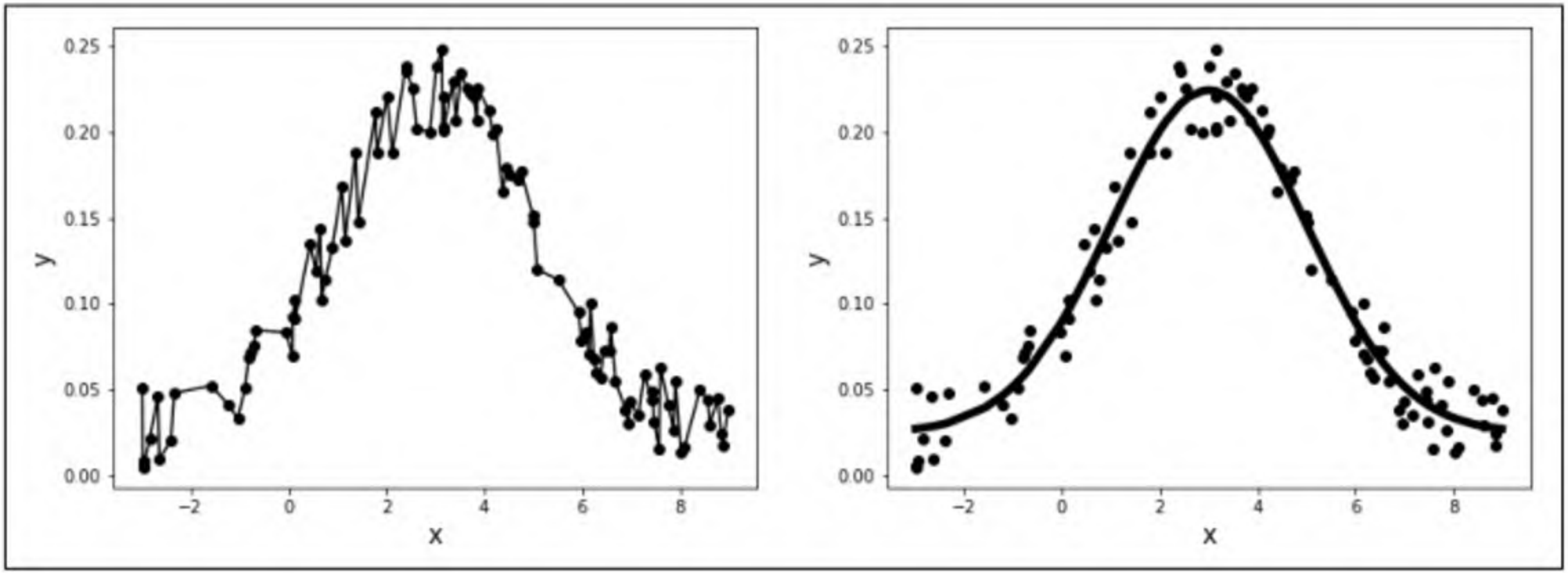
函数与数据的拟合

顾立平

01传统且非常有用的机器学习模型



02数值解与分析解

- 数值解和分析解是指解决数学问题时所得到的结果类型。
- 数值解是通过数值计算方法得出的近似解，通常需要一定的计算量和精度控制；
- 分析解则是通过解析方法得出的精确解，一般只适用于特定的数学问题。
- 在实际应用中，数值解更加常用和实用，但有时也需要结合分析解来进行理论验证和精度校验。

03回归：预测数值

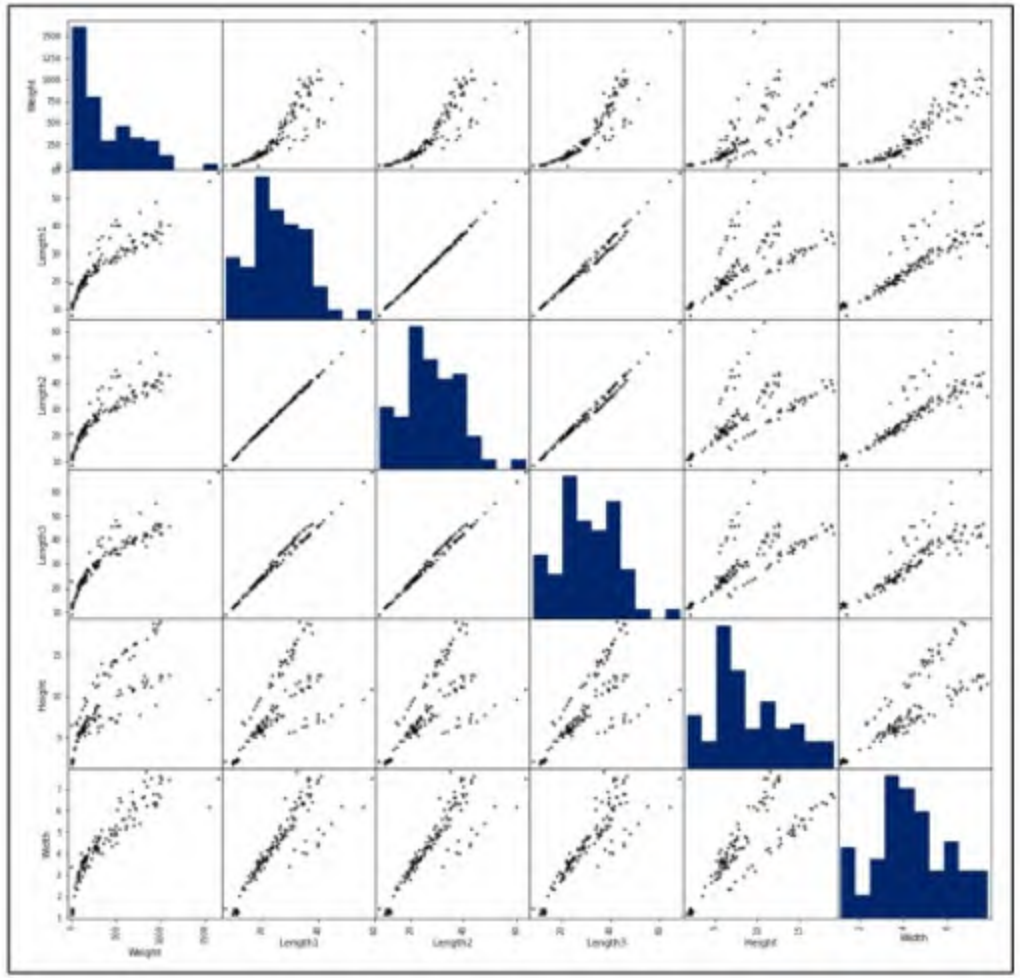
$$y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \omega_4 x_4 + \omega_5 x_5$$

- 回归是一种统计学方法，用于建立自变量和因变量之间的关系模型，从而可以利用已知的自变量来预测未知的因变量。
- 在机器学习中，回归常用于预测数值型数据，例如房价预测、股票价格预测等。

