挺简单！但并不是一无是处

- ~~线性回归~~到逻辑回归
- ~~异常检测~~
- 特征工程
- 决策树到集成算法 (作业一)

开始DL..

- 神经网络介绍
- CNN/RNN

硬核篇

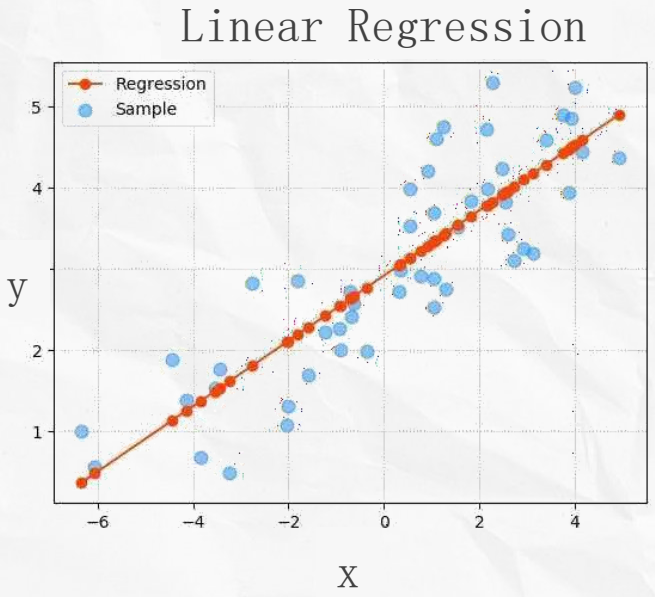
- Transformer (作业二)
- Time-series Forecasting
- LLM
- 大作业

机器学习

从线性回归到逻辑回归

线性回归

- 定义



$$f_{瓜甜}(\boldsymbol{x}) = 0.2 \cdot x_{色泽} + 0.5 \cdot x_{根蒂} + 0.3 \cdot x_{敲声} + 1$$

线性回归

- 最小二乘法

设定模型的形式：$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$

设定误差的形式（均方误差MSE）：$\ell(f(\boldsymbol{x}_i), y_i) = (f(\boldsymbol{x}_i) - y_i)^2$

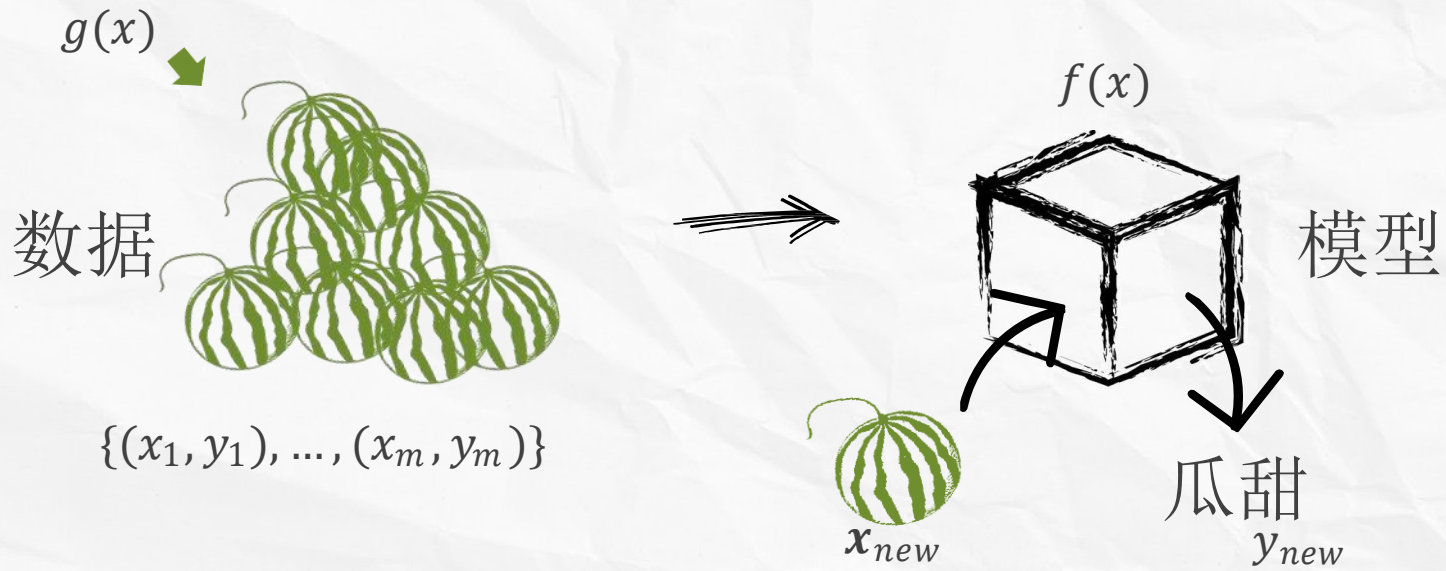
利用最小化训练误差求解模型参数：$\arg\min_{(w,b)} \sum_{i=1}^{m} (f(\boldsymbol{x}_i) - y_i)^2 = \operatorname{argmin}_{(w,b)} \sum_{i=1}^{m} (y_i - w\boldsymbol{x}_i - b)^2$

在欧几里得空间中，点$x = (x_1,\ldots,x_n)$和$y = (y_1,\ldots,y_n)$之间的欧氏距离为

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

向量$\vec{x}$的自然长度，即该点到原点的距离为

$$\|\vec{x}\|_2 = \sqrt{|x_1|^2 + \cdots + |x_n|^2}$$



线性回归

- 最小二乘法（二元）

设定模型的形式：$f(\boldsymbol{x}) = w_1x_1 + w_2x_2 + \cdots + w_dx_d + b$ 简化→$f(x) = wx + b$

设定误差的形式：$\ell(f(x_i), y_i) = (f(x_i) - y_i)^2$

利用最小化训练误差求解模型参数：$\arg\min_{(w,b)} \sum_{i=1}^{m}(f(x_i) - y_i)^2 = \operatorname{argmin}_{(w,b)} \sum_{i=1}^{m}(y_i - wx_i - b)^2$

解析解：

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

线性回归

利用最小化训练误差求解模型参数：$\arg\min_{(w,b)} \sum_{i=1}^{m}(f(x_i)-y_i)^2 = \operatorname{argmin}_{(w,b)} \sum_{i=1}^{m}(y_i - wx_i - b)^2$

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

对参数w求偏导

$\frac{\partial \mathrm{E}(w,b)}{\partial w} = \sum_{i=1}^{m} 2(y_i - wx_i - b)(-x_i) = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)x_i\right)$，令其为0

对参数b求偏导

$\frac{\partial \mathrm{E}(w,b)}{\partial b} = \sum_{i=1}^{m} 2(y_i - wx_i - b)(-1) = 2(mb - \sum_{i=1}^{m}(y_i - wx_i))$，令其为0 ⇒ $b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$

记为$b = \bar{y} - w\bar{x}$，代入上式

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

$$w\sum_{i=1}^{m} x_i^2 = \sum_{i=1}^{m}(y_i - \bar{y} + w\bar{x})x_i$$

线性回归

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

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

然后和书上结果不一样....

变换一下

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

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

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

线性回归

利用最小化训练误差求解模型参数：$\arg\min_{(w,b)} \sum_{i=1}^{m}(f(x_i)-y_i)^2 = \operatorname{argmin}_{(w,b)} \sum_{i=1}^{m}(y_i - wx_i - b)^2$

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

对参数w求偏导

$\frac{\partial \mathrm{E}(w,b)}{\partial w} = \sum_{i=1}^{m} 2(y_i - wx_i - b)(-x_i) = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)x_i\right)$，令其为0

对参数b求偏导

$\frac{\partial \mathrm{E}(w,b)}{\partial b} = \sum_{i=1}^{m} 2(y_i - wx_i - b)(-1) = 2(mb - \sum_{i=1}^{m}(y_i - wx_i))$，令其为0 ⇒ $b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$

代入上式

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

线性回归

- 最小二乘法(多元、矩阵形式)

$$\mathbf{X}=\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix}=\begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix} \qquad \boldsymbol{y}=(y_1;y_2;\ldots;y_m)$$

$$\hat{\boldsymbol{w}}^*=arg\min_{\hat{\boldsymbol{w}}}(\boldsymbol{y}-\mathbf{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y}-\mathbf{X}\hat{\boldsymbol{w}})$$

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

$$\frac{\partial \boldsymbol{a}^T\mathbf{X}}{\partial \mathbf{X}}=\boldsymbol{a},\frac{\partial \mathbf{X}^T\mathbf{X}}{\partial \mathbf{X}}=2\mathbf{X}$$

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

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

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

现实任务中 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 往往不是满秩矩阵，此时可解出多个$\hat{\boldsymbol{w}}$都能使均方误差最小化. 选择哪一个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) 项，例如 Lasso, Ridge.

