ML基础概念类

Overfitting/Underfiting

过拟合（Overfitting）：模型在训练数据上表现很好，但在测试数据上表现差，原因是模型过于复杂，记住了训练数据中的噪声。

欠拟合（Underfitting）：模型在训练数据和测试数据上都表现不好，原因是模型过于简单，无法捕捉数据的特征。

Overfitting: The model performs well on training data but poorly on test data because it is too complex and captures noise in the training data.

Underfitting: The model performs poorly on both training and test data because it is too simple to capture the underlying patterns in the data.

Bias/Variance trade off

偏差-方差权衡（Bias-Variance Trade-off）是机器学习中的核心概念，指模型复杂度与预测性能之间的平衡：

偏差（Bias）：指模型对数据的拟合能力，偏差高时容易欠拟合。

方差（Variance）：指模型对数据波动的敏感性，方差高时容易过拟合。

增加模型复杂度会降低偏差但提高方差，降低模型复杂度会降低方差但提高偏差，权衡二者是训练好模型的关键。

The Bias-Variance Trade-off is a fundamental concept in machine learning, describing the balance between model complexity and prediction performance:

Bias: Represents the model's ability to fit the data; high bias leads to underfitting.

Variance: Reflects the model's sensitivity to data fluctuations; high variance leads to overfitting.

Increasing model complexity reduces bias but increases variance, while reducing complexity decreases variance but increases bias. Balancing the two is key to building a good model.

过拟合预防手段

预防过拟合的常用手段包括：

1. 正则化：加入L1（Lasso）或L2（Ridge）正则化约束，限制模型复杂度。
2. 增加数据量：更多数据能帮助模型更好地泛化。
3. 特征选择：去除冗余或不重要的特征，减少噪声。
4. 早停（Early Stopping）：在验证集误差开始增大时停止训练。
5. 数据增强：在图像等任务中，通过旋转、裁剪等生成更多样本。
6. Dropout：随机丢弃神经网络中的部分神经元，防止过拟合（深度学习中常用）。
7. 交叉验证：使用K折交叉验证来选择合适的模型超参数。

Common techniques to prevent overfitting include:

1. Regularization: Apply L1 (Lasso) or L2 (Ridge) regularization to constrain model complexity.
2. Increase data size: More data helps the model generalize better.
3. Feature selection: Remove redundant or irrelevant features to reduce noise.
4. Early Stopping: Stop training when the validation error starts to increase.
5. Data augmentation: Generate more samples through transformations like rotation or cropping (used in tasks like image classification).
6. Dropout: Randomly drop neurons in neural networks to reduce overfitting (commonly used in deep learning).
7. Cross-validation: Use K-fold cross-validation to select optimal hyperparameters for the model.

Generative和Discrimitive的区别

Generative模型：学习数据的联合分布 P(x,y)P(x, y)，可以生成数据（如GAN、VAE），同时也能用于分类（如朴素贝叶斯）。

Discriminative模型：学习条件概率 P(y∣x)P(y|x) 或直接学习决策边界，用于分类（如逻辑回归、SVM）

区别：Generative关注数据生成，Discriminative关注分类性能。

Generative models: Learn the joint distribution P(x,y)P(x, y), can generate data (e.g., GAN, VAE), and also perform classification (e.g., Naive Bayes).

Discriminative models: Learn the conditional probability P(y∣x)P(y|x) or directly the decision boundary, focusing on classification (e.g., Logistic Regression, SVM).

Difference: Generative models emphasize data generation, while Discriminative models focus on classification performance.

Model Comparison

Give a set of ground truths and two models, how do you be confident that one model is better than another?

Reguarlization:

L1 vs L2 作用

L1正则化：使用 $||w||_1 = \sum |w_i|$，倾向于稀疏特征选择（权重变为0）。

L2正则化：使用 $|w||_2^2 = \sum w_i^2$，倾向于小但不为0的权重，适合平滑模型。

区别：L1更适合特征选择，L2更适合防止过拟合。

L1 Regularization: promotes sparsity by driving weights to zero.

L2 Regularization: keeps weights small but non-zero, leading to smoother models.

Difference: L1 is better for feature selection, L2 is better for preventing overfitting.

为什么L1 L2 能起作用

解释一：移动导数为0的位置

原始目标函数 $L(w)$ 的曲线是棕色，最小值点的 w 不为 0。

加入 L_2 正则化后，正则化目标函数变成 $L(w) + Cw^2$，最小值点在黄点处，对应的w的绝对值减小了，但是仍然非 0。由于梯度是 $L'(w) + 2cw$，所以梯度为 0 的地方不在 0

加入 L_1 正则化后，正则项的导数在原点左边是$-C$，右边是C，所以正则化目标函数的导数在原点左边是$L'(w) - C$，只要原目标函数的导数绝对值小于C，那么正则化的目标函数在原点左边始终递减，右边始终递增，最小值点就在原点。

解释二：相当于加了贝叶斯先验

1. 当 w 的先验分布为高斯或者拉普拉斯，分布会让 w 接近0
    - 拉普拉斯分布在 $x=0$ 处是一个尖峰 ，概率很大，所以拉普拉斯先验分布中参数 w 取值为 0 的可能性很高
    - 高斯分布在$x=0$处是平缓的，也就是高斯先验分布认为 w 在$x=0$附近取不同值的可能性是接近的。所以L2正则化只会让w更接近0点，但不会等于0。
    - 拉普拉斯分布 比 高斯分布更陡峭
2. 目标函数里面加 正则项，相当于加先验分布

$$\hat{w} = \text{argmin}_w L(w) + w^2 = \text{argmax}_w p(X|w) + p(w)$$
$$= \text{argmax}_w \log p(X|w) + \log p(w) = \text{argmax}_w p(X|w) \times p(w) = \text{argmax}_w p(w|X)$$

倒数第二个公式的原因： $p(w|X) = \frac{p(X|w) \times p(w)}{p(X)}$

解释三：定义解空间形状

我们可以从一个带约束条件的优化问题推到 带正则项的目标函数，这证明带正则项就是为参数定义了一个解空间。如果是约束条件是$||w||_2^2$，则解空间是一个椭圆。如果是约束条件是$||w||$，则解空间是一个菱形。

Step1:“带约束条件” 的优化问题

$$\min \ L(w) \tag{3}$$
$$\text{s.t. } \|w\|_2^2 \leq m. \tag{4}$$

Step2: 为了求解带约束条件的凸优化问题，写出拉格朗日函数 $L(w) + \lambda(\|w\|_2^2 - m)$.