03回归：预测数值

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

03回归：预测数值



- 线性回归模型，它用于预测数值型数据。该模型通过线性组合数据的特征并加上一个常数偏置项来计算预测值。这个计算过程可以用数学表达式来表示，形如：

$$y = \omega 0 + \omega 1x1 + \omega 2x2 + \ldots + \omega nxn$$

- 其中y是预测值，x1，x2，...，xn是特征值，ω0，ω1，...，ωn是模型参数，需要通过训练数据来学习确定。

03回归：预测数值

- **线性回归模型的训练过程包括以下几个关键步骤：**

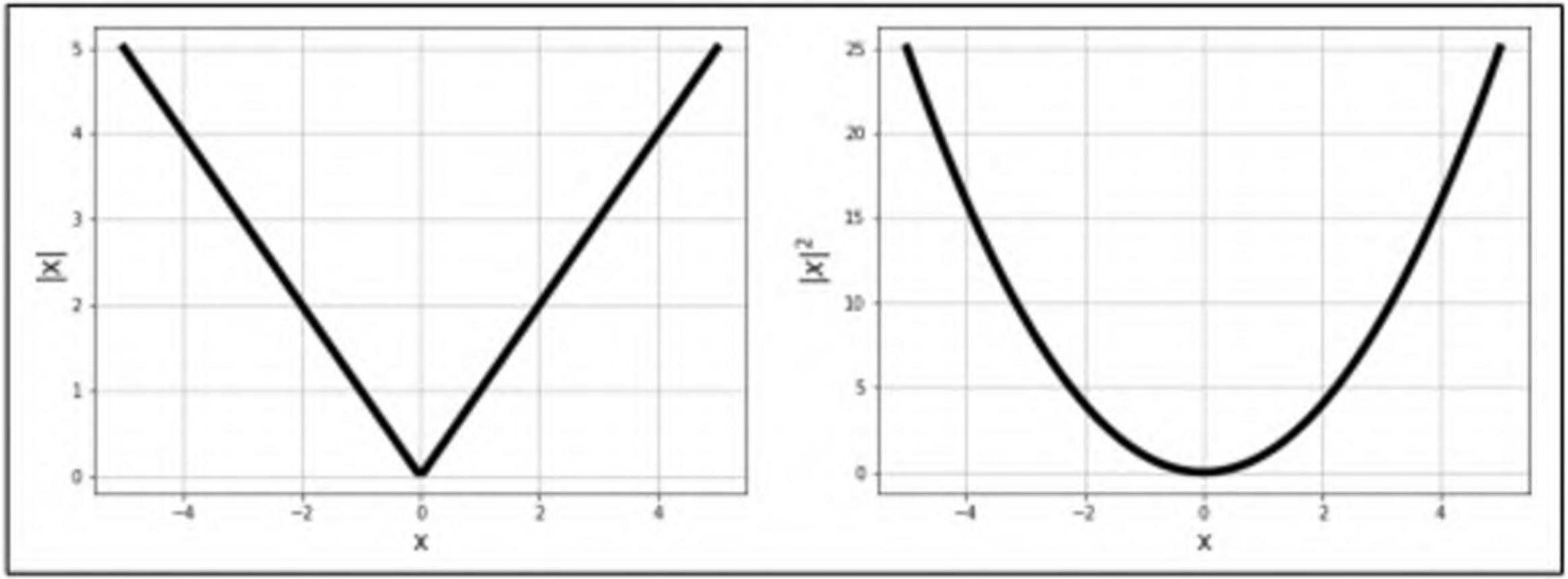
(1) 构建训练函数：选择线性函数作为训练函数，该函数基于输入特征（x值）计算预测值（y值）。

The predicted value versus the true value

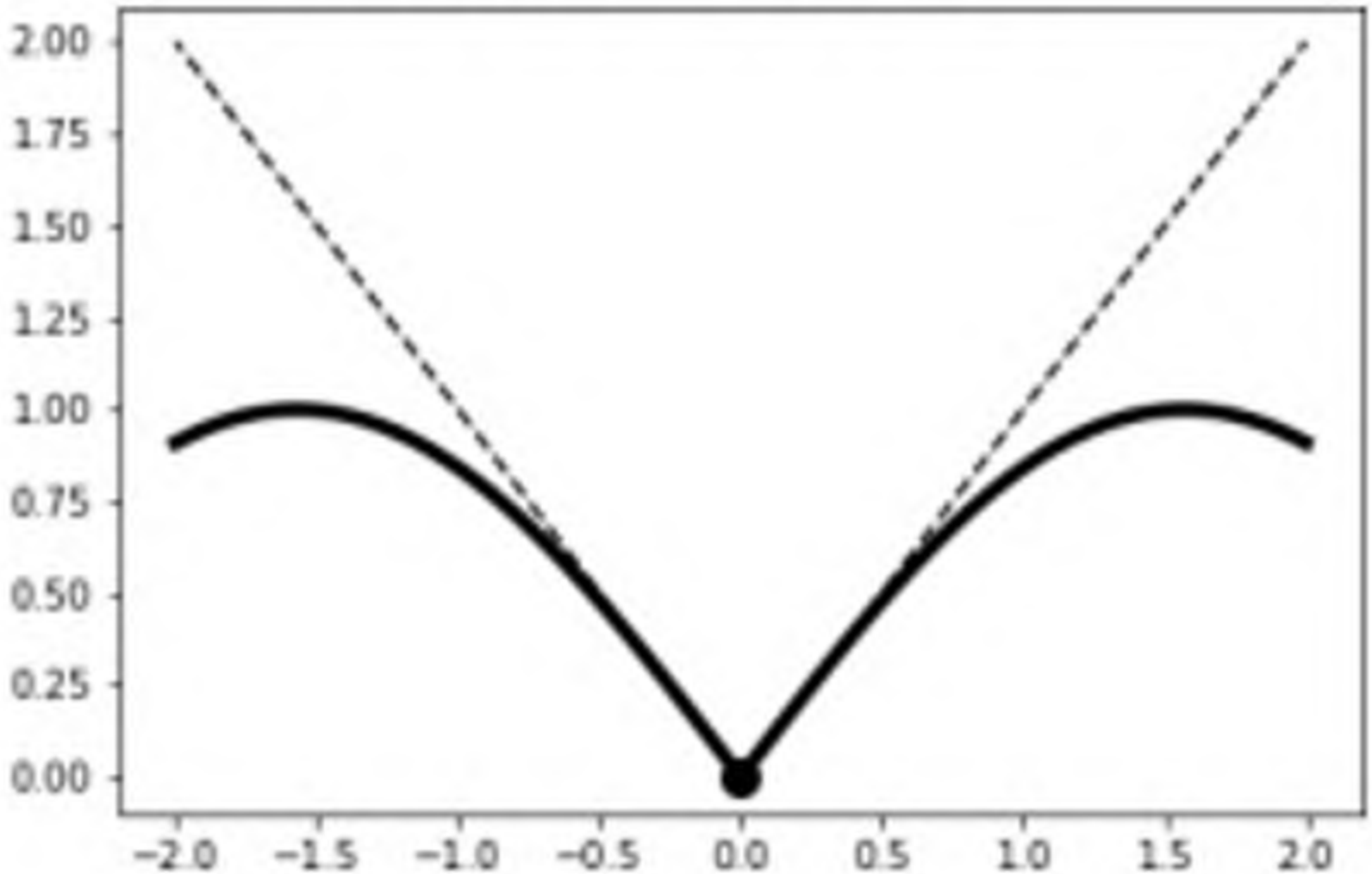
Suppose we assign some random numerical values for each of our unknown ω_0, ω_1, ω_2, ω_3, ω_4, and ω_5—say, for example, $\omega_0 = -3$, $\omega_1 = 4$, $\omega_2 = 0.2$, $\omega_3 = 0.03$, $\omega_4 = 0.4$, and $\omega_5 = 0.5$. Then the formula for the linear training function $y = \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \omega_4 x_4 + \omega_5 x_5$ becomes:

$$y = -3 + 4x_1 + 0.2x_2 + 0.03x_3 + 0.4x_4 + 0.5x_5$$

03回归：预测数值



03回归：预测数值



03回归：预测数值

- (2)确定损失函数：损失函数（也称作成本函数或误差函数）用于量化模型预测值与真实值之间的差异。在线性回归中，常用的损失函数是均方误差（Mean Squared Error），它计算预测值与真实值之间差值的平方和的平均值。

$$\text{Mean Squared Error} = \frac{1}{m}\left(\left|y^1_{predict} - y^1_{true}\right|^2 + \left|y^2_{predict} - y^2_{true}\right|^2 + \cdots + \left|y^m_{predict} - y^m_{true}\right|^2\right)$$

Let's write the above expression more compactly using the sum notation:

$$\text{Mean Squared Error} = \frac{1}{m}\Sigma_{i=1}^{m}\left|y^i_{predict} - y^i_{true}\right|^2$$

03回归：预测数值

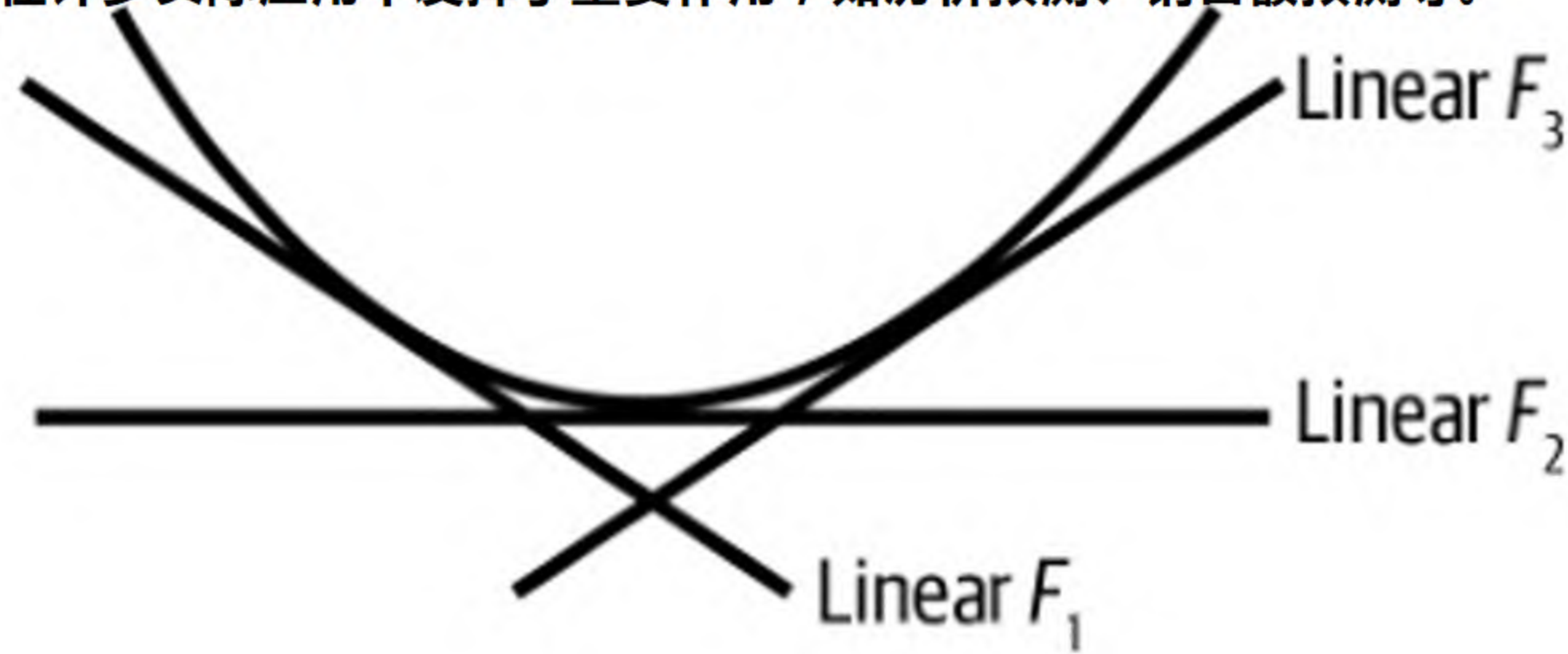
- (3)优化过程：优化过程的目标是找到能够最小化损失函数的模型参数（ω值）。这通常通过梯度下降等数值优化方法来实现。对于线性回归模型，由于其损失函数是凸函数，我们可以通过解析方法（如正规方程）直接求解最优参数，也可以通过迭代方法（如梯度下降）来逼近最优解。

$$\min_{\vec{\omega}} \text{loss function}$$

$$\min_{\vec{\omega}} \frac{1}{m}\left(X\vec{\omega} - \vec{y}_{true}\right)^t\left(X\vec{\omega} - \vec{y}_{true}\right) = \min_{\vec{\omega}} \frac{1}{m} \| X\vec{\omega} - \vec{y}_{true} \|_{l^2}^2$$

03回归：预测数值

- 通过这些步骤，线性回归模型能够学习到从输入特征到输出预测值的映射关系，并用于对新的、未见过的数据进行预测。这种预测能力使得线性回归模型在许多实际应用中发挥了重要作用，如房价预测、销售额预测等。