变量$\mathbf{X}$矩阵中不同列可能存在线性相关性，或者列数＞行数（而行秩=列秩），因此很可能不是满秩。

归纳偏好（inductive bias）是指学习算法在面对数据时所隐含的偏好或者假设。

线性回归

- 梯度下降法

考虑无约束优化问题$min_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$
若能构造一个序列$\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \ldots$
满足$f(\boldsymbol{\theta}^{t+1}) < f(\boldsymbol{\theta}^t)$，$t=0, 1, 2, \ldots$
则不断执行该过程即可收敛到局部极小点
根据泰勒展式有$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \simeq f(\boldsymbol{\theta}) + \Delta\boldsymbol{\theta}^{\mathrm{T}} \nabla f(\boldsymbol{\theta})$
于是，欲满足$f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) < f(\boldsymbol{\theta})$
可选择$\Delta\boldsymbol{\theta} = -\eta \nabla f(\boldsymbol{\theta})$
其中步长η是一个小常数. 这就是梯度下降法

p. s. 梯度的每个分量是函数相对于每个参数的偏导数

线性回归

- 梯度下降法求解最小二乘法

> 批量梯度下降

$$w^{(t+1)} = w^{(t)} - \eta \nabla \mathcal{L}$$

$$w^{(t+1)} = w^{(t)} + \eta \sum_{i=1}^{m} \left(y_i - w^{(t)T} x_i\right) x_i$$

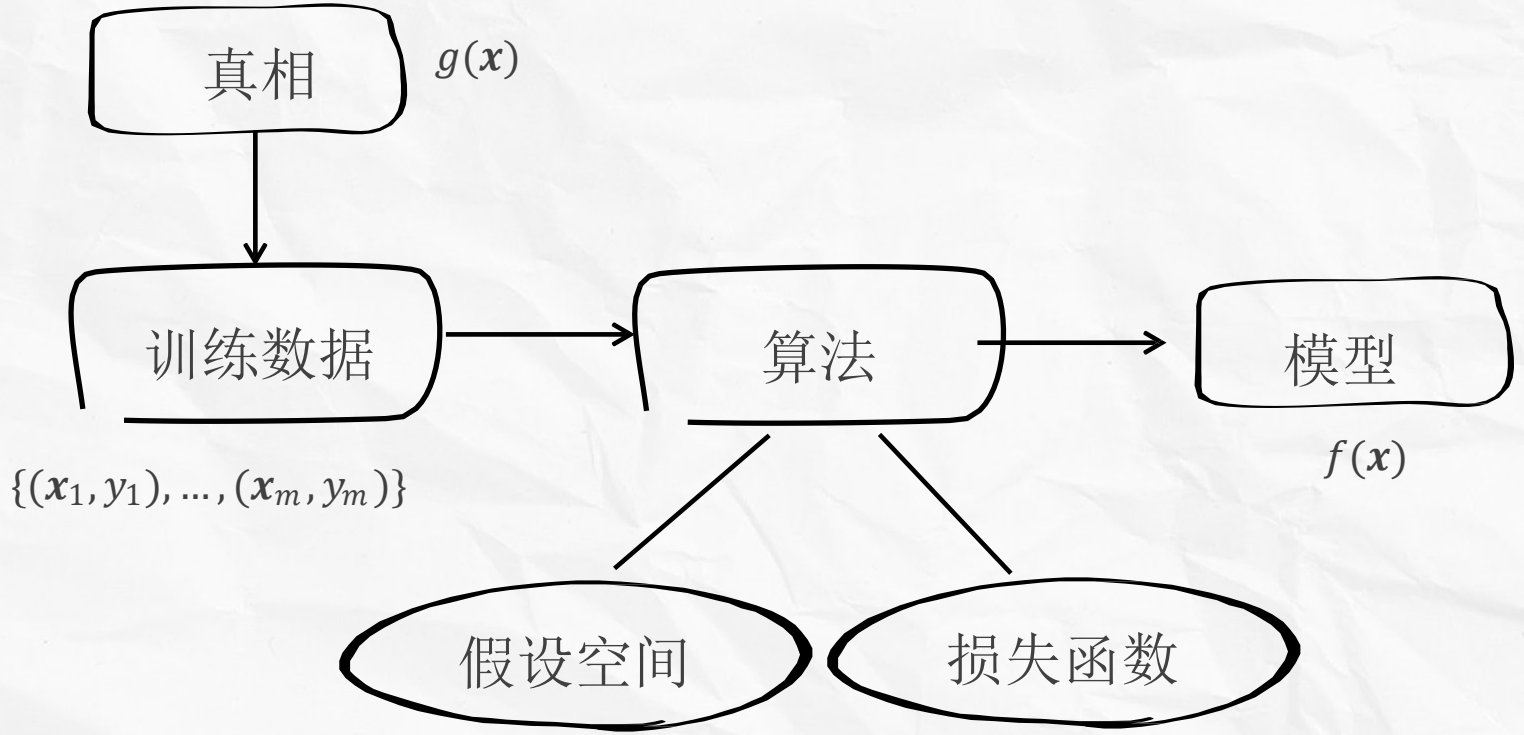
> 随机梯度下降

$$w^{(t+1)} = w^{(t)} - \eta \nabla \mathcal{L}_i$$

$$w^{(t+1)} = w^{(t)} + \eta (y_i - w^{(t)T} x_i) \ \ x_i$$

线性回归

- 机器学习框架



e.$g, f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$

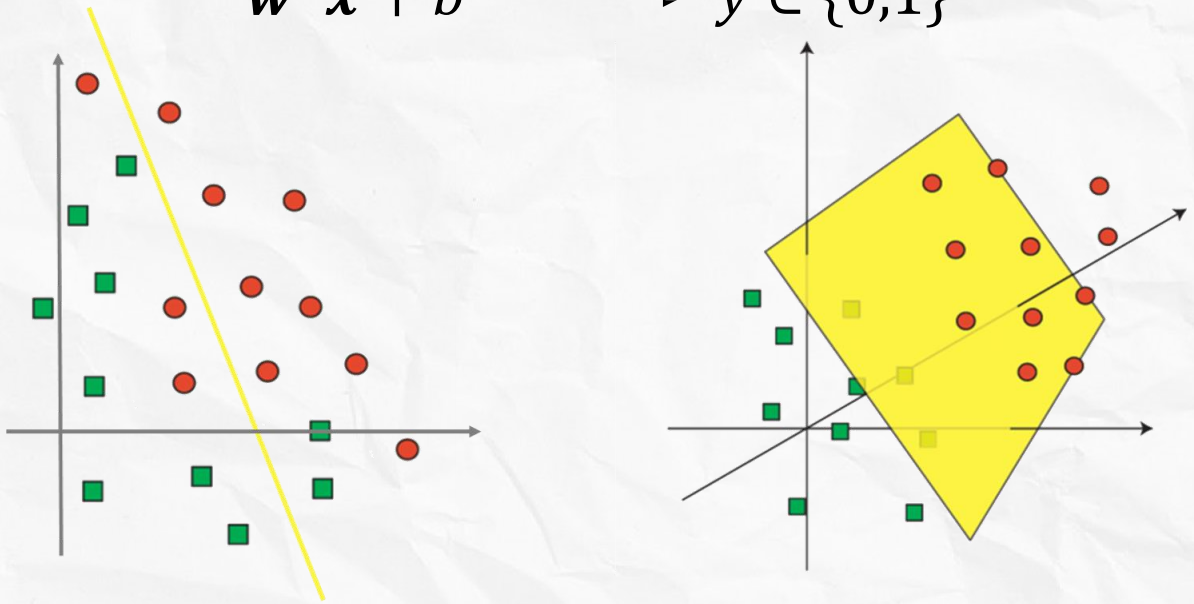
$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$

e.$g, P(f(\boldsymbol{x}_i), y_i) = (f(\boldsymbol{x}_i) - y_i)^2$

逻辑回归

- 线性分类超平面

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b \xrightarrow{\text{决定}} y \in \{0,1\}$$



$$y=\begin{cases}0, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b<0\\ 1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\geq 0\end{cases}$$

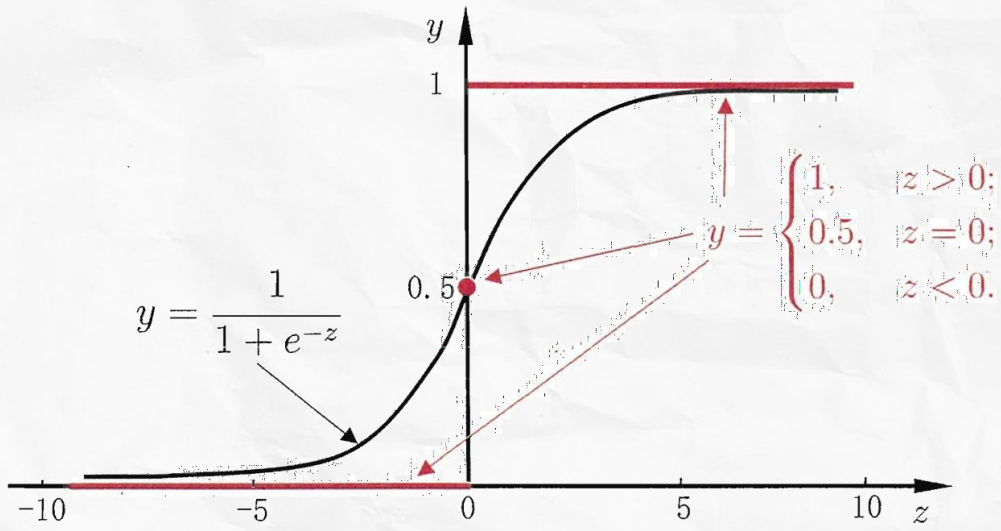
逻辑回归

- sigmoid函数

$$p(y=1 \mid \boldsymbol{x})$$

$$= g^{-1}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)$$

$$= \frac{1}{1+e^{-\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)}}$$



逻辑回归

- 对数几率回归

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

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

$$\ln \frac{p(y=1 \mid \boldsymbol{x})}{p(y=0 \mid \boldsymbol{x})}$$

几率（事件发生于不发生概率之比）

对数几率

回归

逻辑回归

上述logistic regression过程当然可以用NN (Neural Network)来做

- nn.Linear + nn.Sigmoid
- 用nn.Sequential来wrap
- 用binary cross entropy来当做loss，用梯度反向传播来更新参数