Step3: 若 $w*$ 和 $\lambda*$ 分别是原问题和对偶问题的最优解，则根据 KKT 条件，它们应满足

$$\begin{cases} 0 = \nabla_w \left(\sum_{i=1}^N (y_i - w^{*\top} x_i)^2 + \lambda^* \left(\|w^*\|_2^2 - m \right) \right), \\ 0 \leq \lambda^*. \end{cases}$$

这就是一个带正则项的目标函数

为什么regularization用L1 L2，而不是L3, L4

计算简单，便于解释

Metric

Precision vs Recall

Precision: The proportion of predicted positive samples that are truly positive. $\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}$.

Flag 了这么多样本 as 正样本，哪些是真的正样本。Example：垃圾邮件

Recall: The proportion of all actual positive samples that are correctly predicted as positive.

$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$，有这么多正样本，哪些被 flagged 了。Example：病人

Metric for Imbalanced Data

Accuracy becomes unreliable (flagging all as negative), use F1 or PR-AUC.

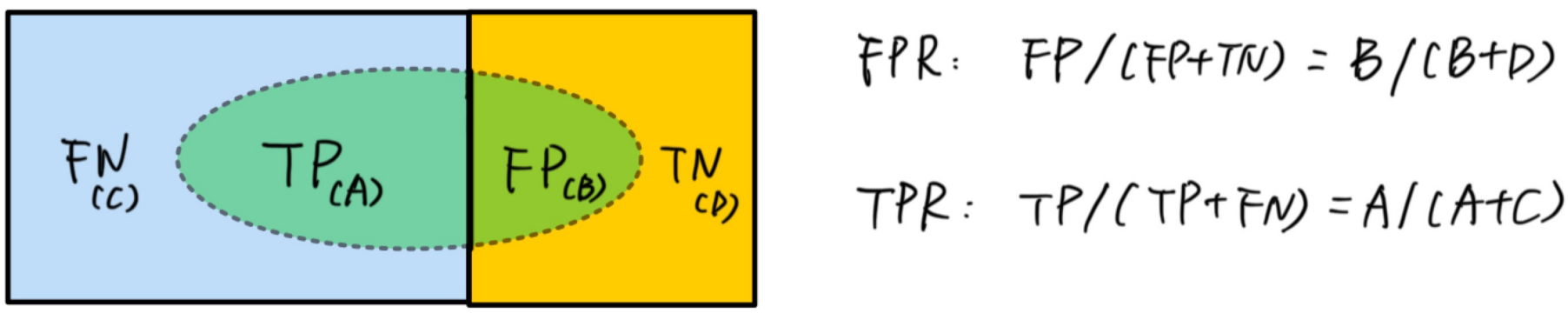
F1=2·Precision·Recall / (Precision+Recall)

Metric for Classification

1. 准确率（Accuracy）：类平衡时使用，简单直观。
2. 精确率（Precision）：减少误报（FP）时使用，如垃圾邮件检测。
3. 召回率（Recall）：减少漏报（FN）时使用，如癌症检测。
4. F1分数：类不平衡且需权衡Precision与Recall时使用。
5. ROC-AUC：评估正负类区分能力，适用于不平衡数据。

6. PR-AUC：极端不平衡场景，关注正类性能。
7. Accuracy: Use for balanced classes; simple and intuitive.
8. Precision: Focus on reducing false positives, e.g., spam detection.
9. Recall: Focus on reducing false negatives, e.g., cancer detection.
10. F1 Score: Use when balancing Precision and Recall for imbalanced classes.
11. ROC-AUC: Assess class separation, suitable for imbalanced data.
12. PR-AUC: Use in extreme imbalance, focuses on positive class performance.

ROC-AUC



x轴是FPR，y轴是TPR，FPR 和 TPR 就是绿色部分在两个方框中的比例

AUC表示模型将随机选择的正样本排在随机选择的负样本之前的概率。

AUC represents the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample.

Confusion Matrix

A 2×2 table shows the number of TP, FP, FN, and TN samples.

Log Loss

适用情境：classification using probability

$$\text{Log-loss (binary)} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$

$$\text{Log-loss (multi-class)} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\log(p_{i,j})$$

Loss与优化

Linear Regression最小二乘法和MLE关系

当假设误差服从正态分布时，线性回归的最小二乘法（OLS）和最大似然估计（MLE）是等价的

最小二乘法：目标是最小化预测值与真实值的平方误差： $\min_w \sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

最大似然估计：

- 假设误差 $\epsilon = y - Xw$ 服从正态分布 $\mathcal{N}(0, \sigma^2)$，似然函数为： $L(w) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - Xw)^2}{2\sigma^2}\right)$
- 对数似然为： $\log L(w) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
- 最大化对数似然函数等价于最小化 $\sum(y_i - \hat{y}_i)^2$，即最小二乘法目标。

关系：最小二乘法是最大似然估计在正态分布假设下的特例。

Cross Entropy vs K-L Divergence

1. 交叉熵（Cross-Entropy）

定义：交叉熵衡量两个概率分布 P 和 Q 的差异，Cross-entropy measures the difference between two probability distributions P and Q

用公式表示为： $H(P,Q) = -\sum_x P(x) \log Q(x)$

直观解释: 如果 $P(x)$ 是真实分布，$Q(x)$ 是模型预测分布，交叉熵表示用 Q 去近似 P 时的 cost。

特点：值越小，说明 Q 越接近 P。

2. Kullback-Leibler (K-L) 散度

定义：K-L 散度衡量两个概率分布 P 和 Q 的差异

用公式表示为： $D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$

直观解释：It represents the “information loss” or “extra cost” when approximating $P(x)$ using $Q(x)$.

特点：非负。值越小，说明 Q 越接近 P。两个分布相同的时候，值为 0 。

3. K-L散度可以拆解为交叉熵减去熵： $D_{KL}(P\|Q) = H(P,Q) - H(P)$

Logistic Regression Loss

Logistic 回归通常使用对数损失（Log-loss）作为目标函数

$\text{Loss} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i) \right]$

$\hat{p}_i = \sigma(w^T x_i + b)$：预测概率，其中 $\sigma(z) = \frac{1}{1+e^{-z}}$ 是 Sigmoid 函数。

[推导过程]

1. 假设分布：假设 $y \sim \text{Bernoulli}(\hat{p})$，其中 $\hat{p} = \sigma(w^T x + b)$。
2. 最大化似然函数：似然函数为：$L(w,b) = \prod_{i=1}^{N} \hat{p}_i^{y_i} (1-\hat{p}_i)^{1-y_i}$

   取对数得到对数似然：$\log L(w,b) = \sum_{i=1}^{N} \left[y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i) \right]$
3. 最大化对数似然等价于最小化负对数似然：$\text{Loss} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i) \right]$

[为什么用 Cross-Entropy 作为损失函数]

1. 概率预测的自然选择

- Logistic 回归的输出是概率分布（通过 Sigmoid 函数得到的概率值）。
- 交叉熵直接衡量预测概率 $p(x)$ 与真实分布 y 之间的差异，适合用于概率模型。

2. 对错误预测的惩罚更敏感

- 交叉熵对错误的概率预测（例如预测值离真实值很远）惩罚较大

3. 数学上的凸性

- 对于 Logistic 回归，使用交叉熵损失的优化问题是凸的，容易求解，适合梯度下降等优化算法

[如果用MSE是convex problem吗?]