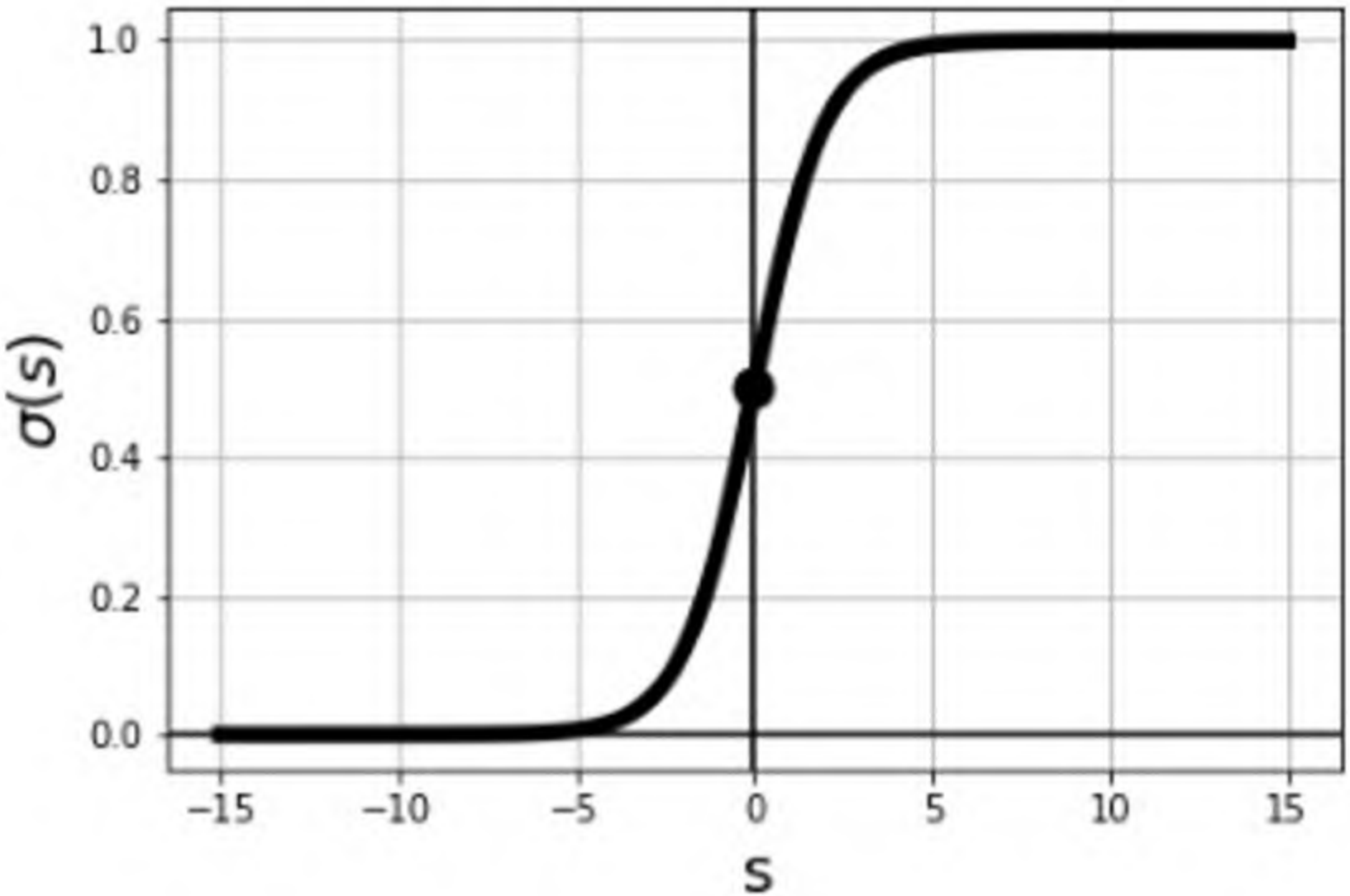


04 逻辑回归：分为两类

$$\sigma(s) = \frac{1}{1 + e^{-s}}$$

- 逻辑回归是一种分类算法，主要用于将数据分为两个类别。它通过计算特征的线性组合并加上常数偏差项，然后将结果通过逻辑函数转换为概率值，表示数据属于某一类别的可能性大小。
- 在实际应用中，通常采用阈值法将概率值转化为具体的类别标签。逻辑回归也可以扩展到多类别分类问题，称为多项式逻辑回归。

04 逻辑回归：分为两类



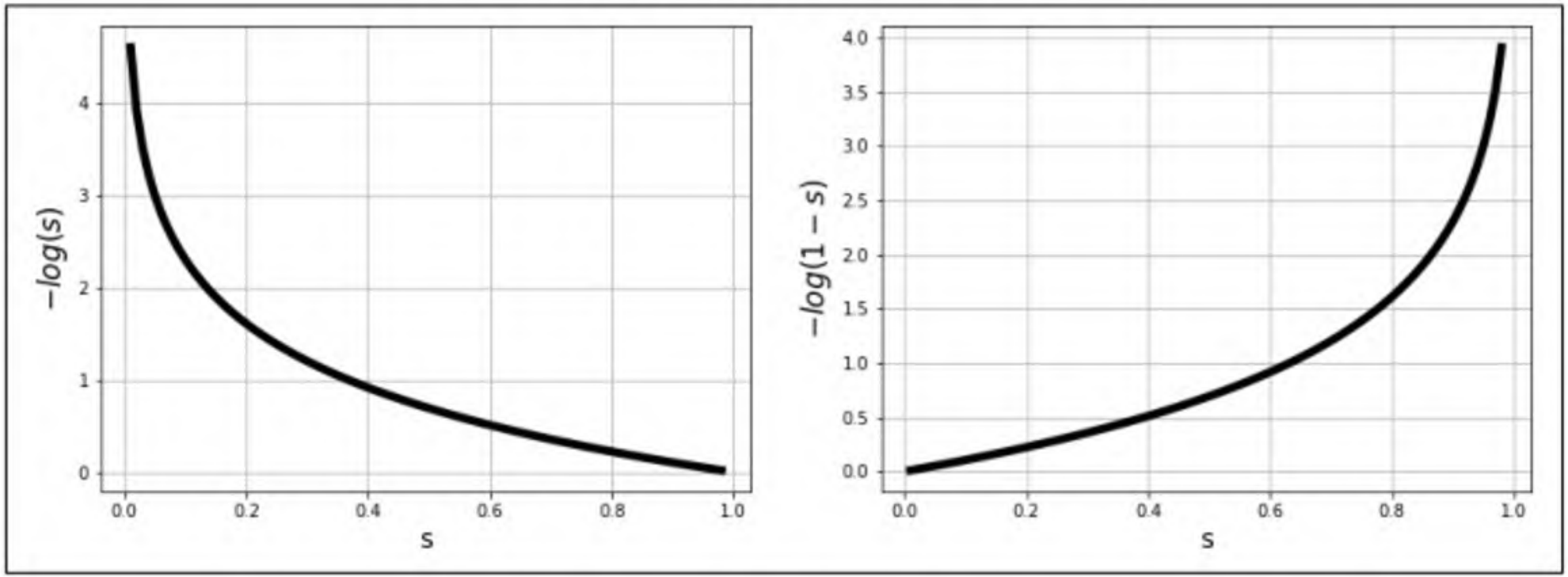
04 逻辑回归：分为两类

- 逻辑回归主要用于分类任务，特别是二分类任务（如将实例分类为“是/否”，“安全/不安全”，“有违约风险/无违约风险”等）。逻辑回归的训练函数首先通过对数据特征的线性组合加上一个常数偏差项来计算一个值，然后将该值传递给逻辑函数（也被称为Sigmoid函数）。逻辑函数的输出值介于0和1之间，因此可以将其解释为数据点属于某一类别的概率。

$$y = Thresh(\sigma(\omega_0 + \omega_1 x_1 + \cdots + \omega_n x_n))$$

- 训练函数：逻辑回归的训练函数首先计算特征的线性组合加上偏差项，然后将结果传递给逻辑函数。逻辑函数的形式为 σ(s) = 1 / (1 + e^(-s))，其中s是线性组合的结果。逻辑函数的输出值在0和1之间，因此可以作为分类概率。

