LN 2. 监督学习 (Supervised Learning)

李钦宾

先进智能计算与系统团队

邮箱: *qinbin@hust.edu.cn*

2025 年 02 月

本章目录

本章目录

定义

首先，给出监督学习的形式化描述。
机器学习的数据集由输入对 $(\mathbf{x}, y)$ 来表示，其中 $\mathbf{x} \in \mathcal{R}^d$ 表示输入数据，y 表示对应的分类标签。整个训练数据集的表示形式如下：

$$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subseteq \mathcal{R}^d \times \mathcal{C}$$

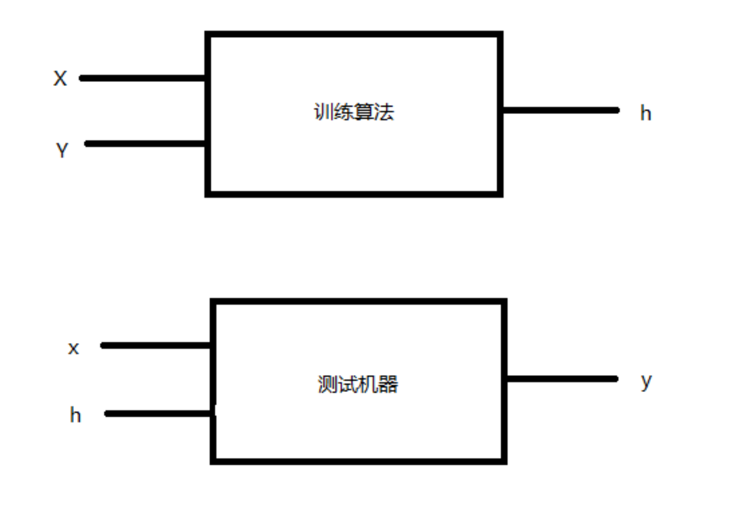
其中：

- $\mathcal{R}^d$ 表示 d 维特征空间 (d-dimensional feature space)
- $\mathbf{x}_i$ 表示第 i 个样本 (sample) 的输入向量
- y_i 表示第 i 个样本的标签 (label)
- $\mathcal{C}$ 表示数据集的标签空间

数据点 $(\mathbf{x}_i, y_i)$ 服从某个未知的分布 $\mathcal{P}(X, Y)$。我们希望通过学习，最终可以找到一个函数 h，对于新的输入数据 $(\mathbf{x}, y) \sim \mathcal{P}$，有比较高的置信度满足 $h(\mathbf{x}) = y$ (或者 $h(\mathbf{x}) \approx y$)。

定义

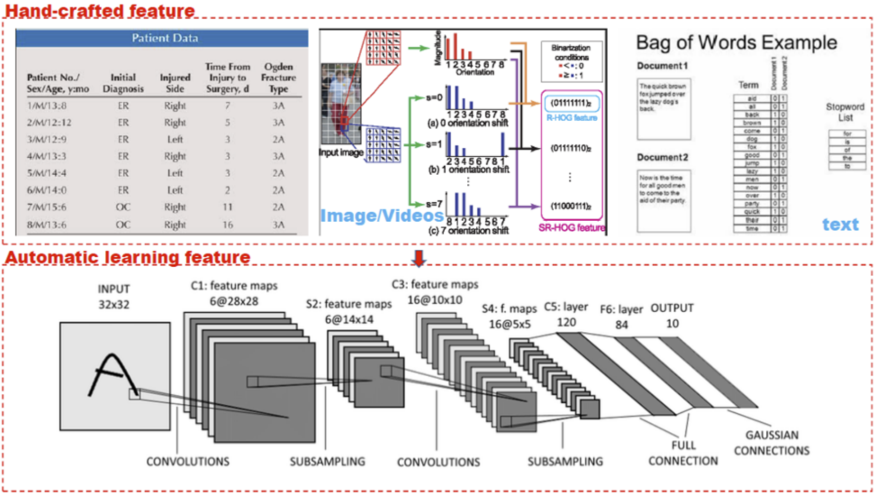
监督学习 (Supervised Learning)