不是，$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$

- 将 $\hat{y}_i = \sigma(z_i)$代入后，$(\sigma(w^T x_i) - y_i)^2$ 变成关于 w 的非线性函数。
- 对该函数求二阶导数（Hessian 矩阵）时，可能会出现负特征值，因此目标函数不是凸的。

SVM Loss

由两部分组成：Hinge Loss和正则化项 $\min_{w,b} \frac{1}{2}\|w\|_2^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i))$

$y \in \{-1, 1\}$，如果 $f(x) > 0$，预测为正类；如果 $f(x) < 0$，预测为负类。

熵

熵越大表示“不确定性越大”或“可能的结果越多”。定义：$H(P) = -\sum_{x \in X} P(x) \log P(x)$

情况 1：均匀分布（熵最大）

如果 $P(x)$ 是均匀分布，那么每个结果的概率相同，没有偏向性，不确定性最高，熵最大。例如：投掷一个公平的六面骰子，熵为 $H = -\sum_{i=1}^{6} \frac{1}{6} \log \frac{1}{6} = \log 6$

情况 2：确定性分布（熵为 0）

如果 $P(x)$ 中某个事件的概率为 1（其他事件概率为 0），则结果是完全确定的，熵为 0。例如：一个硬币的两面分别为 100% 和 0%，则没有不确定性 $H = -(1 \cdot \log 1 + 0 \cdot \log 0) = 0$

Decision Tree Loss

在决策树分裂节点时，优化目标是找到最佳特征及其分裂阈值，使得子节点的“纯度”最大化，即子节点的样本尽可能属于单一类别。常见优化目标：

1. 信息增益（Information Gain）：衡量分裂前后信息的不确定性减少量（基于熵）。

   $\text{Information Gain} = H(\text{parent}) - \sum_i \frac{N_i}{N} H(\text{child}_i)$
2. 基尼不纯度（Gini Impurity）：衡量节点样本中随机抽取两个样本属于不同类别的概率。

   $G = 1 - \sum_k p_k^2$

   p_k 是第 k 类的样本比例；分裂后，最小化加权平均基尼不纯度：$\text{Split Objective} = \sum_i \frac{N_i}{N} G(\text{child}_i)$
3. 方差减少（Reduction in Variance）（用于回归树）：衡量分裂前后目标值的方差减少量。

   $\text{Variance Reduction} = \text{Var}(\text{parent}) - \sum_i \frac{N_i}{N} \text{Var}(\text{child}_i)$

DL基础概念类

Bias Term

Bias term（偏置项）的作用是让神经网络的激活函数有灵活的 shift 能力，而不是始终被限制在原点附近。

如果没有偏置项，神经元的输出总是受到输入加权求和结果的直接影响，所有激活函数的决策边界都会固定经过原点（例如 $w^T x = 0$）。

偏置项相当于为网络增加了一种自由度，可以让模型学习到更加灵活的决策边界，从而更好地拟合数据。

Bias term 让网络不仅学“斜率”，还可以学“截距”，使模型表达能力更强。

Back Propagation

反向传播（Back Propagation）是训练神经网络的一种算法，用于通过逐层计算误差的梯度来更新网络的参数（权重和偏置）。

1. 前向传播：输入数据经过网络计算出预测值。
2. 误差计算：根据预测值与真实值计算损失（Loss）。
3. 误差反向传播：从输出层开始，将误差通过链式法则逐层“传回”，计算每层参数对损失的影响（梯度）。
4. 参数更新：利用梯度下降算法，用梯度调整每一层的参数，使损失减小。

总结：反向传播就是通过链式法则计算误差对参数的影响，再逐层优化参数，让模型更准确。

Back Propagation is an algorithm used to train neural networks by computing the gradient of the error layer by layer to update the network's parameters (weights and biases).

1. Forward pass: Input data is passed through the network to compute predictions.
2. Error computation: Loss is calculated based on the difference between predictions and true values.
3. Backward pass: Starting from the output layer, the error is "propagated back" layer by layer using the chain rule to compute gradients.

4. Parameter update: Gradients are used with gradient descent to adjust the parameters, reducing the loss.

Summary: Back Propagation calculates how the error affects the parameters and updates them layer by layer to make the model more accurate.

梯度消失和梯度爆炸

梯度消失

- 当梯度在反向传播中逐层变小，导致靠近输入层的参数几乎不更新，模型难以训练。
- 原因：激活函数（如Sigmoid或Tanh）在输入绝对值很大时，梯度接近0。

梯度爆炸

- 当梯度在反向传播中逐层变大，导致参数更新过大，训练不稳定或发散。
- 原因：网络层数太深，或者权重初始化较大。

解决方法

1. 梯度消失：
    - 使用激活函数如ReLU、Leaky ReLU。
    - 用更好的权重初始化方法（如Xavier初始化、He初始化）。
    - 使用归一化技术（如Batch Normalization）。
2. 梯度爆炸：
    - 梯度裁剪（Gradient Clipping）限制梯度的最大值。
    - 规范化权重（Weight Regularization）。

Gradient Vanishing

- Gradients become very small during backpropagation, causing parameters in earlier layers to barely update, making the model hard to train.
- Reason: Activation functions like Sigmoid or Tanh have near-zero gradients for large input values.

Gradient Explosion

- Gradients grow very large during backpropagation, leading to excessively large parameter updates and unstable or divergent training.
- Reason: Deep networks or large weight initializations.

Solutions

1. For Gradient Vanishing:
    - Use activation functions like ReLU or Leaky ReLU.
    - Employ better weight initialization (e.g., Xavier or He initialization).
    - Apply normalization techniques (e.g., Batch Normalization).
2. For Gradient Explosion:
    - Use Gradient Clipping to limit the maximum gradient value.
    - Regularize weights (e.g., Weight Regularization).

