

- 极大似然估计：就是利用已知的样本结果信息，反推最具有可能（最大概率）导致这些样本结果出现的模型参数值！Maximum likelihood method（极大似然法）is the method of selecting the parameter values that maximise the probability of the observed data occurring as the optimal estimate.
- Given dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$
- Maximise the probability that a sample belongs to its true label
- Maximise the Log-Likelihood Function

$$\ell(\boldsymbol{w}, b) = \sum_{i=1}^{m} \ln p(y_i \mid \boldsymbol{x}_i; \boldsymbol{w}_i, b)$$

8.5 Linear Discriminant Analysis
线性判别分析

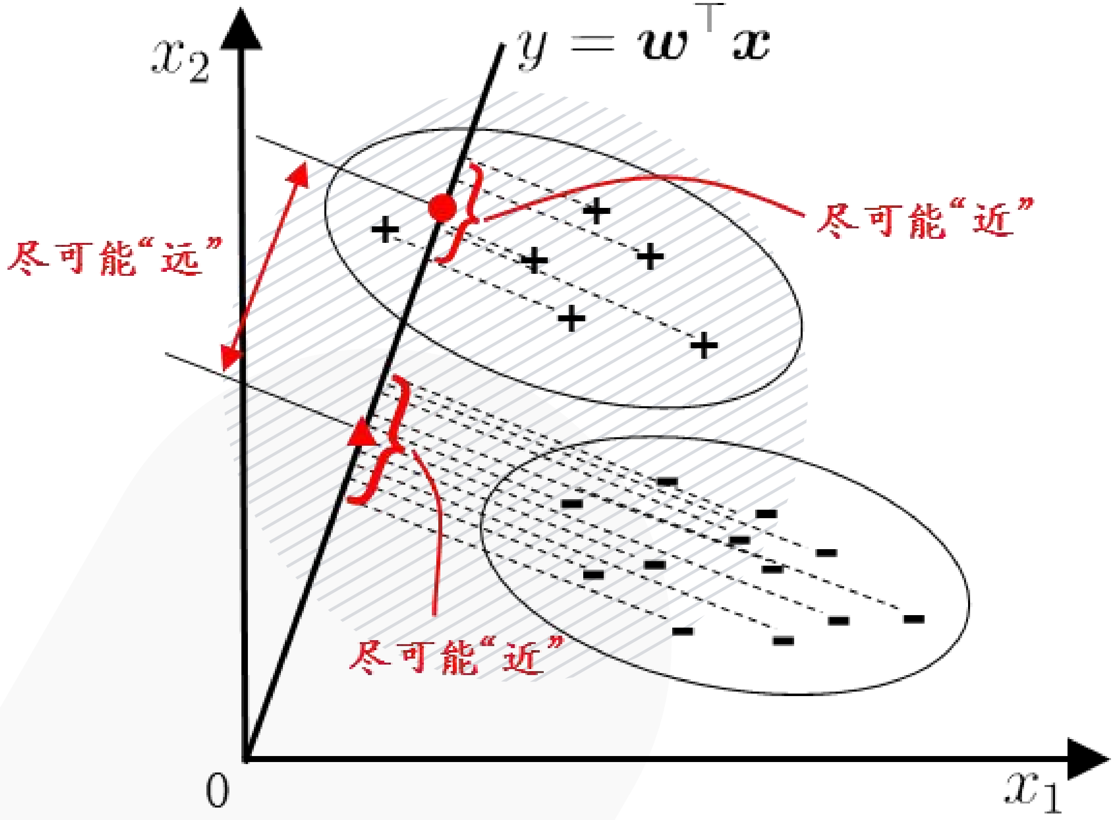
How to best separate two classes?

- **Imagine you're looking at scattered dots in 3D space.**
- 想象你在三维空间中观察散落的点。

- **Goal: Find a viewpoint that makes the separation most clear.**
- 目标：找到一个能最清晰区分类别的视角。

➢ How to best separate two classes?