04 逻辑回归：分为两类


$-log(s)$
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
s
$-log(1-s)$
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
s

04 逻辑回归：分为两类

- **损失函数：为了训练逻辑回归模型，需要设计一个损失函数来惩罚错误分类的训练数据点。对于逻辑回归，常用的损失函数是交叉熵损失函数。该函数基于逻辑函数的输出值和真实标签来计算每个训练实例的损失，并通过平均所有实例的损失来得到总损失。**

Therefore, we can write the cost for misclassifying one training instance $\left(x_1^i, x_2^i, \cdots, x_n^i; y_{true}\right)$ as:

$$
\begin{aligned}
cost &= \left\{ \begin{array}{l} -\log\left(\sigma(s)\right) \text{ if } y_{true} = 1 \\ -\log\left(1-\sigma(s)\right) \text{ if } y_{true} = 0 \end{array} \right\} \\
&= -y_{true} \log\left(\sigma(s)\right) - \left(1 - y_{true}\right) \log\left(1-\sigma(s)\right)
\end{aligned}
$$

04 逻辑回归：分为两类

- 优化：在定义了损失函数之后，下一步是找到使损失函数最小的权重值ω。由于逻辑回归的损失函数没有闭式解，通常使用数值方法（如梯度下降法）来寻找最小化损失函数的权重值。

$$L\left(\overrightarrow{\omega}\right) = -\frac{1}{m}\sum_{i=1}^{m} y_{true}^{i} \log\left(\sigma\left(\omega_0 + \omega_1 x_1^i + \cdots + \omega_n x_n^i\right)\right) +$$

$$\left(1 - y_{true}^{i}\right) \log\left(1 - \sigma\left(\omega_0 + \omega_1 x_1^i + \cdots + \omega_n x_n^i\right)\right)$$

- 逻辑回归模型通过训练数据学习权重值，这些权重值决定了每个特征在分类过程中的重要性。模型训练完成后，可以使用训练好的逻辑回归模型对新数据进行分类预测。

05 Softmax回归：分类为多个类

05 Softmax回归：分类为多个类

- Softmax回归是一种多类别分类算法，它可以将数据分为多个类别。与逻辑回归类似，Softmax回归也是通过计算特征的线性组合并加上常数偏差项，但是它使用的是Softmax函数将结果转换为概率分布，而不是逻辑函数。Softmax函数将每个类别的得分映射到0到1之间，并确保所有类别的得分之和等于1。在实际应用中，通常选择具有最高概率值的类别作为预测结果。Softmax回归可以处理任意数量的类别，因此非常适合于大规模多类别分类问题。

$$\sigma(s) = \frac{1}{1+e^{-s}} = \frac{1}{1+\frac{1}{e^{s}}} = \frac{e^{s}}{1+e^{s}} = \frac{e^{s}}{e^{0}+e^{s}}$$

05 Softmax回归：分类为多个类

$$\sigma(s) = \frac{e^s}{e^0 + e^s}$$

- Softmax回归模型：

✓Softmax回归主要用于多类别分类问题，如分类手写数字图像为0到9的十个数字类别。

✓它将线性回归的思想推广至多类别分类，通过计算数据点属于每个类别的概率来进行分类。

05 Softmax回归：分类为多个类

$$1 - \sigma(s) = \frac{e^0}{e^0 + e^s}$$

- 训练函数：

✓Softmax回归的训练函数首先会对数据的特征进行线性组合，并加上一个偏置项。

✓接着，它会将线性组合的结果输入到softmax函数中，该函数将输出一个概率分布，表示数据点属于每个类别的概率。

05 Softmax回归：分类为多个类

$$
\begin{aligned}
s^1 &= \omega_0^1 + \omega_1^1 x_1 + \omega_2^1 x_2 + \cdots + \omega_n^1 x_n \\
s^2 &= \omega_0^2 + \omega_1^2 x_1 + \omega_2^2 x_2 + \cdots + \omega_n^2 x_n \\
&\vdots \\
s^k &= \omega_0^k + \omega_1^k x_1 + \omega_2^k x_2 + \cdots + \omega_n^k x_n
\end{aligned}
$$

05 Softmax回归：分类为多个类

$$\sigma\left(s^j\right) = \frac{e^{sj}}{e^{s^1} + e^{s^2} + \cdots + e^{s^k}}$$

- 损失函数：
- 对于Softmax回归，常用的损失函数是交叉熵损失函数。这个函数会惩罚错误分类的训练数据点，通过计算真实类别概率的对数损失来实现。

05 Softmax回归：分类为多个类

$$y = j \text{ such that } \sigma\left(\omega_0^j + \omega_1^j x_1 + \cdots + \omega_n^j x_n\right) \text{ is maximal}$$

- 优化过程：

✓在确定了损失函数后，目标是找到能够最小化这个损失函数的参数（即权重ω）。

✓由于交叉熵损失函数是凸函数，因此可以使用梯度下降等方法来有效地找到最优参数。

05 Softmax回归：分类为多个类

We derived the cross-entropy loss function in the case of logistic regression:

$$L\left(\overrightarrow{\omega}\right) = -\frac{1}{m}\sum_{i=1}^{m} y_{true}^{i} \log\left(\sigma\left(\omega_0 + \omega_1 x_1^i + \cdots + \omega_n x_n^i\right)\right) +$$
$$\left(1 - y_{true}^{i}\right) \log\left(1 - \sigma\left(\omega_0 + \omega_1 x_1^i + \cdots + \omega_n x_n^i\right)\right)$$

using:

$$cost = \left\{ \begin{array}{l} -\log\left(\sigma(s)\right) \text{ if } y_{true} = 1 \\ -\log\left(1 - \sigma(s)\right) \text{ if } y_{true} = 0 \end{array} \right\}$$
$$= -y_{true} \log\left(\sigma(s)\right) - \left(1 - y_{true}\right) \log\left(1 - \sigma(s)\right)$$

05 Softmax回归：分类为多个类

$$cost = \begin{Bmatrix} -\log\left(\sigma\left(s^1\right)\right) \text{ if } y_{true,1} = 1 \\ -\log\left(\sigma\left(s^2\right)\right) \text{ if } y_{true,2} = 1 \\ -\log\left(\sigma\left(s^3\right)\right) \text{ if } y_{true,3} = 1 \\ \vdots \\ -\log\left(\sigma\left(s^k\right)\right) \text{ if } y_{true,k} = 1 \end{Bmatrix}$$

$$= -y_{true,1}\log\left(\sigma\left(s^1\right)\right) - \cdots - y_{true,k}\log\left(\sigma\left(s^k\right)\right)$$