Weights 初始化

不能将神经网络的权重初始化为0，否则网络无法有效训练。

原因：

1. 对称性破坏问题（Symmetry Breaking）：
    - 如果所有权重都初始化为0，所有神经元计算出的值、梯度都相同。

    - 这会导致每一层的神经元学不到不同的特征，网络失去表达能力。
2. 梯度更新无差异：
    - 梯度下降算法会对所有权重进行相同的更新，导致所有权重始终相等。

正确做法：

1. 使用随机初始化：随机分布的权重可以打破对称性，让不同神经元学习不同的特征。
2. 标准方法：
    - Xavier初始化：适用于Sigmoid或Tanh激活函数。
    - He初始化：适用于ReLU激活函数。

Weights in a neural network should not be initialized to zero, as this prevents effective training.

Reasons:

1. Symmetry Breaking Problem:
    - If all weights are initialized to zero, all neurons in a layer will compute the same values and gradients.
    - This causes neurons to learn the same features, reducing the network's expressiveness.
2. No Gradient Differentiation:
    - Gradient descent would update all weights equally, keeping them the same throughout training.

Proper Approach:

1. Use random initialization: Randomly distributed weights break symmetry, allowing neurons to learn different features.
2. Standard methods:
    - Xavier Initialization: For Sigmoid or Tanh activation functions.
    - He Initialization: For ReLU activation functions.

DNN vs Logistic Regression

1. 模型结构：
    - Logistic 回归是单层线性模型，输出是线性权重的简单加权求和。
    - DNN 是多层非线性模型，可以学习复杂的非线性关系。
2. 特征提取：
    - Logistic 依赖于手工特征工程，适用于简单任务。
    - DNN 可以在隐藏层中自动提取数据的高阶特征，适合复杂任务（如图像、语音）。
3. 训练过程：
    - Logistic 回归用凸优化问题，容易收敛。
    - DNN 是非凸优化，训练更复杂，需大量数据和算力。
4. Model Structure:
    - Logistic Regression is a single-layer linear model where the output is a simple weighted sum of linear weights.
    - DNN is a multi-layer nonlinear model capable of learning complex nonlinear relationships.
5. Feature Extraction:
    - Logistic Regression relies on manual feature engineering and is suitable for simple tasks.
    - DNN can automatically extract high-level features in hidden layers, making it ideal for complex tasks (e.g., images, speech).
6. Training Process:
    - Logistic Regression uses convex optimization, making it easy to converge.

- DNN involves non-convex optimization, making training more complex and requiring large amounts of data and computational power.

Hyperparameter Tuning Methods

1. 网格搜索（Grid Search）：枚举所有超参数组合，逐一测试，适合参数少的情况。
   - 优点：如果采用较大的搜索范围以及较小的步长，很大概率找到全局最优值
   - 缺点：十分消耗计算资源和时间

   在实际应用中，一般会先使用较广的搜索范围和较大的步长，来寻找全局最优值可能的位置; 然后会逐渐缩小搜索范围和步长，来寻找更精确的最优值。
   - 优点：可以降低所需的时间和计算量
   - 缺点：由于目标函数一般是非凸的，所以很可能会错过全局最优值
2. 随机搜索（Random Search）：随机采样超参数空间，比网格搜索更高效。
   - 理论依据：如果样本点集足够大，那么通过随机采样也能大概率地找到全局最优值，或其近似值
   - 优点：一般会比网恪搜索要快一些
   - 缺点：和网格搜索的快速版一样，它的结果也是没法保证的
3. 贝叶斯优化（Bayesian Optimization）：用概率模型预测超参数性能，自动选择更优参数。
   - 步骤：根据分布选择采样点并测试目标函数值；根据新数据更新目标函数的分布（后验分布）；
     根据后验分布选择下一个最优采样点，继续迭代
   - 优点：相比网格搜索和随机搜索，充分利用历史信息，更高效地找到全局最优解。
   - 缺点：容易陷入局部最优，需要通过探索来规避这一问题。

预防 Overfitting 办法

- 使用更多的训练数据：减少噪声的影响
- 降低模型复杂度：在神经网络模型中减少网络层数、神经元个数
- 正则化：加入L1/L2正则化，限制权重过大。
- 集成学习：将多个模型集成在一起，降低单一模型过拟合风险
- 训练技巧：
  - Dropout：随机丢弃部分神经元，减少模型依赖。
  - 早停（Early Stopping）：在验证集性能不再提升时停止训练。
  - Batch Normalization：归一化中间层输出，稳定训练并提升泛化能力。

Dropout

Dropout是一种正则化方法，在训练时随机丢弃（设置为0）部分神经元，减少模型对特定神经元的依赖，从而防止过拟合。

[为什么能够防止over fit]

- 减少依赖：随机丢弃神经元迫使网络不同部分独立学习普遍特征，而不是噪音中的特定特征。
- 等效集成模型：相当于训练了多个子网络并取平均，从而降低过拟合风险。

[Dropout流程]

训练时：每次前向传播随机丢弃一部分神经元（置为0），丢弃概率为 p。

激活值被缩放为 $1/(1-p)$（防止总激活值变小）。

测试时：不再丢弃神经元，使用完整网络。无需缩放激活值。

Batch Normalization

Batch Normalization（BN）对每一层的每个维度的输入进行标准化

训练时：

- 对每一批次数据的每个特征计算均值和方差： $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
- 标准化：$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
- 通过可学习的参数进行缩放和平移： $y_i = \gamma \hat{x}_i + \beta$

测试时：使用训练中记录的全局均值和方差（而非当前批次的统计量）。

[为什么Batch Normalization 有什么效果]

- 缓解内部协变量偏移：标准化输入使后续层输入分布稳定，减少训练中梯度变化过大的问题。
- 加速收敛：通过稳定输入分布，允许更高的学习率，提高训练速度。
- 有轻微正则化作用：在训练时对小批量数据进行标准化，这些均值和方差有一定随机性，增加了噪声，能够防止过拟合。

- Reduces Internal Covariate Shift: Normalized inputs stabilize the distribution for subsequent layers, reducing large gradient changes during training.
- Speeds up Convergence: By stabilizing input distributions, BN enables higher learning rates, improving training speed.
- Provides Regularization: Normalizing over mini-batches introduces noise, which helps prevent overfitting.

Activation Functions

1. Sigmoid
   - Formula: $\sigma(x) = \frac{1}{1+e^{-x}}$.
   - Pros: Output range (0, 1), suitable for probability outputs.
   - Cons: Gradient vanishing issue; output not zero-centered.
2. Tanh
   - Formula: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$.
   - Pros: Output range (-1, 1), zero-centered.
   - Cons: Still suffers from gradient vanishing.
3. ReLU
   - Formula: $\mathrm{ReLU}(x) = \max(0, x)$.
   - Pros: Simple, fast convergence, mitigates gradient vanishing.
   - Cons: Neurons may "die" (gradient becomes 0 and stops updating).
4. Leaky ReLU
   - Formula: $\text{Leaky ReLU}(x) = \max(\alpha x, x)$, where $\alpha > 0$.
   - Pros: Allows negative gradients, prevents "dead" neurons.
   - Cons: Requires manual tuning of α.