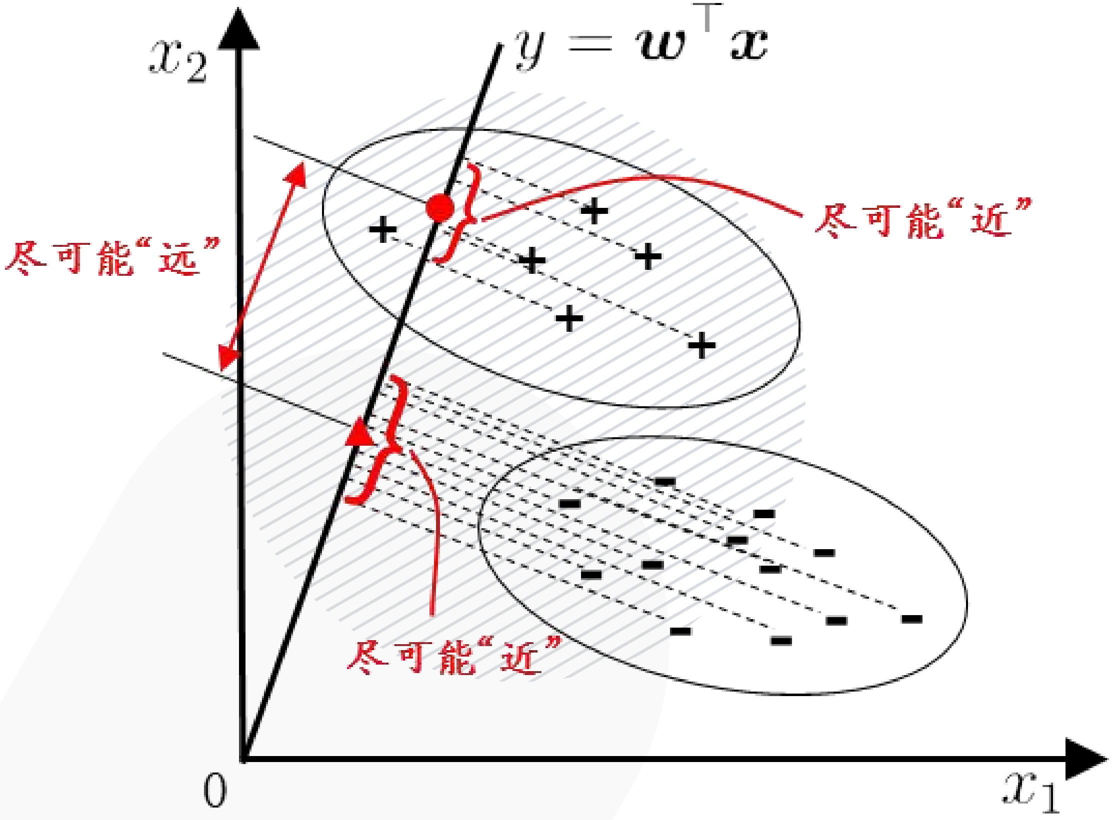
- **LDA finds the best line to project data so that:**
- **LDA找到最佳直线来投影数据，使得:**
- **Same-class points are close together | 同类点尽可能靠近**
- **Different-class points are far apart | 不同类点尽可能远离**



➢ How to best separate two classes?

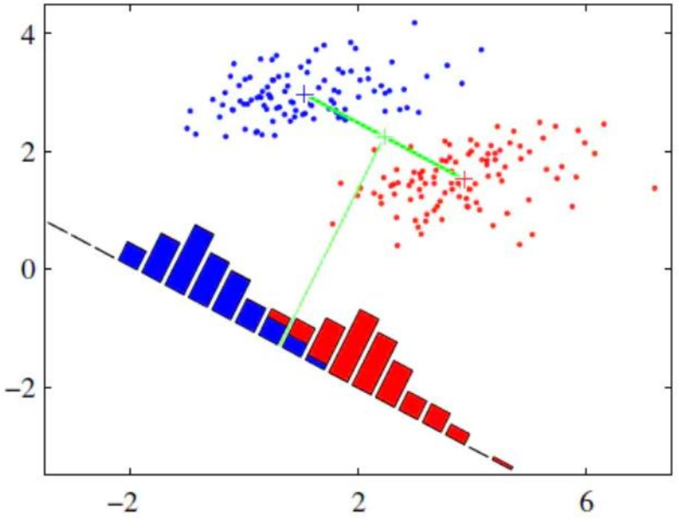
- **LDA finds the best line to project data so that:**
- **LDA找到最佳直线来投影数据，使得:**
- **Same-class points are close together | 同类点尽可能靠近**
- **Different-class points are far apart | 不同类点尽可能远离**