扩展：多分类的实现

- Softmax回归

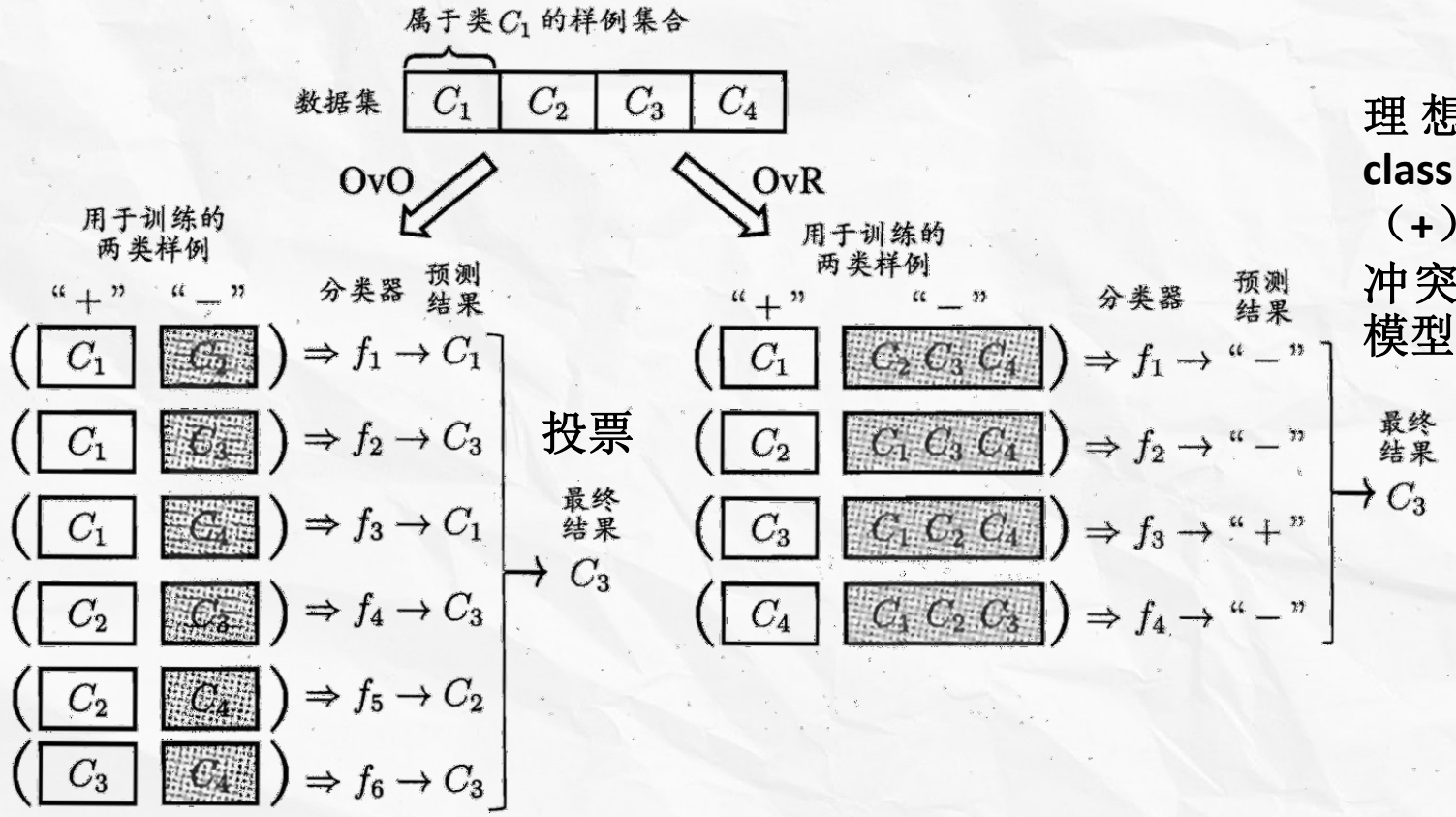
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right) \begin{array}{l} \longleftrightarrow \boldsymbol{\beta}_1^{\mathsf{T}} \hat{\boldsymbol{x}} \\ \longleftrightarrow \boldsymbol{\beta}_2^{\mathsf{T}} \hat{\boldsymbol{x}} \\ \longleftrightarrow \boldsymbol{\beta}_3^{\mathsf{T}} \hat{\boldsymbol{x}} \end{array}$$

用向量来理解，假设一条样本有10个特征，1x10 * ? → 1x3的logits (然后过softmax得到概率分布)，?=10x3的参数矩阵
即1x10 * 10x3 → 1x3，只不过上面的是以列向量（转置）的形式写的，即
(1x10 * 10x3).T→ (1x3).T
3x10 * 10x1 → 3x1

$$p(y = c \mid \boldsymbol{x}) = \text{softmax}(\boldsymbol{\beta}_c^{\mathsf{T}} \hat{\boldsymbol{x}}) = \frac{\exp(\boldsymbol{\beta}_c^{\mathsf{T}} \hat{\boldsymbol{x}})}{\sum_{c=1}^{C} \exp(\boldsymbol{\beta}_c^{\mathsf{T}} \hat{\boldsymbol{x}})}$$

扩展：多分类的实现

- 一对一OVO/一对其余OVR



理想情况下只有一个class被预测为整正类（+）。如果同时有多个冲突的结果，需要考虑模型预测结果的置信度

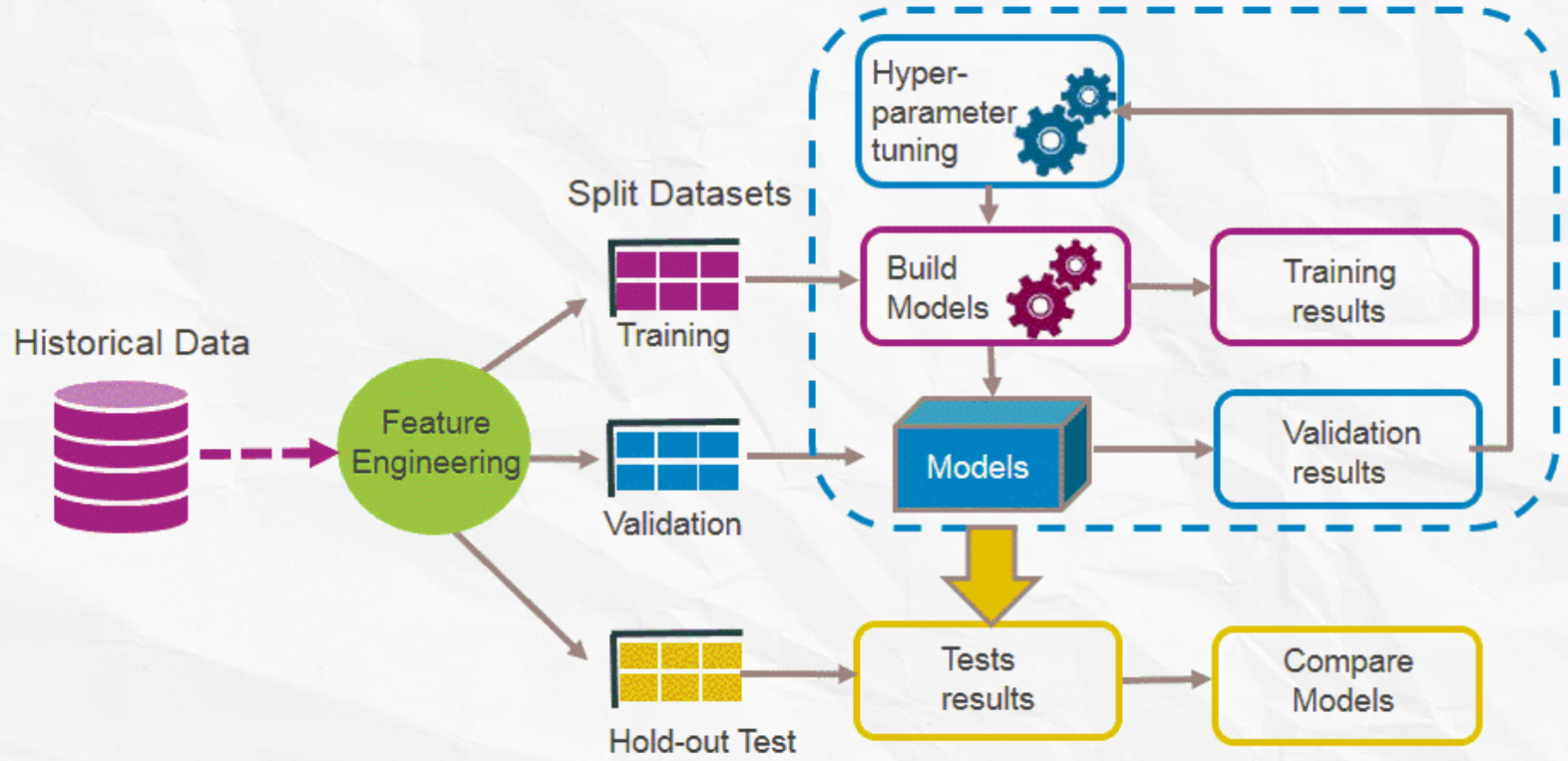
机器学习

特征工程

特征工程

- 涵义

特征工程指的是这样一个过程：将数据转换为能更好的（特征）表示，从而提高后续机器学习的性能。

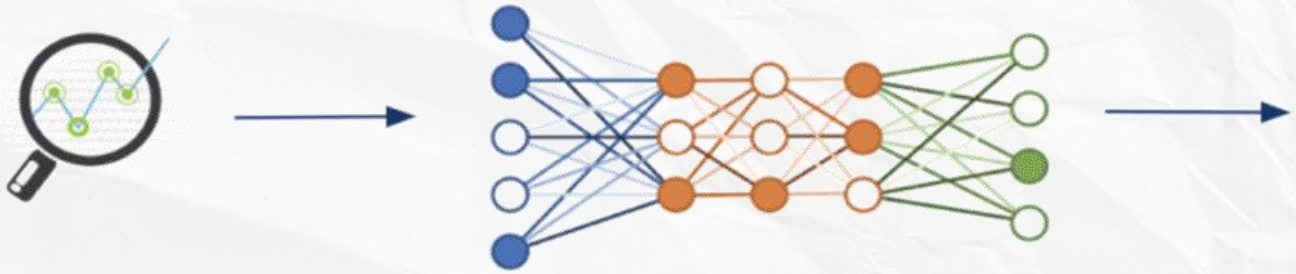


特征工程

- 实现手法
  - **人工分析与调用传统机器学习算法**

  - 深度神经网络 （TODO）

DL理论上不需要依赖于复杂的特征工程，
但实际并不是非黑即白的0/1问题...