Gradient Descent

Batch Gradient Descent：使用整个数据集计算梯度 $\nabla J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla J(\theta; x_i, y_i)$

- 优点：方向稳定。　缺点：计算开销大，内存占用高。
- 如果损失函数是凸的（例如二次函数或逻辑回归的交叉熵损失），BGD 总是可以找到全局最优解。

  如果损失函数是非凸的（例如神经网络中的复杂损失函数），BGD 可能陷入局部最优

Stochastic Gradient Descent（SGD）：使用一个样本计算梯度 $\nabla J(\theta) = \nabla J(\theta; x_i, y_i)$

- 优点：(1) 计算快，适合大规模数据; (2) 有助于跳出局部最优，因为梯度更新带有噪声。
- 缺点：(1) 更新方向不稳定，可能导致收敛缓慢或震荡；(2) 每次迭代的计算效率较低（GPU利用不足）

Mini-Batch Gradient Descent：只用一小部分数据计算梯度，32、64、128 等

- $\nabla J(\theta) = \frac{1}{|\text{Batch}|} \sum_{i \in \text{Batch}} \nabla J(\theta; x_i, y_i)$
- 特点：尤其适合 GPU，能并行处理 Mini-Batch

Optimizers

不同的优化器（如 Adam、SGD、Weight Decay Adam）都使用计算出的梯度来更新模型的权重参数，但它们在使用这些梯度的方式上有所不同。

1. 随机梯度下降（SGD）

- 基本的梯度下降方法，每次更新时使用一个小批量（mini-batch）数据的梯度。
- $\theta \leftarrow \theta - \eta \nabla_\theta J(\theta)$
- 其中，θ 是模型的权重； η 是学习率；$\nabla_\theta J(\theta)$ 是损失函数 $J(\theta)$ 对权重 θ 的梯度。

2. 动量梯度下降（SGD with Momentum）

- 为了加速收敛，尤其是在鞍点附近，动量梯度下降在更新时考虑了之前梯度的累积。

  To accelerate convergence, especially near saddle points, momentum gradient descent takes the accumulation of previous gradients into account during updates.

- $\theta \leftarrow \theta - \eta\, v_t$
- 其中， $v_t = \beta v_{t-1} + (1-\beta)\nabla_\theta J(\theta)$ ， v_t 是动量项，β 是动量因子。

3. RMSProp

- RMSProp makes the learning process more stable by adjusting the learning rate based on how large or small the gradients have been recently. If the gradients are small, it increases the learning rate to make faster progress. If the gradients are large, it decreases the learning rate to take smaller steps and avoid overshooting.
- $g_t = \nabla_\theta J(\theta_t) \qquad E[g^2]_t = \rho \cdot E[g^2]_{t-1} + (1-\rho) \cdot g_t^2 \qquad \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$
- 指数加权平均$E[g^2]_t$:
  - RMSProp 使用 ρ 对历史梯度平方进行加权平滑。
  - ρ 较大时（如 0.9），历史信息权重大，当前梯度对更新的影响较小。
  - $E[g^2]_t$ 使得每次更新时能够平滑掉梯度的大幅波动。
- 学习率调整 $\sqrt{E[g^2]_t + \epsilon}$:
  - 如果梯度的平方较大（说明梯度变化剧烈），分母会增大，从而降低学习率，避免过大的更新步长。
  - 如果梯度的平方较小（说明梯度较平缓），分母会变小，从而增大学习率，避免更新过慢。
- 极小值修正 ϵ:
  - 为了防止梯度的平方值为零时分母变成 0，加入一个极小的修正项ϵ，保证数值稳定性。

4. Adam（自适应矩估计）

- Adam 结合了动量和 RMSProp 方法，自适应地调整每个参数的学习率。
- $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$，$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$，$m_t$ 是一阶矩估计（类似于动量）， β_1 是衰减率。
- $v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$，$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$，$v_t$ 是二阶矩估计， β_2 是衰减率。
- $\theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}\epsilon}$

5. Weight Decay（权重衰减）

- Weight Decay 是一种正则化方法，通过在更新中添加一个权重衰减项，防止过拟合。
- Weight Decay 等效于在损失函数中加入一个 L2 正则化项：$J_{\text{reg}}(\theta) = J(\theta) + \frac{\lambda}{2}\|\theta\|^2$
- $\theta_{t+1} = \theta_t - \eta \cdot (\nabla J(\theta_t) + \lambda \cdot \theta_t)$　其中 λ 是权重衰减系数。

Learning Rate

Too large: Model fails to converge, may oscillate or even diverge in the loss function.

Too small: Slow convergence, longer training time, and potential trapping in local minima.

Plateau

定义：在优化过程中，损失函数的值在某些区域变化极小（接近平坦），导致梯度非常小，训练速度变慢。

影响：模型训练可能停滞，收敛速度显著降低。

解决方法：方向对但是幅度不够大，所以注重调整步长，使用动态调整学习率的优化器（如Adam、RMSprop）；增加 Batch Normalization

Definition: During optimization, the loss function has regions where it changes very little (almost flat), resulting in very small gradients and slow training.

Effect: Training progress stalls, significantly slowing convergence.

Solution: Use optimizers (Adam or RMSprop) to adaptively adjust learning rates; Batch Normalization

Saddle Point

定义：在优化过程中，梯度为零但并非局部最优，某些方向为下降（负曲率），而另一些方向为上升（正曲 率）。高维空间中鞍点比局部极小值多得多，尤其是在深度网络中。

影响：模型在鞍点区域可能陷入不必要的停滞，浪费计算资源。

解决方法：附近梯度方向很复杂，所以注重调整方向，使用带噪声的梯度方法（如SGD），动量优化（Momentum）

Definition: During optimization, gradients are zero, but the point is not a local minimum. Some directions have negative curvature (descending), while others have positive curvature (ascending). In high-dimensional spaces, saddle points are far more common than local minima, especially in deep networks.

Effect: The model can unnecessarily stagnate at saddle points, wasting computational resources.