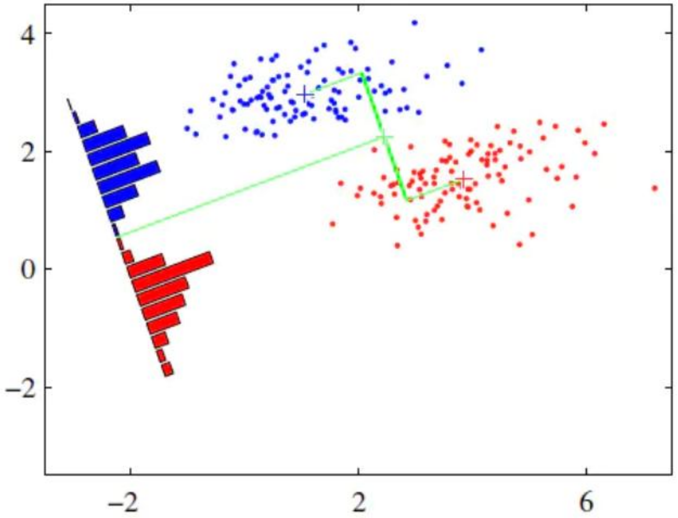
LDA can also be regarded as a supervised dimensionality reduction technique.
LDA也可被视为一种监督降维技术



➢ The idea of LDA:

- **The projection points of the same class are as close as possible, and the covariance between projection places is minimised.**同类别投影点尽可能靠近，且各投影点间的协方差最小化。
- **The projection points of different classes are as far apart as possible, and the distance between different class centres is maximised.**不同类别投影点尽可能远离，且不同类别中心间的距离最大化。





➢ How to Measure "Closeness" and "Separation"?如何衡量"接近"与"远离"?

- **Introducing Scatter Matrices**（散度矩阵）
- 类内散度矩阵 S_W：衡量同类样本的分散程度

$$\begin{aligned}\mathrm{S}_w &= \Sigma_0 + \Sigma_1 \\ &= \sum_{\boldsymbol{x}\in X_0}(\boldsymbol{x}-\boldsymbol{\mu}_0)(\boldsymbol{x}-\boldsymbol{\mu}_0)^{\mathrm{T}} + \sum_{\boldsymbol{x}\in X_1}(\boldsymbol{x}-\boldsymbol{\mu}_1)(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\end{aligned}$$

- 类间散度矩阵 S_b：衡量不同类样本中心的距离

$$\mathrm{S}_b = (\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)(\boldsymbol{\mu}_0-\boldsymbol{\mu}_1)^{\mathrm{T}}$$

- 目标函数：最大化 $\frac{S_b}{S_W}$

➤ Generalized Rayleigh Quotient (广义瑞利商)

- 将分类问题转化为优化问题 | **Turning Classification into an Optimization Problem**

$$J = \frac{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\mathbf{S}_w\boldsymbol{w}}$$

- **w** 是投影方向
- 最大化 **J(w)** → 最大化类间距离、最小化类内距离

➢ Why Do We Need Lagrange Multipliers?

- 我们想最大化 J(w)， 但 w 不能任意缩放 | We want to maximize J(w), but w cannot be arbitrarily scaled.
- Add constraint: (添加约束条件)

$$w^T S_w w = 1$$

- 拉格朗日乘子法可将约束问题转化为无约束问题 | Lagrange multipliers turn constrained problems into unconstrained ones

➢ Introduction to Lagrange Multipliers

- **Core Idea: Incorporate Constraints into the Objective Function**

$$\max_{w} J(w) \quad \text{s.t.} \quad g(w) = 0$$

- **Construct Lagrangian (构造拉格朗日函数):**

$$\mathcal{L}(w, \lambda) = J(w) - \lambda \cdot g(w)$$

- **其中 λ 是拉格朗日乘子 | where λ is the Lagrange multiplier**

➢ Applying Lagrange Multipliers in LDA

- 构造拉格朗日函数 | Constructing the Lagrangian Function

$$\mathcal{L}(w,\lambda) = w^T S_b w - \lambda\left(w^T S_w w - 1\right)$$

- 第一项： 类间散度 | First term: Between-class scatter
- 第二项： 带乘子的约束条件 | Second term: Constraint with multiplier
- 目标： 求 L 对 w 和 λ 的极值 | Goal: Find extremum of L w.r.t. w and λ