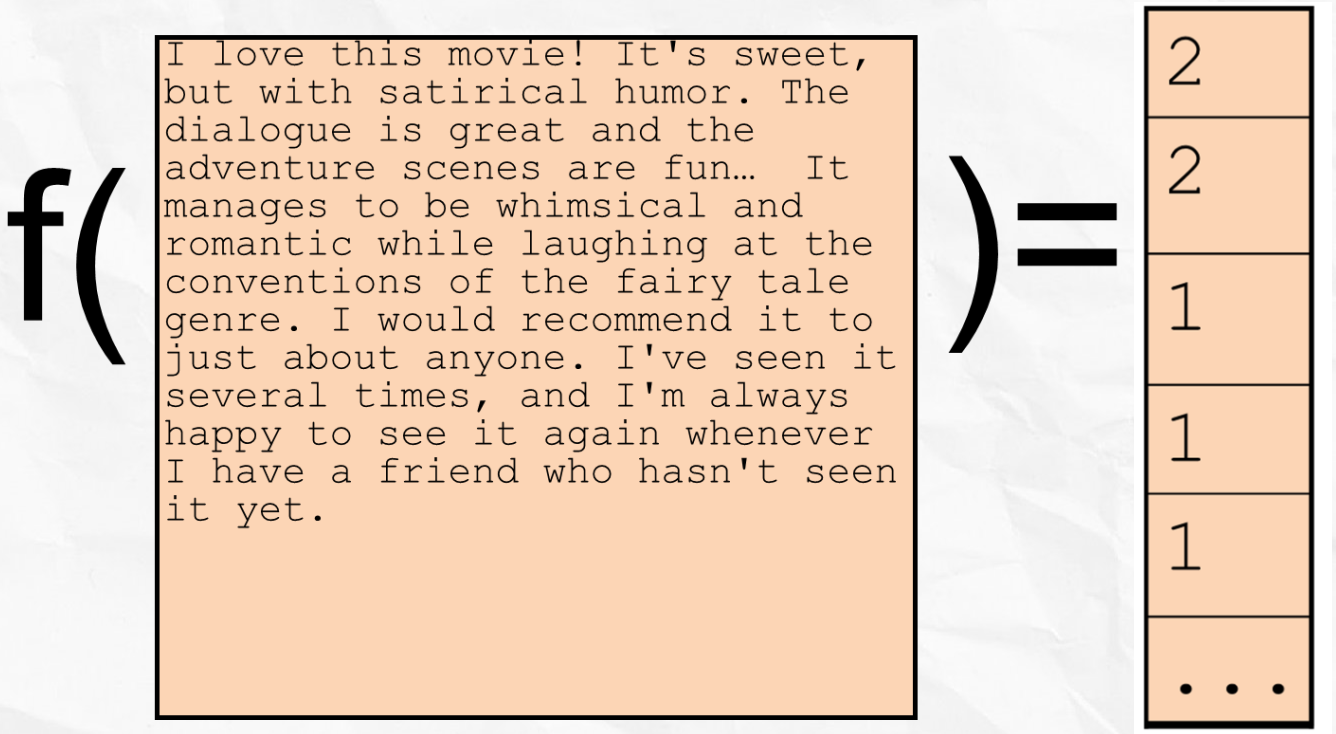
特征工程

- 范围
  - 特征构建
  - 特征探索
  - 特征增强
  - 特征衍生
  - 特征选择
  - 特征转换
  - ...

特征工程

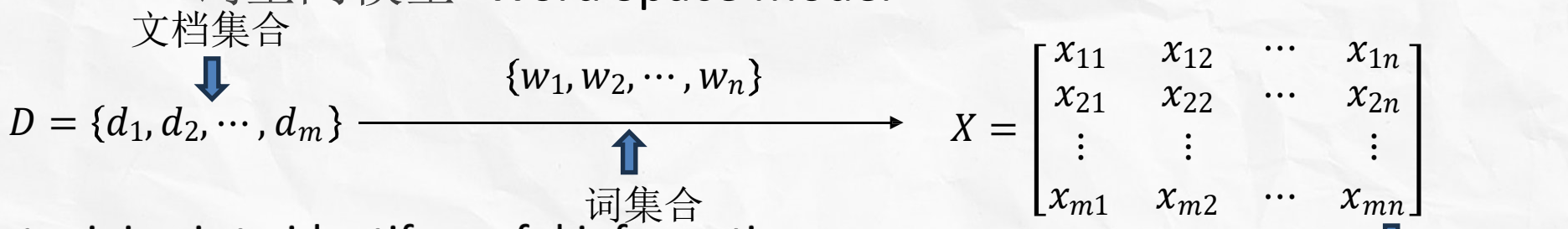
- 特征构建（以文本数据为例）

🤔LLM时代之前我应该怎么做NLP任务？比如文档分类或者情感分析，例如影评的好坏



特征工程

- 特征构建（以文本数据为例）
  - 词空间模型 Word Space Model



Doc1: Text mining is to identify useful information.

Doc2: Useful information is mined from text.

Doc3: Apple is delicious.

	text	information	identify	mining	mined	is	useful	to	from	apple	delicious
Doc1	1	1	1	1	0	1	1	1	0	0	0
Doc2	1	1	0	0	1	1	1	0	1	0	0
Doc3	0	0	0	0	0	1	0	0	0	1	1

特征工程

- 特征构建（以文本数据为例）
  - 词空间模型 Word Space Model

$$D = \{d_1, d_2, \cdots, d_m\} \xrightarrow{\{w_1, w_2, \cdots, w_n\}} X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

> Term Frequency (TF): $x_{12} = c(w_2, d_2)$

> TF-IDF: $x_{12} = c(w_2, d_2)$ x $IDF(w_2)$, $IDF(w_2) = 1 + \log\left(\frac{|D|}{df(w_2)}\right)$, $df(w_2) = c(w_2, D)$

逆文档频率，其中$|D|$是文档个数，$df(w_2)$是包含第二个词的文档个数，因此出现频次越低说明罕见程度越高，具有更高的区分度，使得IDF值越大

特征工程

- 特征探索
  - 探索特征值的分布（例如尖峰厚尾）
  - 探索不同特征的关联性（例如多重共线性）
  - 探索特征与目标的关联性
  - 探索特征的质量 …
- 特征增强
  - 填充缺失值
  - 剔除异常值 …
- 特征衍生
  - 不同特征的加减乘除 …

特征工程

- 特征选择与特征转换

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \xrightarrow{\text{特征选择}} \begin{bmatrix} x_1 \\ x_1 \\ \\ x_{d'} \end{bmatrix}$$

特征选择从给定的特征集合中选取出相关特征子集

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \xrightarrow{\text{特征转换}} \begin{bmatrix} y_1 \\ y_2 \\ \\ y_{d'} \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}\right)$$