Solutions：Use noisy gradient methods like SGD，Apply momentum optimization

Transfer Learning

迁移学习是利用在一个任务上训练好的模型的知识（如权重和特征），应用到另一个相关任务中。

[When Transfer Learning Makes Sense]

1. 目标任务数据少：目标任务缺乏足够的标注数据。
2. 目标数据分布不足：目标任务数据多样性不足，无法有效训练模型。
3. 目标任务相似：源任务和目标任务的领域特征相似。
4. 目标任务复杂但相关：需要利用预训练模型的通用特征。
5. 训练成本高：从零开始训练需要大量计算资源或时间。

ML模型类

3.1 Regression

基础假设是什么

Linearity, No Multicollinearity, Error Independence, Error Normality, Erro constant variance

Multicollinearity

Check: Correlation, Variance Inflation Factor, Factor Analysis

Impact: Hard to interpret contribution; Unstable coefficients; Overfit

Solution:

- Remove Redundant Variables: Drop highly correlated variables using correlation analysis or VIF.
- Regularization: Use L1 or L2 to shrink less important coefficients.
- Dimensionality Reduction: Apply PCA to combine correlated variables into uncorrelated components.
- Collect More Data: Increase data size and diversity to mitigate multicollinearity.

Regression Coefficient Explanation

A regression coefficient represents the change in the dependent variable (y) for a one-unit change in an independent variable (x), assuming all other variables are held constant.

Interaction Effect

The slope of one variable (x_1) depends on the level of another variable (x_2),

3.2 Clustering and EM:

K-Means Clustering

[Algorithm Steps]

1. Initialization: Randomly select kkk cluster centroids.
2. Assignment: Assign each data point to the nearest centroid.
3. Update: Recompute the centroids as the mean of all points in each cluster.
4. Repeat: Iterate steps 2 and 3 until convergence or stopping criteria are met.

[Convergence]

- Guaranteed to converge: The algorithm always reduces within-cluster variance (WCSS) at each step.
- Local Optimum: K-means may converge to a local optimum depending on the initial centroids.

[Stopping Criteria]

1. Centroids no longer change significantly between iterations.
2. Assignments of data points to clusters do not change.
3. Maximum number of iterations is reached.

EM算法

1. 随机初始化每个类的参数 (e.g., 每个分布的均值、方差和混合系数)。
2. 计算每个点属于每个类 (每个分布) 的概率（即责任值），这些概率表示点 x_i 对每个类的贡献。
3. 使用这些概率作为权重，重新计算每个类的参数。
4. 循环执行上述过程，直到收敛。

1. Randomly initialize the parameters for each class (e.g., the mean, variance, and mixture coefficient of each distribution).
2. Calculate the probability (responsibility) that each point belongs to each class (distribution). These probabilities represent the contribution of point xix_i to each class.
3. Use these probabilities as weights to recompute the parameters for each class.
4. Repeat this process iteratively until convergence.

GMM vs Kmeans

GMM（高斯混合模型）是一种基于概率的聚类方法，它假设数据是由多个高斯分布组成，通过估计这些分布的参数（均值、方差、权重）来进行聚类。

[GMM和K-Means的区别]

1. K-Means 输出的是硬分配，一个点属于一个类； GMM输出的是软分配，一个点属于每个类的概率
2. K-Means 中每个数据点都只被用来更新一个类的参数；GMM 中每个数据点会依据概率被切割，来更新每一个类的参数
3. K-Means 更适合明确分界清晰的球形簇；GMM 适合处理边界模糊或有重叠的分布，特别是在数据本身是概率生成时，GMM 更贴合数据结构。

[GMM和K-Means的联系]

K-Means 是 GMM 的特例，当：

1. 高斯分布的协方差矩阵为单位矩阵的倍数（即各向同性的球形分布）。
2. 簇的权重（π_k）相等。
3. 分配方式由软分配退化为硬分配（即每次分配点到概率最大的簇）。

GMM (Gaussian Mixture Model) is a probabilistic clustering method that assumes data is composed of multiple Gaussian distributions. It clusters data by estimating the parameters (mean, variance, and weight) of these distributions.

[Differences Between GMM and K-Means]

1. Assignment Method:
   - K-Means outputs hard assignments, where each point belongs to one specific cluster.
   - GMM outputs soft assignments, where each point is assigned a probability of belonging to each cluster.
2. Parameter Update:
   - In K-Means, each data point contributes to the parameter update of only one cluster.
   - In GMM, each data point is "split" across clusters based on probabilities and contributes to the parameter update of all clusters.
3. Cluster Shape Suitability:
   - K-Means is better suited for clearly separated, spherical clusters.
   - GMM can handle clusters with overlapping boundaries or ambiguous membership and is especially effective when the data is probabilistically generated.

[Connections Between GMM and K-Means]

K-Means can be considered a special case of GMM under the following conditions:

1. Covariance Matrix Assumption:
   - The covariance matrix of each Gaussian distribution is a scaled identity matrix (i.e., isotropic spherical distribution).
2. Equal Cluster Weights:
   - The weights of all clusters are equal.

3. Hard Assignment:
    - The soft assignment of GMM is reduced to hard assignment, meaning each point is assigned to the cluster with the highest probability.

3.3 Decision Tree

Node Split

Regression: Minimizes variance.

Classification: Maximizes purity using Gini or Entropy.

Overfit Prevent

Simplify the tree structure (e.g., pruning, depth limits).

Constrain splits (e.g., minimum samples, regularization using max_leaf_nodes).

Combine multiple trees with ensemble methods.

3.4 Ensemble Learning

Bagging vs Boosting

Bagging (Bootstrap Aggregating): Random Forest

- 核心思想：并行训练多个独立的弱学习器，通过综合(分类：投票，回归：平均)多个模型的结果降低方差。
- 优点：降低方差，减少过拟合。

Boosting: Gradient Boosting Decision Tree

- 核心思想：顺序训练多个弱学习器，每一个都集中优化前一个的错误，最终多个模型结果按权重加和。
- 回归问题：每个模型的任务是预测前一个模型未能捕捉到的残差 $r_i = y_i - \hat{y}_i$

  分类问题：如果某些样本被前一个模型错误分类，这些样本会“权重更高”或更重要。
- 优点：降低偏差，提高模型精度。

3.5 Generative Model

Discrimitive vs Generative

Generative 模型

- 核心目标：学习输入 X 和输出 Y 的联合分布 $P(X, Y)$
- 更容易 Underfitting：
    1. 建模复杂：尤其是在高维或复杂数据中，难以捕获完整分布。
    2. 模型简单化倾向：生成模型可能倾向于简单化处理分布（例如高斯假设）
    3. 数据需求高：生成模型通常需要更多数据才能学到准确的分布，在数据量不足时效果较差。

Discriminative 模型的特点

- 核心目标：直接学习条件分布 $P(Y|X)$，即学习输入特征 X 和输出 Y 的关系，而不关心特征 X 的分布。
- 原因更容易 Overfitting：
    1. 模型容易过于复杂：对训练数据的 decision boundary 拟合过度。
    2. 对噪声敏感：忽略了输入特征的分布，会将训练数据中的噪声误认为是有效特征。
    3. 高灵活性：如深度学习、SVM，很能拟合数据，很容易在训练集上表现很好但泛化能力差。