05 Softmax回归：分类为多个类

$$L\left(\overrightarrow{\omega}\right) = -\frac{1}{m}\Sigma_{i=1}^{m} y_{true,1}^{i} \log\left(\sigma\left(\omega_0^1 + \omega_1^1 x_1^i + \cdots + \omega_n^1 x_n^i\right)\right) + y_{true,2}^{i}$$
$$\log\left(\sigma\left(\omega_0^2 + \omega_1^2 x_1^i + \cdots + \omega_n^2 x_n^i\right)\right) + \cdots + y_{true,k}^{i} \log\left(\sigma\left(\omega_0^k + \omega_1^k x_1^i + \cdots + \omega_n^k x_n^i\right)\right)$$

- 集成到神经网络中：
- Softmax回归模型可以集成到神经网络的最后一层，作为输出层使用，以便处理多类别分类问题。

05 Softmax回归：分类为多个类

$$\log\left(\frac{1}{p}\right) = -\log(p)$$

- Softmax回归是一种多类别分类模型，它通过对数据的特征进行线性组合，并使用softmax函数计算每个类别的概率来进行分类。通过最小化交叉熵损失函数，可以找到最优的参数设置。
- 此外，Softmax回归可以集成到神经网络的输出层，以处理多类别分类任务。

06 将一些模型合并到神经网络的最后一层

- 将不同的模型合并到神经网络的最后一层是一种常见的技术，被称为模型融合。
- 这种技术可以通过结合多个模型的预测结果来提高模型的准确性和鲁棒性。具体来说，可以将多个模型的输出连接起来，形成一个多通道的输出层，然后使用softmax或其他激活函数将它们映射到各自的类别上。这样做的好处是可以减少过拟合的风险，同时也可以利用不同模型的优点来提高整体性能。
- 在实践中，常用的模型融合方法包括投票法、加权平均法、堆叠法等。

07支持向量机

$$f\left(\vec{\omega};\vec{x}\right)=sign\left(\vec{\omega}^{t}\vec{x}+\omega_0\right)$$

$$\max\left(0,1-y_{true}\left(\vec{\omega}^{t}\vec{x}+\omega_0\right)\right)$$

- 支持向量机（Support Vector Machine，SVM）是一种二分类模型，它的目标是找到一个超平面，能够将不同类别的样本分开，并使得这个超平面距离最近的样本点到超平面的距离最大化。
- 这些最近的样本点被称为支持向量，它们决定了超平面的位置和形状。
- SVM 的核心思想是在高维空间中寻找最优的超平面来进行分类，它可以处理线性和非线性的分类问题，并且对于高维数据集有很好的表现。SVM 还有一些变种，如核函数 SVM 和多类 SVM，它们可以进一步扩展 SVM 的应用范围。

07支持向量机

$$L\left(\overrightarrow{\omega}\right) = \frac{1}{m}\Sigma_{i=1}^{m} \max\left(0, 1 - y_{true}^{i}\left(\overrightarrow{\omega}^{t}\overrightarrow{x}^{i} + \omega_0\right)\right) + \lambda \parallel \omega \parallel_2^2$$

- 训练函数：支持向量机首先将数据的特征通过一个线性组合加上一个常数偏置项进行计算，然后将结果通过一个符号函数（sign function）进行处理。
- 如果线性组合的结果为正数，则分类为第一类（返回1），如果为负数，则分类为第二类（返回-1）。其训练函数公式为：f(ω; x) = sign(ω^T x + ω0)，其中ω为权重向量，x为特征向量，ω0为偏置项。

07支持向量机

It has two terms: $\frac{1}{m}\sum_{i=1}^{m} \max\left(0, 1 - y_{true}^{i}\left(\vec{\omega}^{t}\vec{x}^{i} + \omega_0\right)\right)$ and $\lambda \| \vec{\omega} \|_2^2$.

- 损失函数：支持向量机的损失函数基于一个称为铰链损失函数（hinge loss function）的概念，其公式为：max(0, 1 - y_true ω^T x + ω0)。
- 这个损失函数用于惩罚分类错误的点，以及分类正确但距离决策边界过近的点。

07支持向量机

$$\min_{\vec{\omega}} \frac{1}{m}\Sigma_{i=1}^{m} \max\left(0,1 - y_{true}^{i}\left(\vec{\omega}^{t}\vec{x}^{i} + \omega_0\right)\right) + \lambda \| \omega \|_2^2$$

- 优化：支持向量机的目标是最小化损失函数。这个损失函数由两部分组成：一部分是铰链损失的平均值，用于惩罚分类错误的点和距离决策边界过近的点；
- 另一部分是权重向量的L2范数的平方（ // ω // ^2 ），用于控制模型的复杂度，避免过拟合。损失函数的优化是一个凸优化问题，因此可以确保找到全局最优解。此外，支持向量机还可以使用对偶问题（dual problem）进行求解，这在某些情况下会更加高效。

07支持向量机

$$\max_{\vec{\alpha}} \Sigma_{j=1}^{m} \alpha_j - \frac{1}{2} \Sigma_{j=1}^{m} \Sigma_{k=1}^{m} \alpha_j \alpha_k y_{true}^{j} y_{true}^{k} \left(\left(\vec{x}^{j} \right)^{t} \vec{x}^{k} \right)$$

$$f\left(\vec{x}_{new}\right) = sign\left(\vec{\omega}^{t} \vec{x}_{new} + \omega_0\right)$$

$$= sign\left(\sum_{j} \alpha_j y^{i} \left(\vec{x}^{j}\right)^{t} \vec{x}_{new} + \omega_0\right)$$

07支持向量机

$$K\left(\vec{x}^{j},\vec{x}^{k}\right)=\vec{\phi\left(\vec{x}^{j}\right)}^{t}\phi\left(\vec{x}^{k}\right)$$

- 核技巧：支持向量机可以通过核技巧扩展到非线性分类问题。通过将原始数据映射到高维空间，使得原本非线性可分的数据在高维空间中变得线性可分。
- 核技巧允许我们在不显式计算映射函数的情况下，直接在原始数据空间中使用核函数计算高维空间中的点积，从而大大减少了计算量。

07支持向量机

- 支持向量机是一种强大的分类和回归工具，它通过最大化分类间隔来寻找最优的决策边界，并能够通过核技巧处理非线性问题。
- 由于其数学原理的优越性，支持向量机在机器学习和数据挖掘领域得到了广泛应用。

$$K\left(\vec{x}^{j},\vec{x}^{k}\right)=\left(\left(\vec{x}^{j}\right)^{t}\vec{x}^{k}\right)^{2}$$

The polynomial kernel: $K\left(\vec{x}^{j},\vec{x}^{k}\right)=\left(1+\left(\vec{x}^{j}\right)^{t}\vec{x}^{k}\right)^{d}$