特征转换通过某种数学变换将原始的特征空间变为另外的一个空间

特征工程

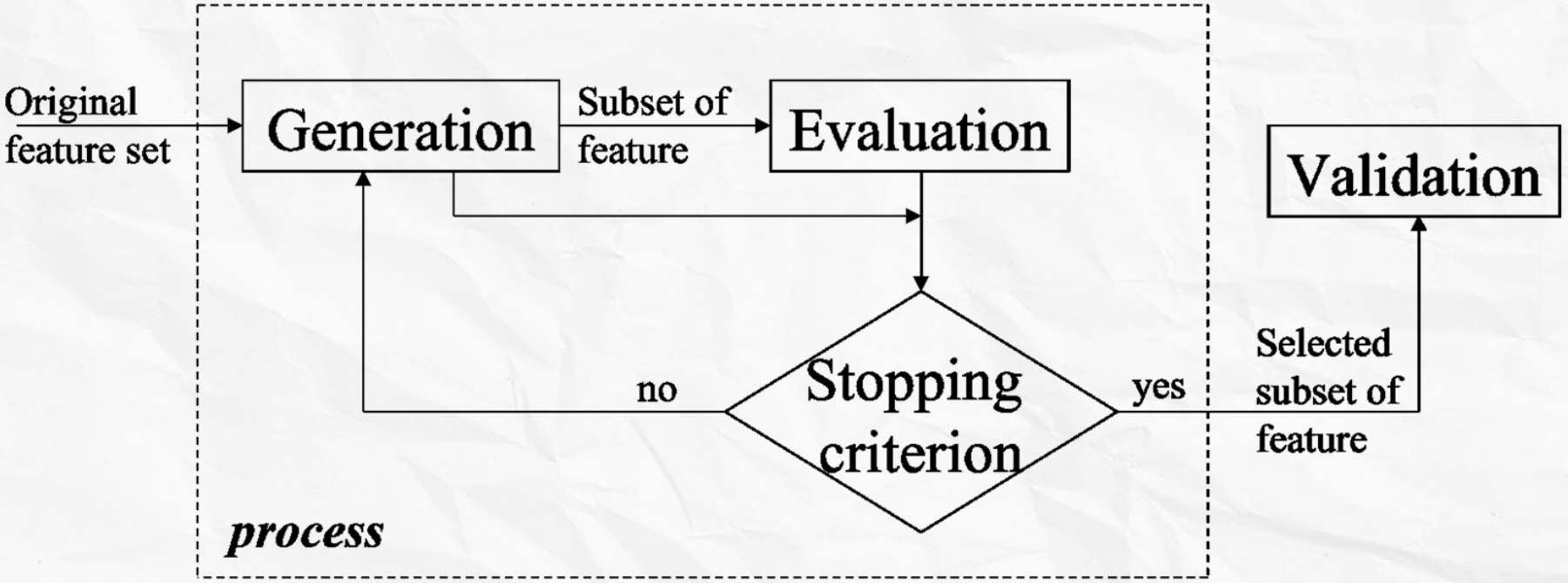
特征选择与特征转换

特征选择与特征转换通常达到一个特征约简目的，从而：

- 提升后续任务效果
- 降低后续任务的开销
- 对数据能有更进一步地认识

特征选择

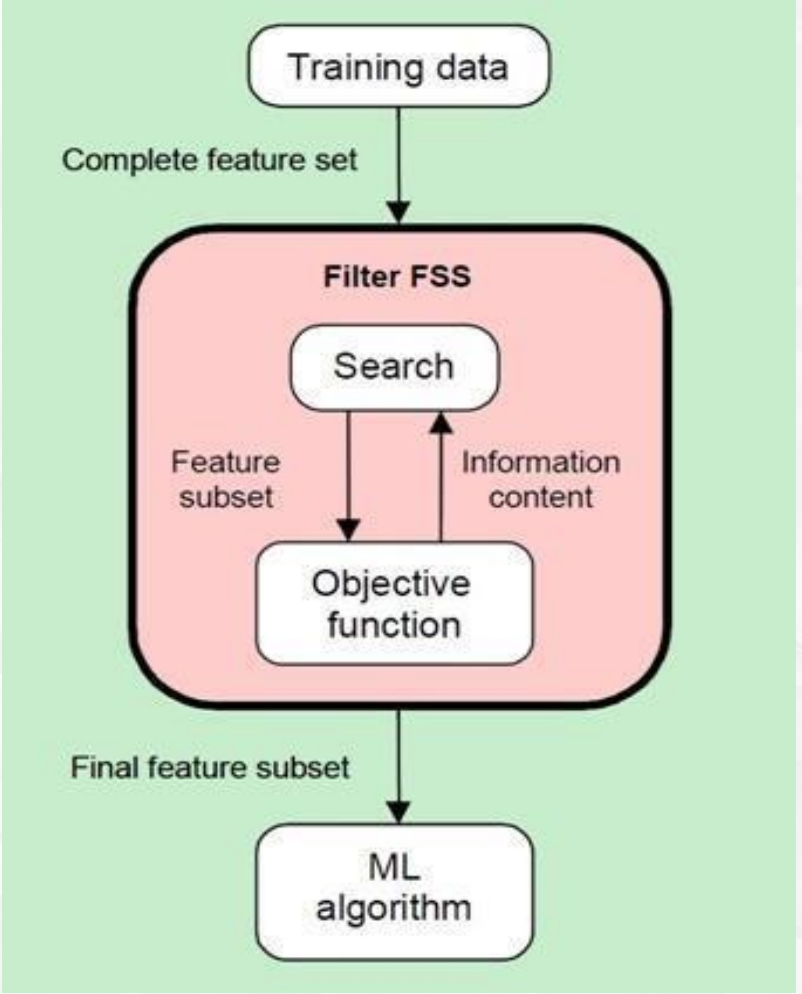
- 框架


Original
feature set
Generation
Subset of
feature
Evaluation
Validation
Selected
subset of
feature
yes
no
Stopping
criterion
process

特征选择

- 过滤式

过滤式方法先对数据集进行特征选择，然后再训练学习器，**特征选择过程与后续学习器无关**. 这相当于先用特征选择过程对初始特征进行“过滤”，再用过滤后的特征来训练模型



特征选择

- 过滤式
  - 基于单个特征评价

为单一特征的好坏设定一个打分规则，这个分数可以是其区分数据的能力等，然后设定：

特征子集的评估 Evaluation = 特征子集中每个特征的分数相加
特征子集的生成 Generation = 每次挑选最高的特征加入到当前的特征子集中
停止策略 Stopping criterion = 特征个数或欲加入子集的特征的分数小于某个阈值

特征选择

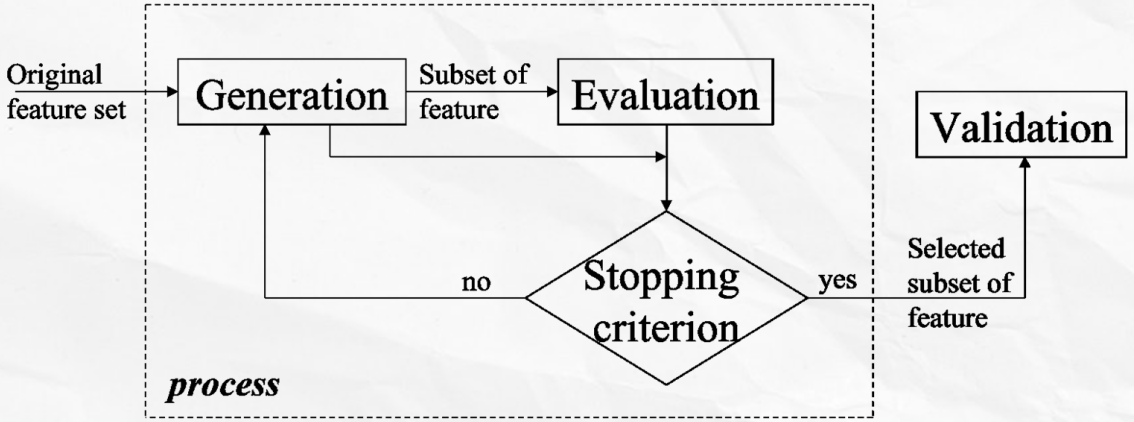
- 过滤式
  - 基于特征子集评价

考虑到特征之间的冗余性与协作性，不一定能够满足一个特征子集的分数可以由其中包含的特征的分数简单相加，此时当我们有了一个为一个特征子集的好坏设定一个打分规则，然后设定：

特征子集的评估 Evaluation = 特征子集的分数

特征子集的生成 Generation = **多用启发式搜索**

停止策略 Stopping criterion = 特征个数或其他



特征选择

- 过滤式
  - 基于序列前向选择 (SFS , Sequential Forward Selection)