Naïve Bayes vs LDA vs QDA

$\hat{Y} = \arg\max_Y P(Y|X) = \arg\max_Y P(X|Y) \cdot P(Y)$

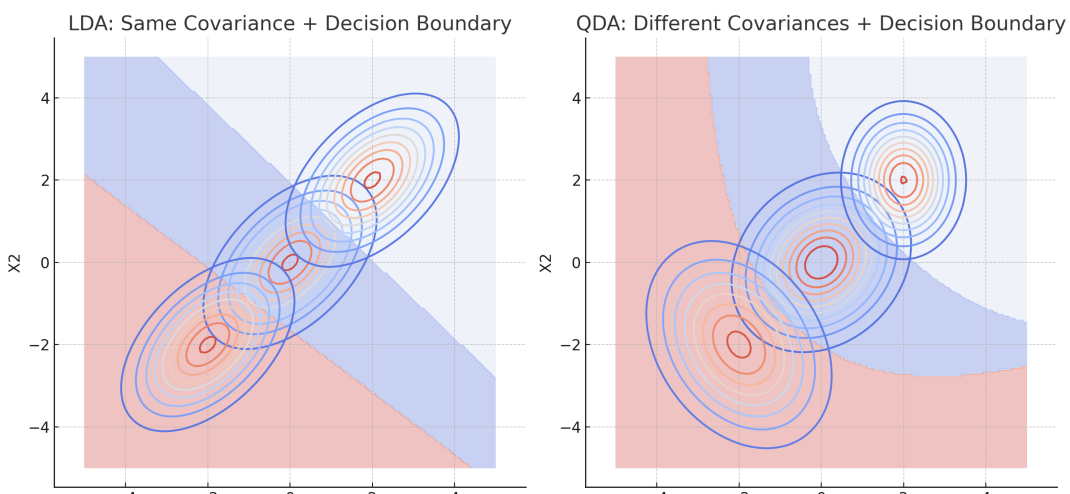
三个模型都是 choose the class with the highest posterior probability，训练都是为了获得 $P(X|Y)$

Naïve Bayes：假设 features are independent.

LDA： 假设各个类别的 features 服从多元高斯分布，并且所有类别共用同一个协方差矩阵 Σ

所以决策边界是线性的

QDA： 允许每个类别有自己的协方差矩阵，决策边界是二次的。



3.6 Classification

Logistic Regression vs SVM

soft vs hard classification; linear vs non-linear devision boundary; minimize log loss vs maximize margin

SVM

SVM (Support Vector Machine) is a classification algorithm that finds a hyperplane to separate data points while maximizing the margin between classes, improving robustness.

How non-linearity introduced: using a kernel function to map data into a higher-dimensional space. The kernel function also computes the dot product in this space, making data linearly separable.

SVM 本质是将每个 feature vector 映射到某个高维甚至无限维的特征空间，在这个空间中寻找一个线性超平面（decision boundary）来分割数据。然而，SVM 求解过程中只需要样本间的点积（内积）矩阵，而不需要显式地进行高维映射。所以，SVM 不直接计算高维特征，而是 使用了 Kernel Trick，即通过定义一个核函数（kernel function），直接计算两个向量在高维空间中的点积，跳过显式的映射过程，从而高效地解决非线性问题。

Kernal Methods

Kernel Methods use a kernel function $K(x_i, x_j)$ to compute the relationship (dot product) between two data points in a high-dimensional space without explicitly mapping the data into that space.

[Why Use Kernel Methods]

1. Solve Nonlinear Problems: Kernel functions implicitly map data to a high-dimensional space, where linearly inseparable data becomes linearly separable.
2. Efficiency: Avoids explicitly constructing high-dimensional features, saving computational cost while retaining the power of high-dimensional representation.
3. Flexibility: Supports various kernel functions (e.g., linear kernel, polynomial kernel, Gaussian/RBF kernel) to handle different types of nonlinear problems.

[Kernal Functions]

1. Polynomial kernel: $K(x_i, x_j) = (x_i^T x_j + c)^d$

- The mapping function $\phi(x)$ expands the original features into all possible polynomial combinations.
- For example, if $x \in \mathbb{R}^2$ and $d = 2$, the mapping would be: $\phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1]$

2. Gaussian (RBF) Kernel: $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

- The Gaussian kernel corresponds to a mapping into an infinite-dimensional space. The exact form of the mapping function $\phi(x)$ is typically not explicitly known.
- However, it can be thought of as: $\phi(x) = [\exp(-\|x - x_1\|^2), \exp(-\|x - x_2\|^2), \ldots]$

3. Sigmoid kernel: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c)$

- The sigmoid kernel is similar to the activation function in neural networks, but its corresponding mapping function $\phi(x)$ is not easily expressed explicitly.
- Under certain conditions, it is equivalent to the hidden layer output of a shallow neural network.

3.7 Unsupervised Models

Principal Component Analysis

PCA is a dimensionality reduction technique. It aims to Identify the directions (principal components) where the data varies the most and project the data onto these directions.

K-Nearest Neighbors

KNN is a simple classification or regression algorithm that conduct class (value) prediction based on the majority class (or average value) of its neighbors.

1. For a given point, find the K closest points (neighbors) in the training data.
2. Classification: The predicted class is determined by the majority class among the neighbors. Regression: The predicted value is the average of the neighbors' values

数据处理类

High-dim Classification

问题：

1. 维度灾难：高维数据使得点之间的距离不明显，分类边界难以确定。
2. 过拟合：特征多而样本少，模型容易记住训练数据而失去泛化能力。
3. 计算复杂度高：高维数据增加训练和推理的时间成本。

解决方法：

1. 降维：
    - 使用 PCA 或 t-SNE/UMAP 提取主要特征。
    - 特征选择：通过Lasso回归或树模型的重要性选择特征。
2. 正则化：
    - 使用 L1正则化（Lasso） 或 L2正则化（Ridge），减少不必要特征的权重，防止过拟合。
3. 增加样本量：
    - 用数据增强技术生成更多样本，缓解样本不足问题。
4. 选择高维友好算法：
    - 随机森林、XGBoost：对高维数据鲁棒。
    - 线性模型（如正则化后的Logistic回归）：高效处理高维特征。
    - 核方法（SVM with kernel）：在高维数据中表现良好。

Missing Data

缺失值比例低：直接删除。

缺失值比例适中：均值/回归插值 (使用其他特征预测缺失值 by 线性回归、KNN)。

缺失值比例高：多重插补（Multiple Imputation）或模型预测 (如深度学习、随机森林)。

Feature Selection

1. Remove Highly Correlated Variables: Use correlation analysis or Variance Inflation Factor (VIF)
2. Stepwise Method to Assess Variable Contribution:

- Forward Selection: Start with an empty feature set and iteratively add the feature that improves model performance the most.
- Backward Elimination: Start with all features and iteratively remove the feature that contributes the least to model performance.