➤ Solving for the Optimal Projection Direction

- 对拉格朗日函数求导并令为零 | **Take Derivatives and Set to Zero**

$$\frac{\partial \mathcal{L}}{\partial w} = 2S_b w - 2\lambda S_w w = 0$$

- 整理得广义特征值问题:
- **Simplify to generalized eigenvalue problem:** $S_b w = \lambda S_w w$
- 解 **w** 即为最优投影方向 | **The solution w is the optimal projection direction**

➢ Extension to Multi-Class and Dimensionality Reduction

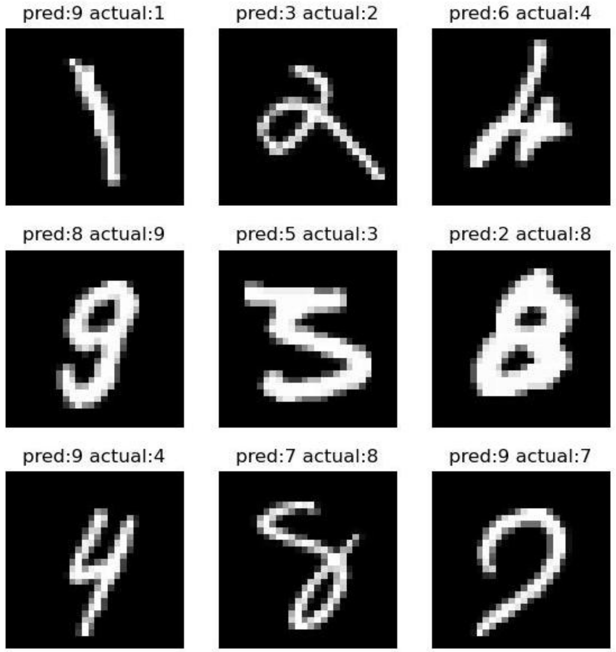
- **LDA 不仅是分类器，也是监督降维方法 | LDA is Both a Classifier and a Supervised Dimensionality Reduction Method**
- **多分类：构建全局散度矩阵，求解多个投影方向 | Multi-class: Use global scatter matrices, solve for multiple projection directions**
- **投影后低维空间便于分类与可视化 | Low-dimensional projection aids classification and visualization**
- **LDA 可作为特征提取器使用 | LDA can serve as a feature extractor**

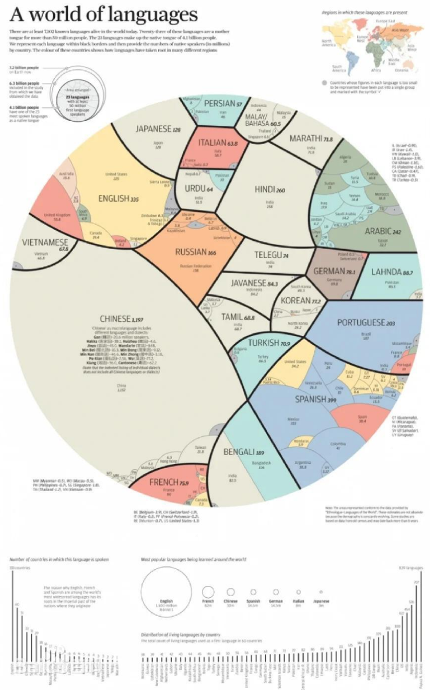
8.6 Multi-class Learning Methods
多类别学习方法

Most real-world problems have more than 2 classes!
大多数现实世界问题都有超过2个类别!

➢ From Binary to Multi-class

- Digit recognition (0-9) → 10 classes | 数字识别（0-9）→ 10个类别
- Animal species classification → 100+ classes | 动物物种分类 → 100+个类别
- Language identification → 100+ classes | 语言识别 → 100+个类别





➢ From Binary to Multi-class

- **Problem: Most classifiers are designed for binary tasks!**
- 问题：大多数分类器是为二分类任务设计的!

- **Solution: Convert multi-class into multiple binary problems.**
- 解决方案：将多分类问题转换为多个二分类问题。

➢ Strategy 1: One-vs-One (OvO)|一对一

- **Basic Idea: Every class vs every other class!** （每个类别与每个其他类别比较！）
- **For N classes, we train:**

$$\frac{N(N-1)}{2} \text{ binary classifiers}$$

- **Each classifier learns to distinguish between exactly 2 classes.**
- 每个分类器学习区分恰好两个类别。

➢ Strategy 1: One-vs-One (OvO)|一对一

- **OvO Training Process**
  - **4 classes (A,B,C,D) → 6 binary classifiers**
  - **(4个类别 → 6个二分类器)**