特征集合 $\{a_1, a_2, \ldots, a_d\}$

对这d个候选单特征子集进行评价, 假定 $\{a_2\}$ 最优

↓

将 $\{a_2\}$ 作为第一轮的选定集

加入一个特征构成包含两个特 征的候选子集, 假定在这$d-1$个候选两特征子集中, $\{a_2, a_4\}$ 最优, 且优于 $\{a_2\}$

↓

将 $\{a_2, a_4\}$ 作为本轮的选定集

⋮

假定在第 $k+1$ 轮时 最优的候选 $(k+1)$ 特征子集不如 上一轮的选定集, 则停止生成候选子集

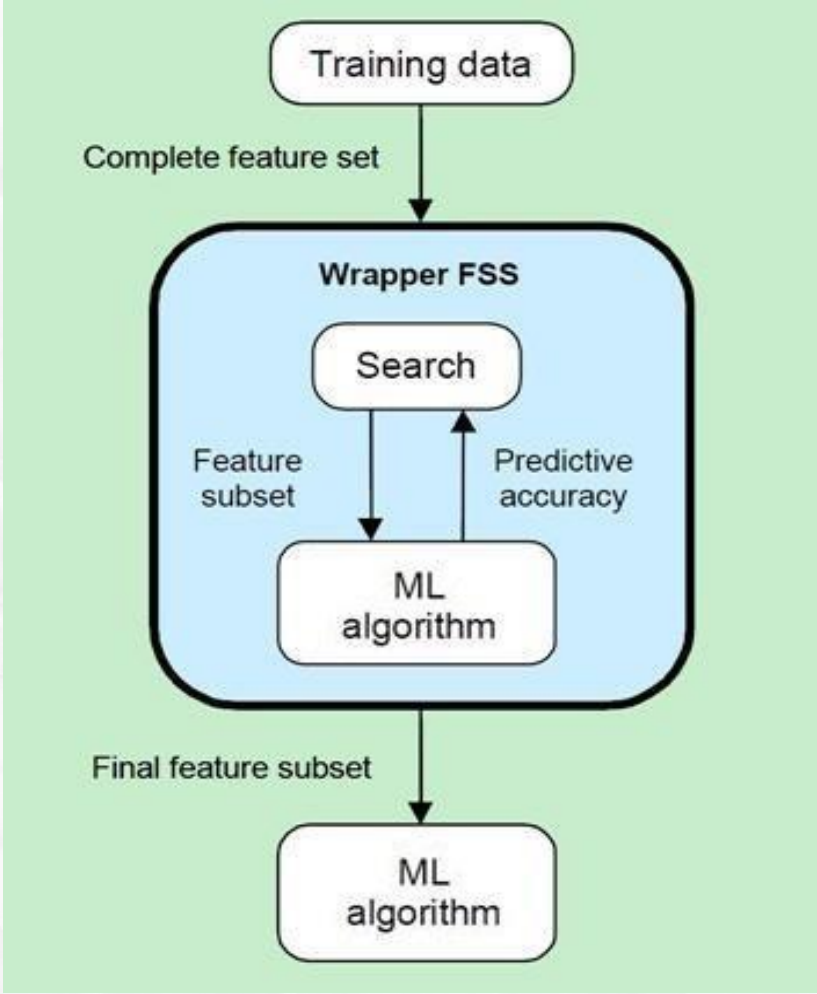
↓

将上一轮选定的k个特征集合作为特征选择结果

特征选择

- 包裹式

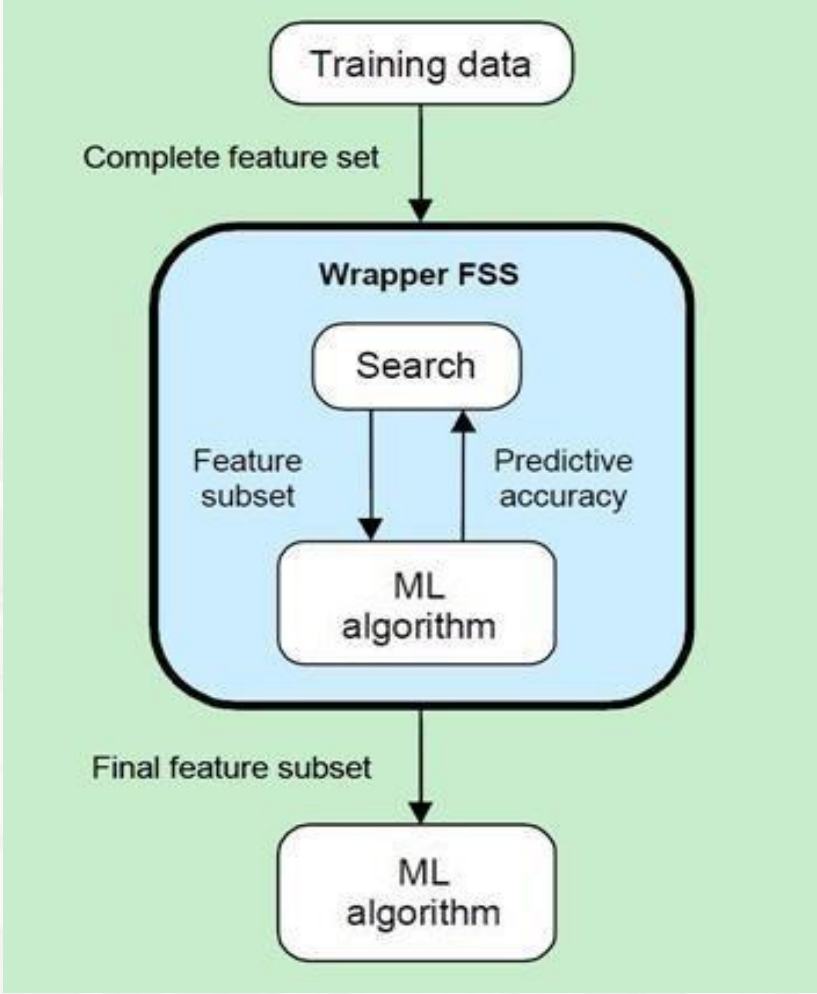
与过滤式特征选择不考虑后续学习器不同，包裹式特征选择直接把最终将要使用的学习器的性能作为特征子集的评价准则. 换言之，包裹式特征选择的目的就是为给定学习器选择最有利于其性能、“量身定做”的特征子集.



特征选择

- 包裹式

包裹式特征选择的实现与“基于整个特征子集的评价标准”的过滤式特征选择的实现类似。在子集的生成方式中，一般用启发式搜索或随机搜索。只是在特征子集的评估的时候将利用基于某个特征子集学习后续任务的**模型**的性能作为特征子集的评价准则。



特征选择

- 包裹式

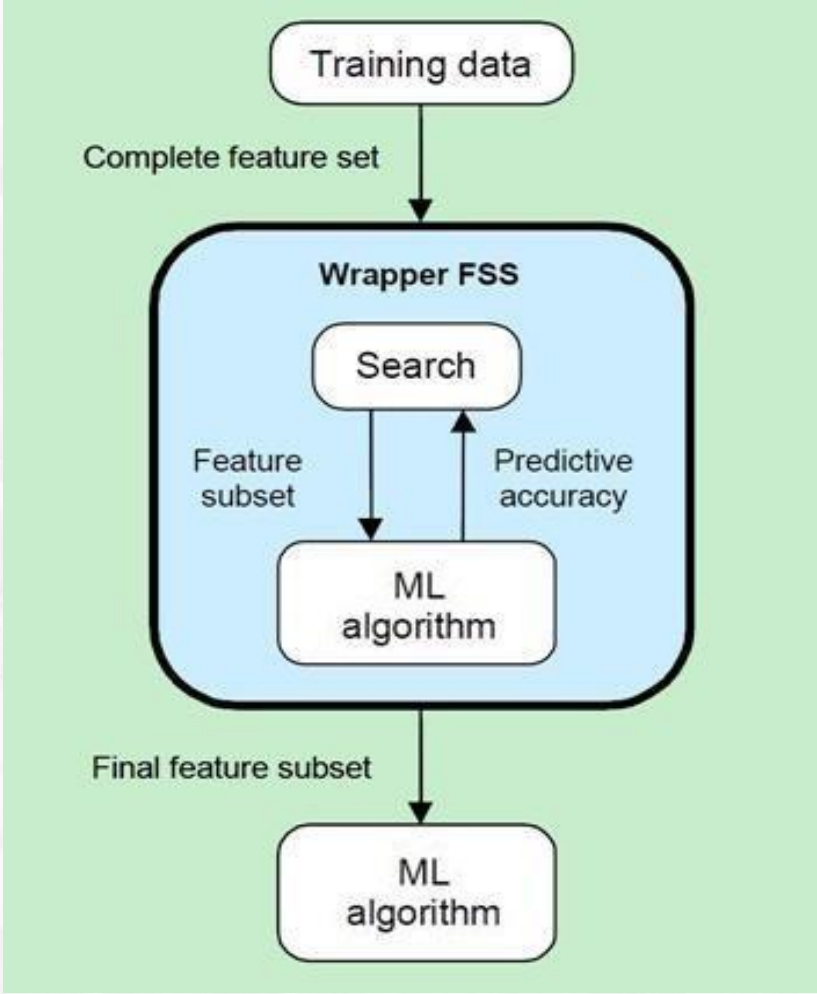
- Las Vegas Wrapper （LVW)

给定：学习算法 $\mathfrak{L}$；数据集 D；特征集 A；
停止条件控制参数 T
求：最优子集 A^*

$E = \infty; d = |A|; A^* = A$
$t = 0$
while $t < T$ do
　随机产生特征子集 A′;
　$d' = |A'|$
　$E' = \text{CrossValidation}\left(\mathfrak{L}\left(D^{A'}\right)\right)$
　if $(E' < E) \vee \left((E' = E) \wedge (d' < d)\right)$ do
　　$t = 0, E = E', d = d', A^* = A'$
　else do $t = t + 1$

直接用算法评价（而不是设计种种评价指标或方法）

算法表现更好了，或者没有变得更差/计算成本变低了，更新——直到达到最大迭代次数



特征选择

◦ 包裹式

- Recursive Feature Elimination (RFE)

初始化：

当前特征子集向量：$\mathbf{s} = [1,2,\cdots,k]$

特征排序向量：$\mathbf{r} = []$

重复以下过程直到$\mathbf{s} = []$：

在当前$\mathbf{s}$下表达数据：$\mathbf{X}(:,\mathbf{s})$

训练模型，得到能够体现特征重要性的信息，例如：

$SVM\text{train}(\mathbf{X},\mathbf{y}) \rightarrow \mathbf{w}$, 特征重要性：$c_i = (w_i)^2$

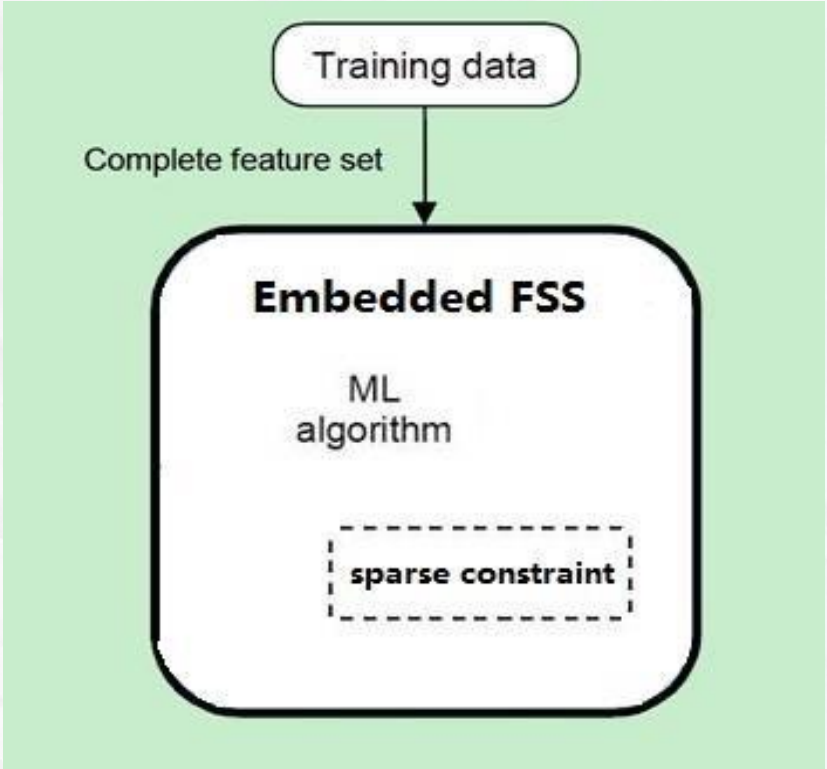
将得分最小的特征放入特征排序向量$\mathbf{r}$中，并将其从$\mathbf{s}$中剔除

特征选择

- 嵌入式

嵌入式特征选择是将特征选择过程与学习器训练过程融为一体，两者在同一个优化过程中完成，即在学习器训练过程中自动地进行了特征选择.