几个标签空间 $\mathcal{C}$ 的例子:

二分类问题	$\mathcal{C}=\{0,1\}$ 或 $\mathcal{C}=\{-1,+1\}$.	例如垃圾邮件分类问题. 一个邮件要么是垃圾邮件 (-1), 要么不是垃圾邮件 $(+1)$
多分类问题	$\mathcal{C}=\{1,2,\cdots,K\}$ $(K\geq 2)$.	例如人脸识别. 一个人只能是 K 个身份里的某一个 (e.g., 1=" 张三", 2=" 李四", 等)
回归问题	$\mathcal{C}=\mathbb{R}$	例如预测未来的温度或是一个人的身高

特征空间举例



Patient No./ Sex/Age, y:mo	Initial Diagnosis	Injured Side	Time From Injury to Surgery, d	Ogden Fracture Type
1/M/13:8	ER	Right	7	3A
2/M/12:12	ER	Right	5	3A
3/M/12:9	ER	Left	3	2A
4/M/13:3	ER	Right	3	3A
5/M/14:4	ER	Left	3	2A
6/M/14:0	ER	Left	2	2A
7/M/15:6	OC	Right	11	2A
8/M/13:6	OC	Right	16	3A

本章目录

什么是损失函数

损失函数的核心非常简单：它是一种评估算法对数据集建模效果的方法。如果预测完全错误，你的损失函数将输出一个较大的值。如果预测很好，它将输出一个较小的值。当你改变算法的一部分，试图改进你的模型时，你的损失函数会告诉你是否有效。

事实上，可以自己设计基本损失函数来进一步解释它是如何工作的。对于所做的每个预测，损失函数将简单地测量预测值与实际值之间的绝对差值。

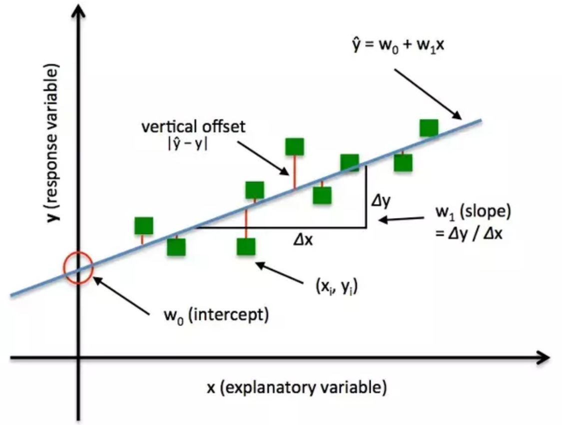
什么是损失函数——实例 1

如果我们试图预测一些某城市的公寓租金有多贵，下面是一些实际情况:

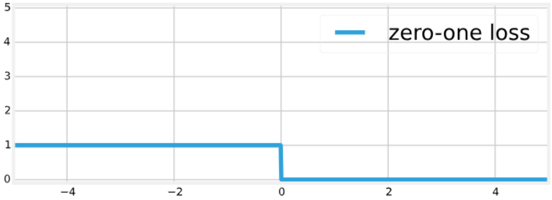
Our Predictions	Actual Values	Our Total Loss
Harlem: $1,000 SoHo: $2,000 West Village: $3,000	Harlem: $1,000 SoHo: $2,000 West Village: $3,000	0 (we got them all right!)
Harlem: $500 SoHo: $2,000 West Village: $3,000		500 (we were off by $500 in Harlem)
Harlem: $500 SoHo: $1,500 West Village: $4,000		2000 (we were off by $500 in Harlem, $500 in SoHo, and $1,000 in the West Village)

什么是损失函数

注意，在我们定义的损失函数中，预测过高或过低并不重要。重要的是有多不正确，这是和方向无关的。这并不是所有损失函数的特征。事实上，你的损失函数会根据你应用机器学习的场景和上下文而显著变化。在你的项目中，如果猜得太高可能比猜得太低更糟糕，那么你选择的损失函数必须反映这一点。



0-1 损失（Zero-one loss）



0-1 损失（Zero-one loss）

形式上，0-1 损失可以表述为:

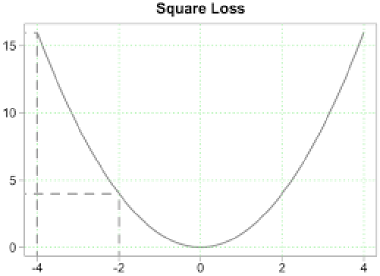
$$\mathcal{L}_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta_{h(\mathbf{x}_i)\neq y_i}, \text{ where } \delta_{h(\mathbf{x}_i)\neq y_i} = \begin{cases} 1, & \text{if } h(\mathbf{x}_i) \neq y_i \\ 0, & \text{o.w.} \end{cases}$$

该损失函数返回数据集 D 上的错误率。分类器若错误分类一个样本，则损失值加 1，而正确分类的样本损失值加 0。

0-1 的损失对每个错误分类的点施加相同的惩罚，因此“错误离谱”的点 (即边界点) 不会受到太多关注，这在直观上是不适宜的。

此外，0-1 损失为**不连续和非凸**的，优化的难度较大。

均方损失（Squared loss）



均方损失函数通常用于回归任务。形式上损失的平方是:

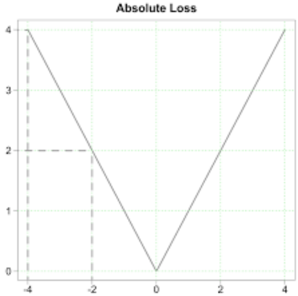
$$\mathcal{L}_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2.$$

均方损失函数有两个效果:

- 1) 计算得到的损失值总是非负的;
- 2) 计算得到的损失值与预测错误差的绝对值呈平方增长关系

缺点: 如果一个预测非常接近正确，如 $|h(\mathbf{x}_i) - y_i| = 0.001$，平方将是微小的，很少会注意到这个示例 (example)，以获得零误差。

（绝对值损失）Absolute Loss



与均方损失类似，绝对损失函数也通常用于回归设置。它会受到 $|h(\mathbf{x}_i) - y_i|$ 的惩罚。由于所遭受的损失与错误预测呈线性增长，因此更适合于有噪声的数据 (当一些错误预测是不可避免的，不应该主导损失)。如果给定一个输入 $\mathbf{x}$，标签 y 是概率分布 $P(y|\mathbf{x})$，那么将绝对损失最小化的最佳预测是预测**中值**，即 $h(\mathbf{x}) = \mathrm{median}_{P(y|\mathbf{x})}[y]$。

形式上，绝对损失可以表示为:

$$\mathcal{L}_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n}|h(\mathbf{x}_i) - y_i|.$$

本章目录

泛化（Generalization）

给定一个损失函数，我们可以尝试找到使损失最小化的函数 h:

$$h = \mathrm{argmin}_{h \in \mathcal{H}} \mathcal{L}(h)$$

机器学习的很大一部分内容都集中在这个问题上，即如何有效地最小化。

如果你发现一个函数 $h(\cdot)$ 对你的数据集 $\mathcal{D}$ 损失很小，如何能确定它对不在 $\mathcal{D}$ 中的示例，仍然能得到正确的结果?

反例 1（Bad example 1）

Bad example 1：死记硬背型，Memorizer $h(\cdot)$

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ 0, & \text{o.w.} \end{cases}$$

对于这个 $h(\cdot)$，我们在训练数据 D 上得到 0% 的错误，但是在 D 之外的样本上做得很糟糕，也就是说，这个函数存在严重的过拟合问题。

反例 2（Bad example 2）

Bad example 2: 穷举型

是否可以枚举所有的 $h \in \mathcal{H}$，并从中选出最好的?

枚举的代价过高，或不可能枚举所有。

反例 2（Bad example 2）

Bad example 2: 穷举型
是否可以枚举所有的 $h \in \mathcal{H}$，并从中选出最好的?