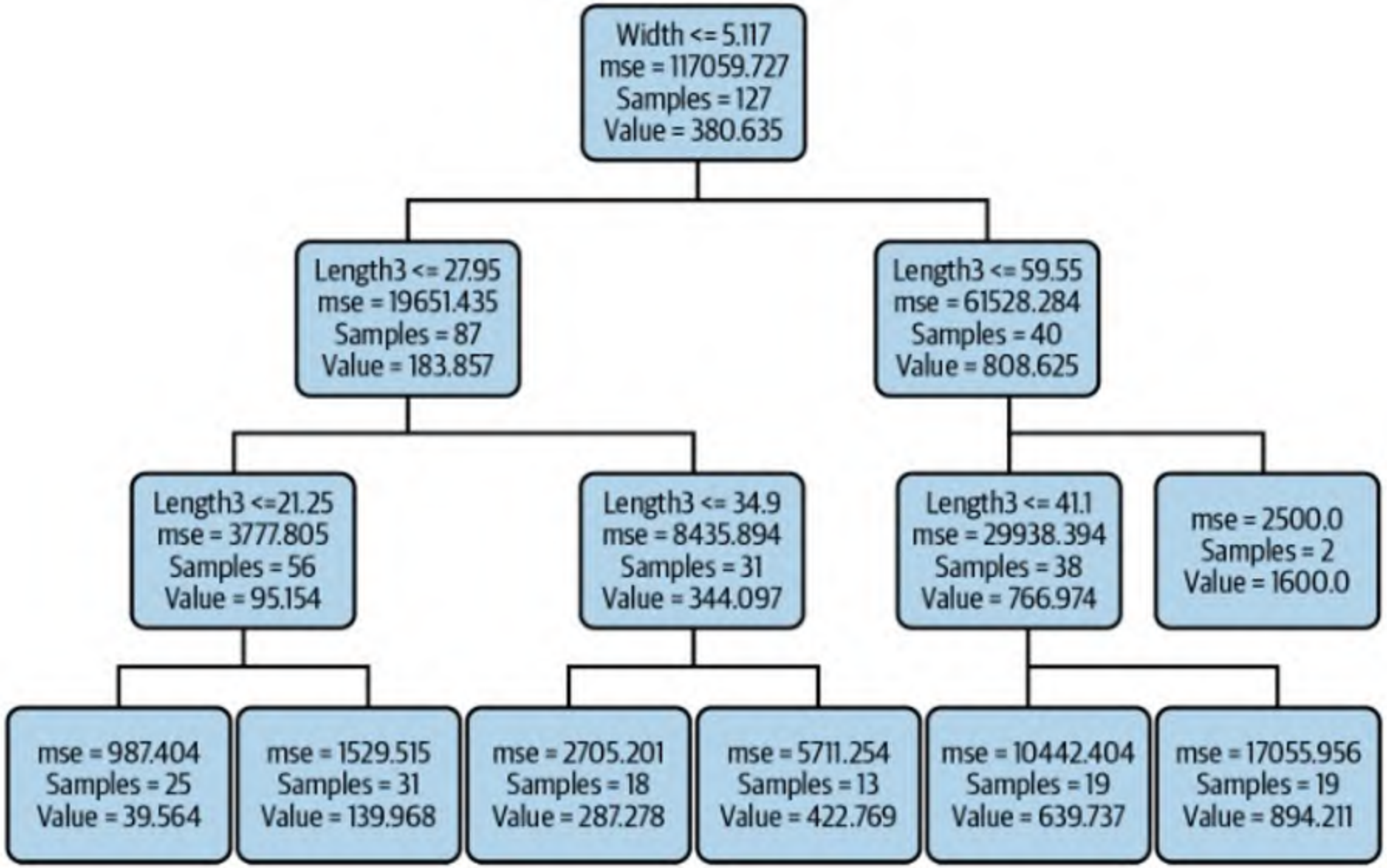
The Gaussian kernel: $K\left(\vec{x}^{j},\vec{x}^{k}\right)=e^{-\gamma\left|x_j-x_k\right|^2}$

08决策树

1. $a1 = (\text{Width} \leq 5.117)$
2. $a2 = (\text{Length3} \leq 59.55)$
3. $a3 = (\text{Length3} \leq 41.1)$
4. $a4 = (\text{Length3} \leq 34.9)$
5. $a5 = (\text{Length3} \leq 27.95)$
6. $a6 = (\text{Length3} \leq 21.25)$

- 决策树是一种基于树结构进行决策的算法，它通过对数据进行递归划分，生成一棵树形结构，从而实现对数据的分类或预测。
- 决策树的核心思想是通过选择最优的特征来进行分裂，使得子节点中的样本尽可能属于同一类别或具有相似的属性。决策树的优点包括易于理解和解释、适用于各种类型的数据、能够处理缺失值等。

08决策树


Width <= 5.117
mse = 117059.727
Samples = 127
Value = 380.635
Length3 <= 27.95
mse = 19651.435
Samples = 87
Value = 183.857
Length3 <= 59.55
mse = 61528.284
Samples = 40
Value = 808.625
Length3 <=21.25
mse = 3777.805
Samples = 56
Value = 95.154
Length3 <= 34.9
mse = 8435.894
Samples = 31
Value = 344.097
Length3 <= 41.1
mse = 29938.394
Samples = 38
Value = 766.974
mse = 2500.0
Samples = 2
Value = 1600.0
mse = 987.404
Samples = 25
Value = 39.564
mse = 1529.515
Samples = 31
Value = 139.968
mse = 2705.201
Samples = 18
Value = 287.278
mse = 5711.254
Samples = 13
Value = 422.769
mse = 10442.404
Samples = 19
Value = 639.737
mse = 17055.956
Samples = 19
Value = 894.211

08决策树

- 然而，决策树也存在一些缺点，例如容易出现过拟合、对噪声敏感等问题。为了克服这些问题，人们发展出了很多改进的决策树算法，如随机森林、梯度提升决策树等。

$$f(a1,a2,a3,a4,a5,a6) = (a1 \text{ and } a5 \text{ and } a6) \times 39.584 + (a1 \text{ and } a5 \text{ and not } a6) \times 139.968 + (a1 \text{ and not } a5 \text{ and } a4) \times 287.278 + (a1 \text{ and not } a5 \text{ and not } a4) \times 422.769 + (\text{not } a1 \text{ and } a2 \text{ and } a3) \times 639.737 + (\text{not } a1 \text{ and } a2 \text{ and not } a3) \times 824.211 + (\text{not } a1 \text{ and not } a2) \times 1600$$

$$Surprise(event) = \log \frac{1}{p(event)} = -\log\left(p(event)\right)$$

08决策树

	Weight	Length1	Length2	Length3	Height	Width
Weight	1.000000	0.908678	0.911888	0.917883	0.747700	0.896036
Length1	0.908678	1.000000	0.999493	0.991731	0.637844	0.870414
Length2	0.911888	0.999493	1.000000	0.993869	0.653291	0.877268
Length3	0.917883	0.991731	0.993869	1.000000	0.716450	0.882716
Height	0.747700	0.637844	0.653291	0.716450	1.000000	0.802115
Width	0.896036	0.870414	0.877268	0.882716	0.802115	1.000000

08决策树

- 决策树的构建通常包括以下几个步骤：
- 特征选择：从数据集中选择最优特征进行分裂。常用的特征选择方法有信息增益、增益率、基尼指数等。

$$Entropy(X) = -p(outcome_1)\log\left(p(outcome_1)\right) - p(outcome_2)\log\left(p(outcome_2)\right) - \cdots - p(outcome_n)\log\left(p(outcome_n)\right)$$

$$Entropy(Feature) = -p(value_1)\log\left(p(value_1)\right) - p(value_2)\log\left(p(value_2)\right) - \cdots - p(value_n)\log\left(p(value_n)\right)$$