例如我们可以利用LASSO，决策树算法的模型来得到特征的重要性度量，从而进行特征选择



MASTER: Market-Guided Stock Transformer for Stock Price Forecasting, AAAI 2024, SJTU

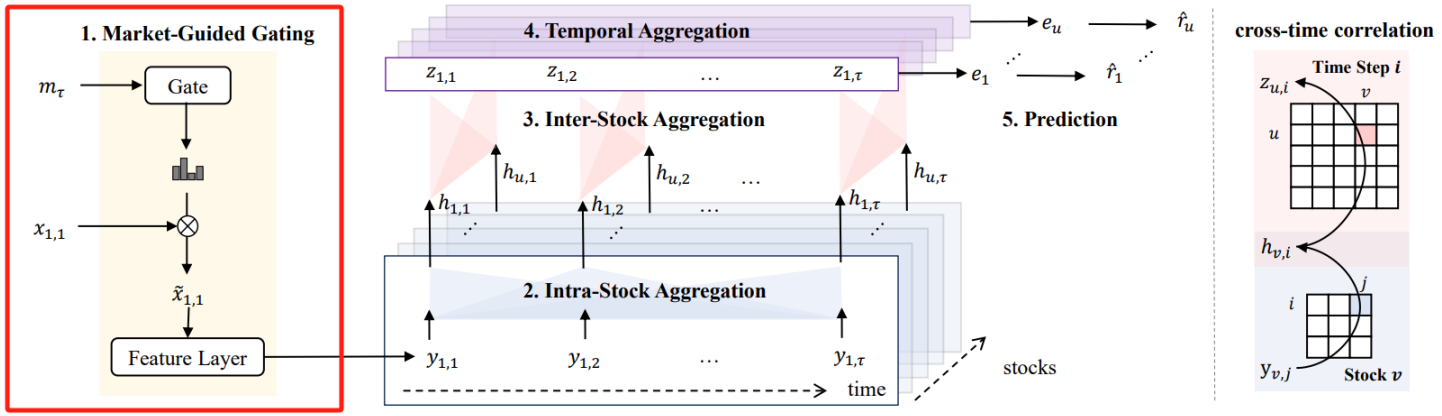


Figure 2: Overview of the MASTER framework.

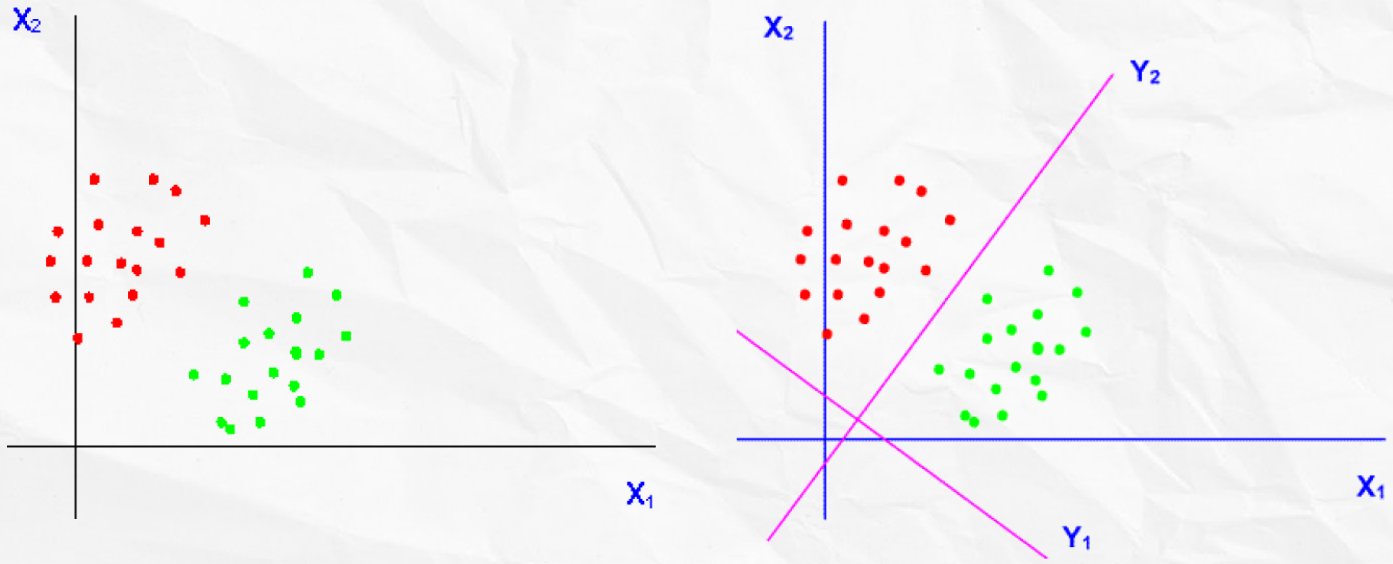
```python
class Gate(nn.Module):
    def __init__(self, d_input, d_output,  beta=1.0):
        super().__init__()
        self.trans = nn.Linear(d_input, d_output)
        self.d_output =d_output
        self.t = beta

    def forward(self, gate_input):
        output = self.trans(gate_input)
        output = torch.softmax(output/self.t, dim=-1)
        return self.d_output*output
```

- Gate input就是market信息，对齐到输入x的维度
- 过softmax，假设有10个特征，如果gate什么也没学到（换句话说10个特征重要程度都是一样的），概率分布就是[0.1, 0.1, ..., 0.1]
- 最后gate输出的是self.d_output * output，即10 * [0.1, 0.1, ..., 0.1] = [1, 1, ..., 1]，然后用这个概率element-wise production到原始输入特征
- 即如果gate什么都没学到，经过gate变换后仍是原始输入，如果gate学到了信息，那么输入也会发生相应变化

特征转换

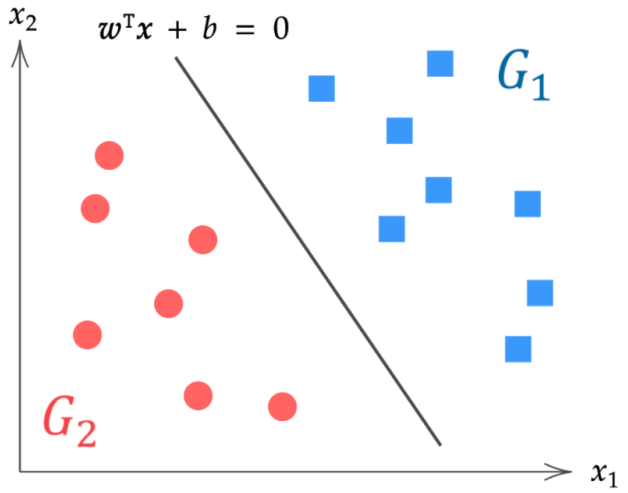
- 思路

  通过某种数学变换将原始高维属性空间转变为一个低维“子空间”，在这个低维嵌入子空间中更容易进行学习.

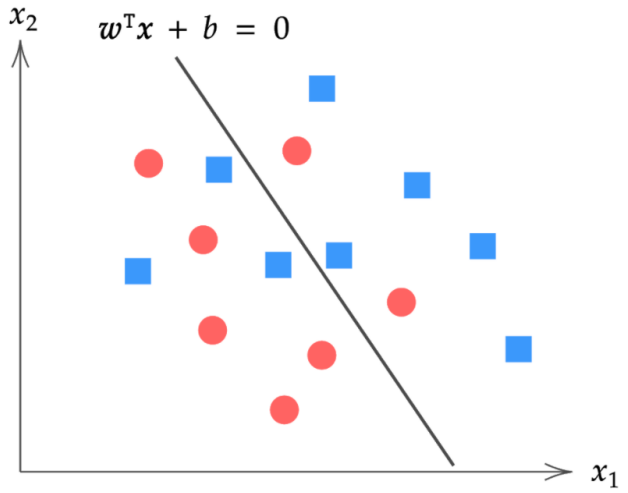


特征转换

線形分離可能



線形分離不可能



怎么办？学一个映射函数（非线性），把右图的特征映射到三维，然后用线性超平面就可分了！

特征转换

PCA

- “最重要的特征”通常指的是那些能够解释数据最大变异性的特征，并通过保留最大的方差成分，通常会剔除掉一些噪声特征
- 转换后的特征是正交的，即彼此不相关，因此可以比较好处理多重共线性问题（线性回归不满秩没有逆矩阵，有多个解）