枚举的代价过高，或不可能枚举所有。

反例 3 （Bad example 3）

Bad example 3: 死记硬背 + 乱猜型

$$h(x) = \begin{cases} y_i, & \text{if } \exists(\mathbf{x}_i, y_i) \in D, \text{ s.t., } \mathbf{x} = \mathbf{x}_i, \\ random\{y_1, ..., y_k\} & \text{o.w.} \end{cases}$$

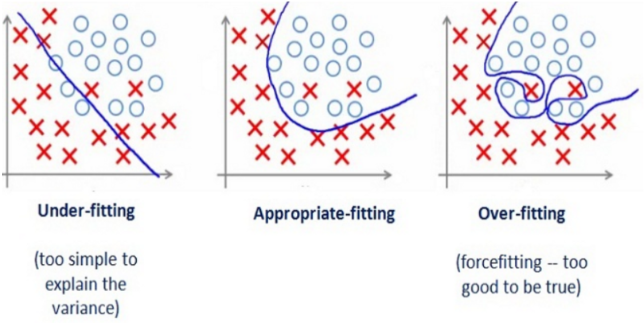
对于这个 $h(\cdot)$，我们在训练数据 $\mathcal{D}$ 上得到 0% 错误，但对于不在 $\mathcal{D}$ 中的样本仍然做得不好，即这个函数存在过拟合问题。

什么是过拟合?

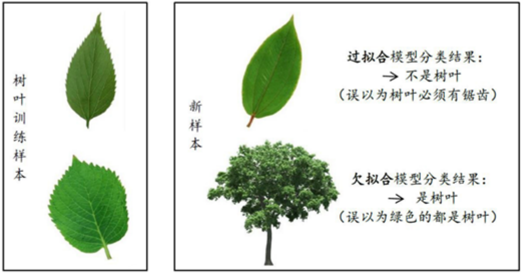
每当使用数据集来预测或分类问题时，我们一般是在训练集上训练模型，然后在测试集上测试模型的准确性。如果准确率令人满意，可以通过增加或减少数据特征，或特征选择，或在机器学习模型中应用特征工程，来进一步提高数据集预测的准确率。

但有时我们的模型可能会给出较差的结果。

我们的模型表现不佳可能是因为模型过于简单而无法描述目标，也可能是模型太复杂而无法展示目标。



什么是过拟合（Overfitting）？

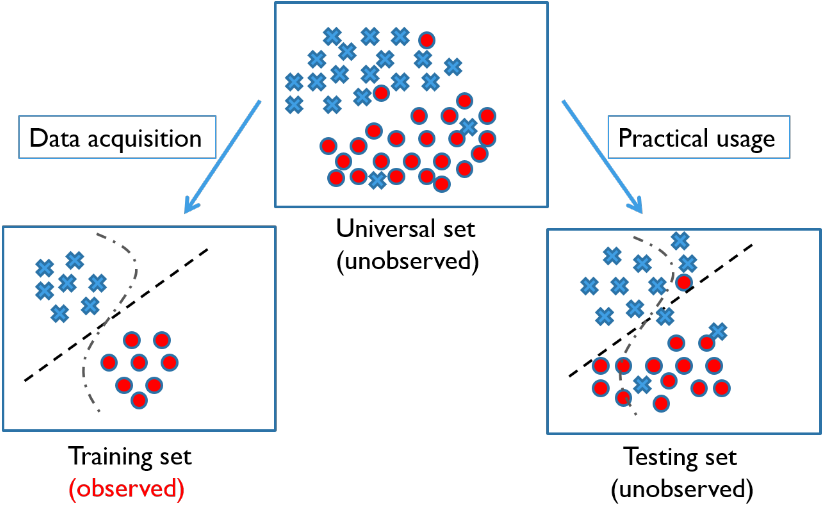


过拟合、欠拟合的直观类比

过拟合：学习器把训练样本本身特点当做所有潜在样本都会具有的一般性质.
欠拟合：训练样本的一般性质尚未被学习器学好.