Classifier 1: A vs B	**Classifier 4: B vs D**
Classifier 2: A vs C	**Classifier 5: C vs D**
Classifier 3: A vs D	**Classifier 6: B vs C**

  - **Training data for A vs B: Only samples from A and B!**
  - **A vs B的训练数据：仅来自A和B的样本!**

➢ Strategy 1: One-vs-One (OvO)|一对一

- **OvO Prediction Process**
  - **For a new sample: |** 对于一个新样本:
  - **Submit to all N(N-1)/2 classifiers |** 提交给所有**N(N-1)/2**个分类器
  - **Each classifier votes for one class |** 每个分类器为一个类别投票
  - **Class with most votes wins! |** 得票最多的类别获胜!

➢ Strategy 1: One-vs-One (OvO)|一对一

- Voting example:
  - A vs B → votes for A (A vs B → 投票给A)
  - A vs C → votes for A (A vs C → 投票给A)
  - A vs D → votes for D (A vs D → 投票给D)
  - B vs C → votes for C (B vs C → 投票给C)
  - B vs D → votes for B (B vs D → 投票给B)
  - C vs D → votes for C (C vs D → 投票给C)

Final: A:2 votes, B:1 vote, C:2 votes, D:1 vote → Winner: ?

最终：A:2票，B:1票，C:2票，D:1票 → 胜者：？

➢ Strategy 2: One-vs-Rest (OvR)|一对其余

- **Basic Idea: One class vs all other classes combined!**一个类别与所有其他类别组合对比!
- **For N classes, we train:**

  N binary classifiers

- **Each classifier answers: "Is this class X or not class X?"**
- 每个分类器回答："这是类别**X**，还是非类别**X**？"

➢ Strategy 2: One-vs-Rest (OvR)|一对其余

- **OvR Training Process**
  - **4 classes (A,B,C,D) → 4 binary classifiers(4个类别 → 4个二分类器)**
  - **Classifier 1: A vs (B,C,D)**
  - **Classifier 2: B vs (A,C,D)**
  - **Classifier 3: C vs (A,B,D)**
  - **Classifier 4: D vs (A,B,C)**
  - **Training data for A vs Rest: All samples! A=positive, others=negative**
  - **A vs 其余的训练数据：所有样本！A=正类，其他=负类**

➢ Strategy 2: One-vs-Rest (OvR)|一对其余

- **OvR Prediction Process**
  - **For a new sample:** | 对于一个新样本:
  - **Submit to all N classifiers** | 提交给所有N个分类器
  - **Each classifier gives a confidence score** | 每个分类器给出置信度分数
  - **Class with highest confidence wins!** | 置信度最高的类别获胜!

➢ Strategy 2: One-vs-Rest (OvR)|一对其余

- **OvR Prediction Process**
  - **Confidence scores:** | 置信度分数:
  - **Classifier 1 (A vs Rest): confidence = 0.85**
  - **Classifier 2 (B vs Rest): confidence = 0.42**
  - **Classifier 3 (C vs Rest): confidence = 0.73**
  - **Classifier 4 (D vs Rest): confidence = 0.61**

➢ OvO vs OvR: Pros and Cons

Aspect 方面	OvO 一对一	OvR 一对其余
Number of classifiers 分类器数量	N(N-1)/2 (更多)	N (更少)
Training data per classifier 每个分类器的训练数据	Smaller (仅2类)	Larger (所有数据)
Training time 训练时间	Shorter per classifier 每个分类器时间短	Longer per classifier 每个分类器时间长
Testing time 测试时间	Longer (需所有分类器投票)	Shorter (只需N个分类器)
Imbalanced data 不平衡数据	Less affected 受影响小	More affected 受影响大
Storage 存储	More classifiers to store 存储更多分类器	Fewer classifiers 更少分类器

➢ Strategy 3: Many-vs-Many (MvM) |多对多

- **Basic Idea: Group classes into positive and negative groups! |** 基本思路：将类别分组为正类和负类!
- **We need: |** 我们需要:
- **A coding matrix (**编码矩阵**)**
- **M binary classifiers (M**个二分类器**)**
- **A decoding scheme (**解码方案**)**

➢ Strategy 3: Many-vs-Many (MvM)： Coding Matrix Example

- For 4 classes (A,B,C,D), using 3 classifiers:

Class	Classifier 1	Classifier 2	Classifier 3
A	+1	-1	-1
B	-1	+1	-1
C	-1	-1	+1
D	+1	+1	-1

- Where: | 其中：
- +1 means "treat as positive class" | +1表示"视为正类"
- -1 means "treat as negative class" | -1表示"视为负类"

➢ Strategy 3: Many-vs-Many (MvM)

- MvM Training Process
  - Look at coding matrix column | 查看编码矩阵的列
  - Group classes with +1 as positive | 将+1的类别分组为正类
  - Group classes with -1 as negative | 将-1的类别分组为负类
  - Train binary classifier | 训练二分类器
  - Example for Classifier 1: | 分类器1的示例：
    - Positive: A, D (because column has +1 for A and D)
    - Negative: B, C (because column has -1 for B and C)

➢ Strategy 3: Many-vs-Many (MvM)

- **MvM Prediction Process**
  - **Get predictions from all M classifiers | 从所有M个分类器获取预测**
  - **Get a code vector of length M | 得到长度为M的编码向量**
  - **Compare with each class's code | 与每个类别的编码比较**
  - **Choose class with smallest distance | 选择距离最小的类别**
  - **Example:**
    - **Predicted: [+0.9, -0.8, -0.7]**
    - **Distances: d(A)=0.3, d(B)=2.1, d(C)=2.2, d(D)=1.7**

➢ Strategy 3: Many-vs-Many (MvM)：Error Correcting Output Codes (ECOC)

Coding: Perform M partitions on N categories, with each partition assigning a portion of the categories to the positive class and another portion to the negative class.

M binary tasks

Each category consists of an M-length coding sequence

The category with the smallest distance is the final category.

Decoding: Test samples are submitted to M classifiers for prediction.

Prediction of an M-length coding sequence

➢ Strategy 3: Many-vs-Many (MvM)

- **Why Error Correction Works?**
- **Without error correction: | 没有纠错:**
  - **True code: [+1, +1, +1, -1, +1]**
  - **Prediction: [+1, +1, -1, -1, +1] ← 1 error at position 3**
  - **Result: Wrong classification!**
- **With error correction: | 有纠错:**
  - **We find closest valid code in codebook.**
  - **Closest is [+1, +1, +1, -1, +1] → Class A!**

8.7 Class Imbalance Problem
类别不平衡问题

What happens when a dataset contains 998 negative examples and 2 positive ones?
当一个数据集有998个反例，2个正例的时候会怎样?

安徽理工大學
ANHUI UNIVERSITY OF SCIENCE & TECHNOLOGY

➢ What is Class Imbalance?

- When the number of training examples varies considerably across different classes. 当不同类别的训练样本数量差异巨大时。
- Imbalanced dataset (不平衡数据集)
  - Class 0 (正例|多数): 998 samples
  - Class 1 (负例|少数): 2 samples
- A classifier can achieve 99.8% accuracy if it returns only negative examples.

➢ Solution 1 - Oversampling（过采样）

- 增加少数类样本的数量
  - Simple oversampling: Randomly duplicate minority class samples
  - 简单过采样：随机复制少数类样本
  - Problem: Leads to overfitting! (model sees same examples repeatedly)
  - 问题：导致过拟合！（模型重复看到相同的样本）
  - 智能生成新样本（插值）

➢ Solution 2 - Undersampling（欠采样）

- 减少多数类样本的数量
  - Simple undersampling: Randomly remove majority class samples
  - 简单欠采样：随机移除多数类样本
  - Problem: Loss of potentially useful information!
  - 问题：丢失潜在的有用信息！

- 使用多个子集，训练多个分类器！

➢ Solution 3 - Threshold Moving（阈值移动）

- 在不改变训练数据的情况下调整决策阈值

- Standard: Predict positive if probability > 0.5
- 标准：如果概率 > 0.5，预测为正类

- Threshold moving: Find optimal threshold using validation set
- 阈值移动：使用验证集找到最优阈值

➢ Basic Strategy - Rescaling (Re-weighting)

- **Idea: Adjust decision threshold based on class ratio | 根据类别比例调整决策阈值**
- **Standard classification: Predict positive if p(y=1|x) > 0.5**

$$\frac{y}{1-y} > 1$$

- **Rescaled classification: Predict positive if p(y=1|x) > threshold**

$$\frac{y}{1-y} > \frac{m^+}{m^-}$$

- **However, accurately estimating m-/m+ is often difficult!**