LDA

- 旨在找到最佳的投影方向，以**最大化类间差异**和**最小化类内差异**，常用于分类问题

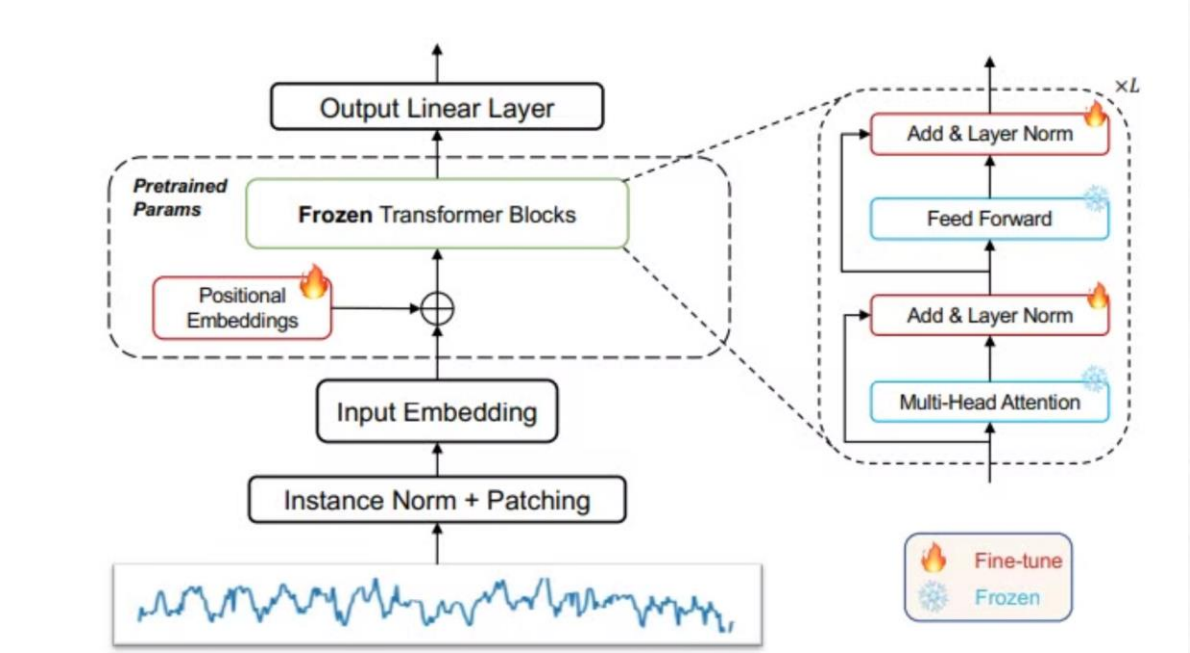
AutoEncoder

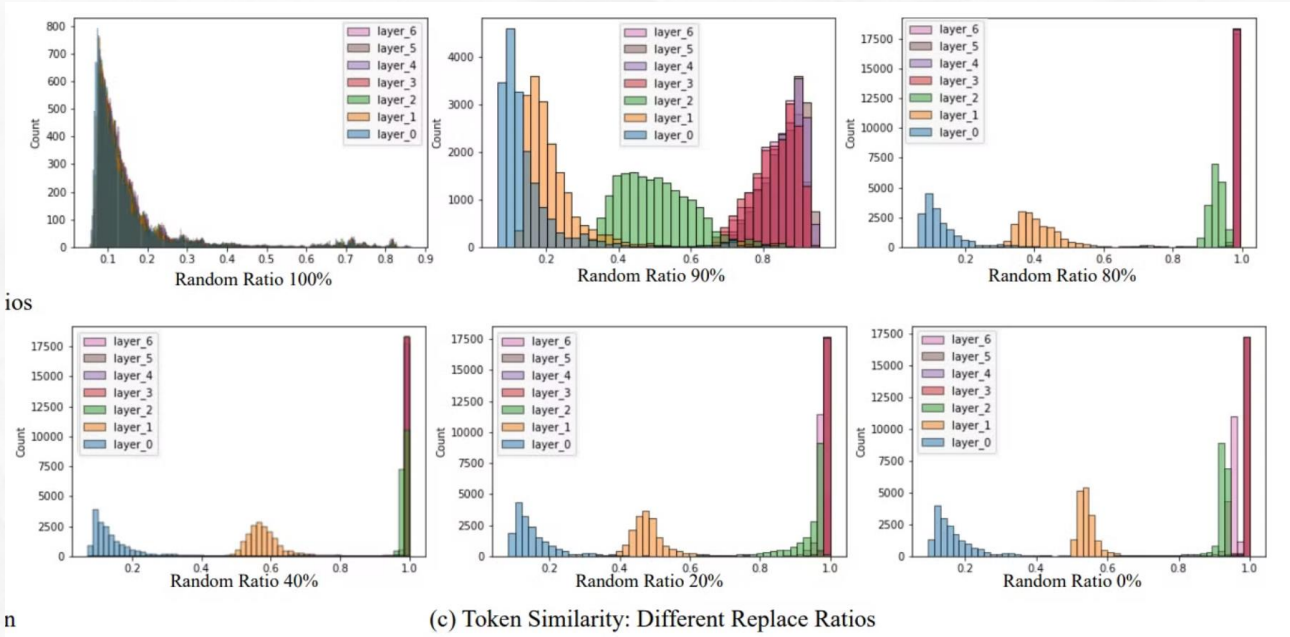
- 异常检测里面提过，无监督任务的特征提取方法

Word Embedding

- 常用在NLP领域

One Fits All: Power General Time Series Analysis by Pretrained LM, NeurIPS 2023, DAMO



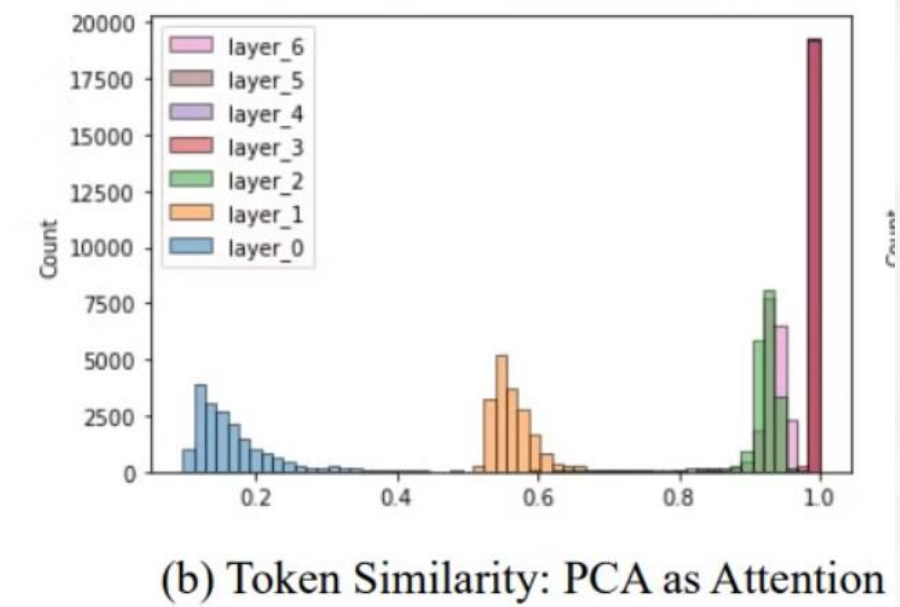


(c) Token Similarity: Different Replace Ratios

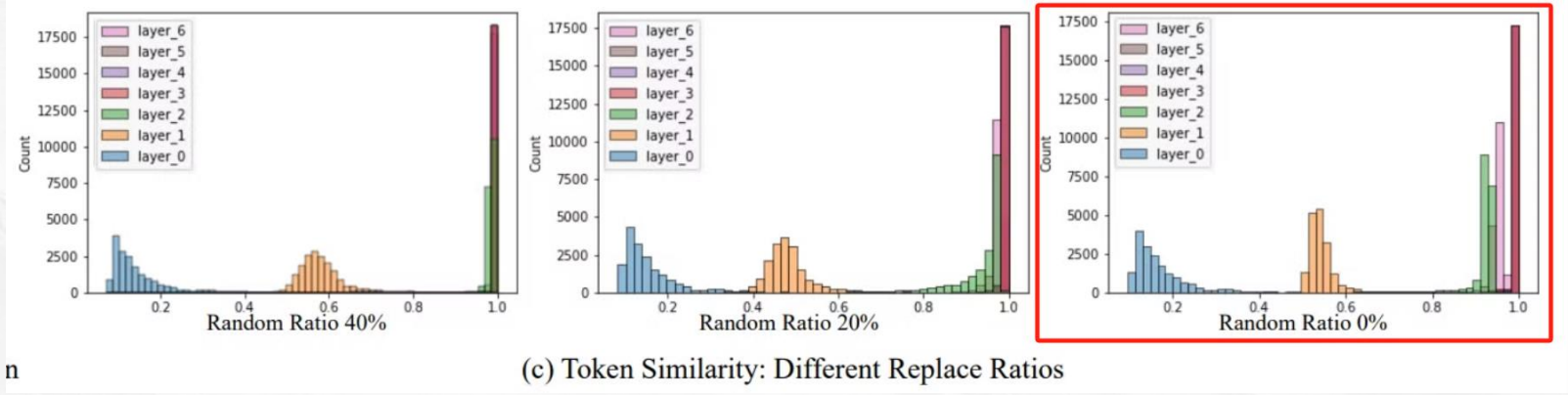
作者分析了token（也就是patching后的series）之间的相似度，对于模型每一层输出都计算token之间两两相似度（有点类似于attention map）

- 全部随机初始化时，token之间相似度非常低
- 完全用预训练权重时（Random Ratio=0%），在较高layer层数上相似度明显非常高，说明token向量都被投影到了输入的低维特征向量空间中。类似于curse of dimension，维度越高计算相似度高的概率指数越低

特征转换



(b) Token Similarity: PCA as Attention



n (c) Token Similarity: Different Replace Ratios

作者将self-attention替换为PCA，对应子图(b)，结果和Random Ratio=0%，即预训练权重下的self-attention结果非常像，说明两者作用比较相似，也就是类似于PCA，预训练好的self-attention建模各种模态数据具有一定的通用性