3. Automatic Feature Selection by Models:

- Regularization: L1 regularization (Lasso)
- Tree-Based Models: Leverage feature importance from models like Random Forest or XGBoost

Feature Interaction

手动添加交互项

用模型自动学习交互：树模型（如随机森林、XGBoost）或神经网络

用模型筛选交互：因子分解机（FMs）或 DeepFM

项目经验类

训练表现好但现实不好的原因

1. 训练和现实数据分布不一致：
    - 原因：现实中的数据特征与训练数据分布差异较大（如采样偏差、季节性变化）。
    - 解决：检查数据分布，用迁移学习或持续学习更新模型。
2. 模型过拟合训练数据：
    - 原因：模型学到了训练数据中的噪声或特定模式，而不是泛化能力。
    - 解决：正则化、增加训练数据、使用简单模型。
3. 模型假设与现实情况不匹配：
    - 原因：现实中的复杂关系超出了模型假设的范围（如非线性模型使用线性假设）。
    - 解决：使用更复杂或更灵活的模型（如树模型或神经网络）。

生产和开发时发生Data shift

[检测]

1. 基于统计检测：(1) 比较特征（如均值、方差、最大/最小值）(2) 比较分布：Kullback-Leibler (KL) 散度
2. 基于模型检测：

    (1) Train-Test Split检测：将训练数据标记为"0"，生产数据标记为"1"，训练一个分类模型。如果分类模型能准确区分两者，说明数据分布有明显偏移。

    (2) 使用分类模型分析哪些特征最显著地导致分布偏移。

[补救]

1. 特征 X 偏移：

- 重新加权：对生产数据中的样本进行加权，使其匹配训练数据的特征分布。
- 更新模型：用包含生产数据的新数据重新训练或微调模型。

2. 目标 Y 偏移：

- 目标分布校正：对模型预测结果进行后处理 Post Processing（如调整预测概率）。
- 更新模型：用生产数据中真实标签重新训练模型。

3. 特征与目标之间的关系发生变化

- 模型微调：用生产数据中的新样本及其标签重新训练或微调模型。

Loss 成 Inf 或 NaN

1. Inf 和 NaN 的含义

- NaN (Not a Number)：表示在计算过程中出现了非法操作，比如除以零或对负数取平方根等。NaN 通常意味着出现了数学上未定义的行为，而不一定与溢出直接相关。
- Inf (Infinity)：表示一个数值超过了计算机可以表示的最大值（即正无穷大 `Inf` 或负无穷大 `-Inf`），通常由溢出导致。

2. Loss 变成 Inf 或 NaN 的常见原因

(1) 输入数据或标签异常

- 原因1：数据中存在异常值（如极大的值、零或负数）可能导致数值计算问题。例如：
  - 在对数运算中，输入为负数或零会导致 NaN（如 `log(0)` 或 `log(负数)`）。
  - 在平方或指数运算中，输入为极大值可能导致 Inf。
- 原因2：输入数据没有进行适当的归一化或标准化，导致输入特征范围过大（或过小）。
  - 特征值范围的异常会通过网络传播放大，导致模型计算不稳定。

(2) 初始化权重不当

- 原因：如果模型权重初始化得太大，第一步前向传播的输出可能会出现极大值，从而导致指数运算（如 softmax）溢出。
- 后果：计算时出现 Inf 或 NaN，尤其在激活函数（如 ReLU、Sigmoid）或损失函数中。
- 解决：用 Xavier 初始化或 He 初始化

(3) 激活函数导致溢出

- 原因：某些激活函数（如 Sigmoid、Tanh）在输入值绝对值过大时会接近饱和，或可能导致溢出。例如：
  - Sigmoid 的公式 `1 / (1 + exp(-x))` 中，如果 `x` 的绝对值过大，`exp(-x)` 会出现溢出，导致数值不稳定。
  - ReLU 激活函数可能导致梯度爆炸（Gradient Explosion）问题。

(4) 损失函数的数值不稳定

- 原因：损失函数本身可能存在数值不稳定性，例如：
  - 在交叉熵损失中，如果 softmax 的概率接近零，取对数（`log(softmax)`）会导致 `log(0)`，从而得到 NaN。
  - 在 MSE 损失中，如果预测值过大，平方操作可能导致溢出。

(5) 学习率过大

- 原因：过大的学习率会导致权重更新幅度过大，从而使模型预测值（如 logits 或输出）变得非常大或发散。
- 后果：`Loss` 可能变成 `Inf` 或 `NaN`，尤其是在对数操作或指数操作（如 softmax）中。

(6) 梯度爆炸

- 原因：在梯度计算过程中，如果梯度值逐步累积到非常大的数值，会导致参数更新幅度过大。

- 后果：权重可能变成极大值，进一步导致前向传播的输出过大，最终损失变成 Inf 或 NaN。
- 措施：对梯度值进行裁剪（Gradient Clipping），限制梯度的绝对值不超过某个阈值，防止梯度爆炸。

(7) 数值精度问题

- 原因：浮点数计算的精度有限（尤其是 32 位浮点数），在累积或极端计算时可能出现精度丢失或溢出问题。
- 后果：数值精度问题会进一步放大训练过程的不稳定性，最终导致 Inf 或 NaN。

Limited Label Data

Data Augmentation，Transfer Learning (Fine-tuning)， Self-Supervised Learning，

Semi-Supervised Learning，Active Learning，Few-shot Learning

- Active Learning：在模型训练过程中选择最有信息价值的样本，让人工优先标注这些样本。例如：选择分类概率最不确定的样本（接近决策边界）。
- Few-shot Learning：使用专门的算法（如元学习、Prototypical Networks）在少量标注数据上进行训练

NLP/RNN相关

RNN: Captures short-term dependencies but struggles with long-term dependencies.

- Limitation: vanishing/exploding gradients, making it hard to capture long-term dependencies in sequences.
- Solution: Gated Recurrent Unit, LSTM

LSTM: Designed to handle long-term dependencies by solving the vanishing gradient problem.

- LSTM: forget gate, input gate, output gate

Attention: a mechanism that allows a model to focus on the most relevant parts of input data when making predictions.

- [Why] It helps capture long-range dependencies, improves interpretability, and reduces the limitations of fixed-length context in RNNs/LSTMs.