08决策树

- 决策树剪枝：为了解决决策树过拟合问题，可以通过剪枝来简化决策树结构。剪枝有预剪枝和后剪枝两种策略。

$$
\begin{aligned}
&Entropy(\text{Outcome Feature}) \\
&= -p(positive)\log\left(p(positive)\right) - p(negative)\log\left(p(negative)\right) \\
&= -\frac{p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{n}{p+n}\right)
\end{aligned}
$$

08决策树

$$\text{Entropy}(value_1) = -\frac{p_1}{p_1+n_1}\log\left(\frac{p_1}{p_1+n_1}\right) - \frac{n_1}{p_1+n_1}\log\left(\frac{n_1}{p_1+n_1}\right)$$

$$\text{Entropy}(value_2) = -\frac{p_2}{p_2+n_2}\log\left(\frac{p_2}{p_2+n_2}\right) - \frac{n_2}{p_2+n_2}\log\left(\frac{n_2}{p_2+n_2}\right)$$

$$\text{Entropy}(value_3) = -\frac{p_3}{p_3+n_3}\log\left(\frac{p_3}{p_3+n_3}\right) - \frac{n_3}{p_3+n_3}\log\left(\frac{n_3}{p_3+n_3}\right)$$

$$\text{Entropy}(value_4) = -\frac{p_4}{p_4+n_4}\log\left(\frac{p_4}{p_4+n_4}\right) - \frac{n_4}{p_4+n_4}\log\left(\frac{n_4}{p_4+n_4}\right)$$

08决策树

- **决策树生成：根据选择的特征将数据集划分为子集，并递归地在每个子集上重复此过程，直到满足停止条件（如子集中所有样本都属于同一类别，或没有特征可以再用等）。**

$$
\begin{aligned}
&\text{Expected Entropy(Feature A)} \\
&= p(value_1)\text{Entropy}(value_1) + p(value_2)\text{Entropy}(value_2) \\
&+ p(value_3)\text{Entropy}(value_3) + p(value_4)\text{Entropy}(value_4) \\
&= \frac{k_1}{m}\text{Entropy}(value_1) + \frac{k_2}{m}\text{Entropy}(value_2) + \frac{k_3}{m}\text{Entropy}(value_3) + \frac{k_4}{m}\text{Entropy}(value_4)
\end{aligned}
$$

08决策树

- 为了提高决策树的性能，可以采取以下优化策略：

✓集成学习：通过结合多个决策树来提高预测精度，如随机森林和梯度提升树等算法。

$$\text{Information Gain} = \textit{Entropy}(\text{Outcome Feature}) - \text{Expected Entropy}(\text{Feature A}) =$$

$$-\frac{p}{p+n}\log\left(\frac{p}{p+n}\right) - \frac{n}{p+n}\log\left(\frac{n}{p+n}\right) - \text{Expected Entropy}(\text{Feature A})$$

08决策树

- 为了提高决策树的性能，可以采取以下优化策略：

✓连续变量处理：对于连续变量，可以采用离散化或基于信息增益等方法进行处理。

$$\text{Information Gain}=$$

$$Entropy(\text{Outcome Feature}) - \text{Expected Entropy(Feature A)} =$$

$$-\frac{p}{p+n}\log\left(\frac{p}{p+n}\right)-\frac{n}{p+n}\log\left(\frac{n}{p+n}\right)-\text{Expected Entropy(Feature A)}$$

08决策树

- 为了提高决策树的性能，可以采取以下优化策略：

✓特征选择优化：除了常用的信息增益等方法外，还可以尝试其他特征选择方法以提高性能。

$$\text{Gini impurity} = 1 - \left(\frac{n_1}{n}\right)^2 - \left(\frac{n_2}{n}\right)^2 - \left(\frac{n_3}{n}\right)^2$$

$$\min_{Feature, FeatureSplitValue} \frac{n_{left}}{n} Gini(\text{Left Node}) + \frac{n_{right}}{n} Gini(\text{Right Node})$$

08决策树

- 为了提高决策树的性能，可以采取以下优化策略：

✓参数调优：调整决策树的参数（如最大深度、最小样本数等）以找到最佳模型。

$$\Sigma_{LeftNodeInstances}\left|y^i_{true}-y_{left}\right|^2$$

$$\Sigma_{RightNodeInstances}\left|y^i_{true}-y_{right}\right|^2$$

$$\frac{n_{left}}{n}\sum_{LeftNodeInstances}\left|y^i_{true}-y_{left}\right|^2+\frac{n_{right}}{n}\sum_{RightNodeInstances}\left|y^i_{true}-y_{right}\right|^2$$

09随机森林

- 随机森林是一种基于决策树的集成学习算法，它由多个决策树组成，每个决策树独立地对数据进行划分和预测。
- 随机森林的特点是采用随机抽样的方式来构建每个决策树，包括随机抽取特征和随机抽取样本，这样可以减少过拟合的风险，提高模型的泛化能力。
- 随机森林的优点包括准确率高、鲁棒性强、能够处理高维数据、能够处理缺失值等。
- 随机森林的应用广泛，如分类、回归、异常检测等领域。

10k-均值聚类

- k-均值聚类是一种常见的无监督机器学习算法，用于将数据集分成k个不同的类别。
- 它通过迭代的方式不断更新每个数据点所属的簇心位置，使得同一簇内的数据点到簇心的距离最小，不同簇之间的距离最大。
- k-均值聚类的结果取决于初始簇心的选择，通常需要多次运行才能得到最优解。
- k-均值聚类常用于图像分割、文本聚类、生物信息学等领域。

11 分类模型的性能度量

- 分类模型的性能度量是用来评估分类模型在对新数据进行分类时的表现能力的指标，帮助我们选择最适合特定任务的分类模型，并对其进行调优。
- 常用的性能度量包括准确率、混淆矩阵、精度得分等。
- 其中，准确率是指分类正确的样本占总样本数的比例；混淆矩阵则可以用来计算真阳性、假阳性、真阴性和假阴性的数量；精度得分则是指分类器正确预测出正类的数量占预测出的所有正类样本的比例。

True negative	False positive
False negative	True positive

$$Accuracy = \frac{\text{true positives} + \text{true negatives}}{\text{all predicted positives} + \text{all predicted negatives}}$$

11 分类模型的性能度量

$$Precision = \frac{\text{true positives}}{\text{all predicted positives}} = \frac{\text{true positives}}{\text{true positives + false positives}}$$

$$Recall = \frac{\text{true positives}}{\text{all positive labels}} = \frac{\text{true positives}}{\text{true positives + false negatives}}$$

$$Specificity = \frac{\text{true negatives}}{\text{all negative labels}} = \frac{\text{true negatives}}{\text{true negatives + false positives}}$$

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

谢谢！

gulp@mail.las.ac.cn