本章目录

训练与测试

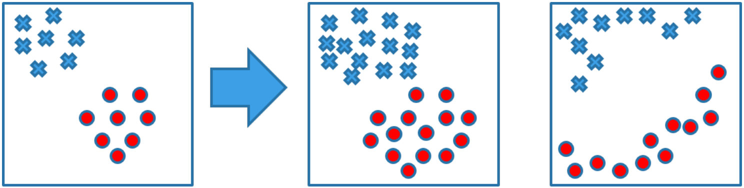

Data acquisition
Practical usage
Universal set
(unobserved)
Training set
(observed)
Testing set
(unobserved)

训练与测试

- 训练是使模型能够学习的过程

没有免费的午餐 (No free lunch rule)

- 训练集和测试集来自同一个分布
- 需要做一些假设 (Assumptions) 或偏好 (Bias)



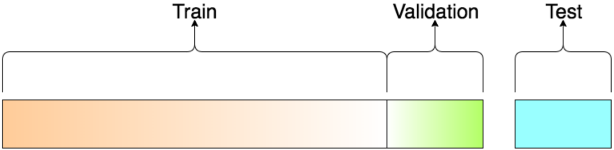
训练 / 测试划分

D 为包含 n 个样本的数据集，$D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.

为了解决过拟合问题，通常将 D 分成三个子集:

D_{TR} 为训练数据，D_{VA} 为验证数据，D_{TE} 为测试数据。

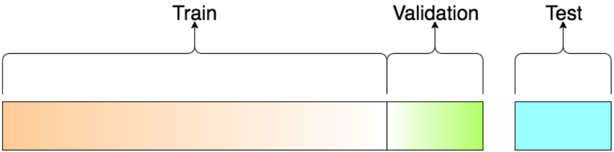
通常，它们被分成 80%、10% 和 10% 的比例。然后，根据 D_{TR} 选择 $h(\cdot)$，并在 D_{TE} 上计算 $h(\cdot)$。



测试: 为什么我们需要 D_{VA}?

训练 / 测试划分

D 为包含 n 个样本的数据集，$D = (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.
为了解决过拟合问题，通常将 D 分成三个子集:
D_{TR} 为训练数据，D_{VA} 为验证数据，D_{TE} 为测试数据。
通常，它们被分成 80%、10% 和 10% 的比例。然后，根据 D_{TR} 选择 $h(\cdot)$，并在 D_{TE} 上计算 $h(\cdot)$。



Q: 为什么我们需要 D_{VA}?

D_{VA} 用于检查从 D_{TR} 获得的 $h(\cdot)$ 是否存在过拟合问题。$h(\cdot)$ 将需要在 D_{VA} 上验证，如果损失太大，$h(\cdot)$ 将根据 D_{TR} 进行修改，并在 D_{VA} 上再次验证。
这个过程将反复执行，直到它在 $D_{mathmVA}$ 损失足够小。
D_{TR} 和 D_{VA} 的大小之间的权衡:
较大的 D_{TR} 的训练结果会更好，但如果 D_{VA} 更大，验证将更可靠 (噪音更小)。

如何分割数据?

当你在 **Train**，**Validation**，**Test** 中分割数据时，必须非常小心。

测试集必须模拟真实的测试场景，即你想要模拟在现实生活中遇到的设置。例如，如果想训练一个电子邮件垃圾邮件过滤器，可以用过去的数据训练一个系统来预测未来的电子邮件是否为垃圾邮件。

1) **按时间划分**：按时间先后顺序来分割训练/测试集是非常重要的。这样你就可以严格地从过去预测未来。

一般来说，如果数据中有时间的因素，则必须将其按时间分割。

2) **均匀随机**：如果没有时间的因素，通常最好是均匀随机分割。

绝对不要按字母顺序或特征值分割。

当 (且通常仅当) 数据为独立同分布（*i.i.d.*），采用均匀随机。

测试误差近似泛化损失

用测试误差/测试损失近似真实的泛化误差/泛化损失。

如何分割数据?

- 留出法：直接将数据集划分为两个互斥集合
- 交叉验证法：将数据集分层采样划分为 k 个大小相似的互斥子集，每次用 k-1 个子集的并集作为训练集，余下的子集作为测试集，最终返回 k 个测试结果的均值，k 最常用的取值是 10.
- 自助法：对数据集 D 有放回采样 m 次得到训练集 D'，剩余的作为测试集。在数据集较小、难以有效划分训练/测试集时很有用.

混淆矩阵

混淆矩阵的列代表模型预测的标签，行表示样本实际的标签。二分类时候，如果将正预测记作 P，负预测记作 N，分别对应于样本的实际标签予以逻辑判断正确 T 和错误 F，可以生成 TP,FP,TN,FN 四个标签，如下表所示，混淆矩阵有助于研究者观察模型预测各类别的能力，并据此分析潜在的可能原因和模型性能提升方法。

	正预测	负预测
正样本	TP	FN
负样本	FP	TN

查准率 P(precision) 与查全率 R(recall)

$$\text{查准：宁可漏网，不可错选} \qquad P = \frac{TP}{TP + FP} \tag{1}$$

$$\text{查全：宁可错选，不可漏网} \qquad R = \frac{TP}{TP + FN} \tag{2}$$

本章目录

我们通过最小化损失来训练分类器:

$$\text{学习: } h^*(\cdot) = \operatorname{argmin}_{h(\cdot)\in\mathcal{H}} \frac{1}{|D_{\mathrm{TR}}|} \sum_{(\mathbf{x},y)\in D_{\mathrm{TR}}} \ell(\mathbf{x}, y|h(\cdot)),$$

其中 $\mathcal{H}$ 是假设类 (即，所有可能的分类器 $h(\cdot)$ 的集合)。我们的目标是找到一个假设 h，它在过去/已知数据上均表现良好。
根据测试损失来评估分类器:

$$\text{评估: } \epsilon_{\mathrm{TE}} = \frac{1}{|D_{TE}|} \sum_{(\mathbf{x},y)\in D_{\mathrm{TE}}} \ell(\mathbf{x}, y|h^*(\cdot)).$$

如果样本从相同的分布 $\mathcal{P}$ 中抽取，则测试损失是真实泛化损失的无偏估计:

$$\text{泛化: } \epsilon = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{P}}[\ell(\mathbf{x}, y|h^*(\cdot))].$$

本章小结

问题

为什么 $\epsilon_{\mathrm{TE}} \to \epsilon$ as $|D_{\mathrm{TE}}| \to +\infty$?

回答

????

本章小结

问题

为什么 $\epsilon_{\mathrm{TE}} \to \epsilon$ as $|D_{\mathrm{TE}}| \to +\infty$?

回答

这是由于弱大数定律。由该定律，从分布中提取的数据的经验平均值会收敛到其真实的平均值。

本章小结

没有免费的午餐

每个 ML 算法都必须对该选择哪个假设类 $\mathcal{H}$ 进行假设。这个选择取决于数据，并编码关于数据集/分布 $\mathcal{P}$ 的**你的假设**。显然，对于任何问题，没有一个完美的 $\mathcal{H}$。

例子

假设 $(\mathbf{x}_1, y_1) = (1, 1)$，$(\mathbf{x}_2, y_2) = (2, 2)$，$(\mathbf{x}_3, y_3) = (3, 3)$，$(\mathbf{x}_4, y_4) = (4, 4)$，和 $(\mathbf{x}_5, y_5) = (5, 5)$.

问题

如果 $\mathbf{x} = 2.5$，y 的值是多少?

回答

????

本章小结

没有免费的午餐

每个 ML 算法都必须对该选择哪个假设类 $\mathcal{H}$ 进行假设。这个选择取决于数据，并编码关于数据集/分布 $\mathcal{P}$ 的**你的假设**。显然，对于任何问题，没有一个完美的 $\mathcal{H}$。

例子

假设 $(\mathbf{x}_1, y_1) = (1, 1)$，$(\mathbf{x}_2, y_2) = (2, 2)$，$(\mathbf{x}_3, y_3) = (3, 3)$，$(\mathbf{x}_4, y_4) = (4, 4)$，和 $(\mathbf{x}_5, y_5) = (5, 5)$.

问题

如果 $\mathbf{x} = 2.5$，y 的值是多少?

回答

没有假设是完全不可能知道答案的。ML 算法最常见的假设是，要逼近的函数是局部光滑的。

The End