

# Practice of AI

C2: Machine learning & Data analyze

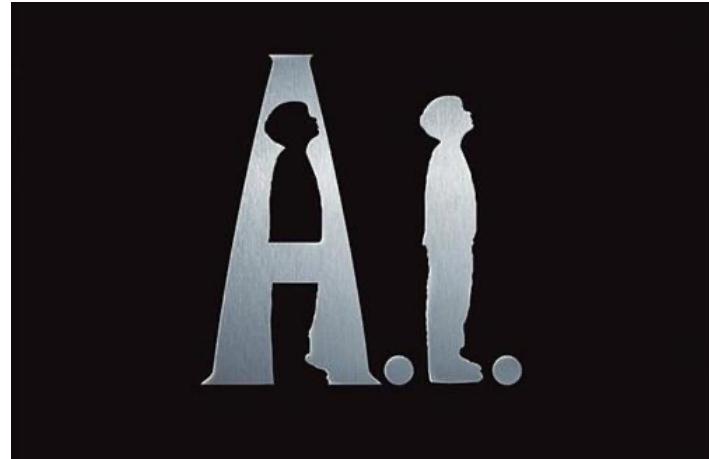
Jim Xie

2020/10/6



# Goal

---



ML剩下部分：  
特征工程，模型选择

# Preface

---

 $\Sigma$  $\Delta$  $\Omega$  $\epsilon$  $\omega$  $\Psi$  $\theta$  $\gamma$

# 特征工程

---

## 特征工程的作用与方法

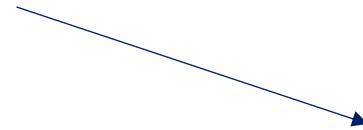
- ❖ 从不同角度提炼一些有价值的特征
- ❖ 不能自动化，无法替代专家角色



# 为什么需要降维？

---

特征过多坏处



1. 计算缓慢
2. 模型复杂
3. 增加干扰项
4. 造成维度灾难



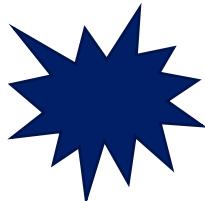
维度灾难指什么？

# 维度灾难：现象

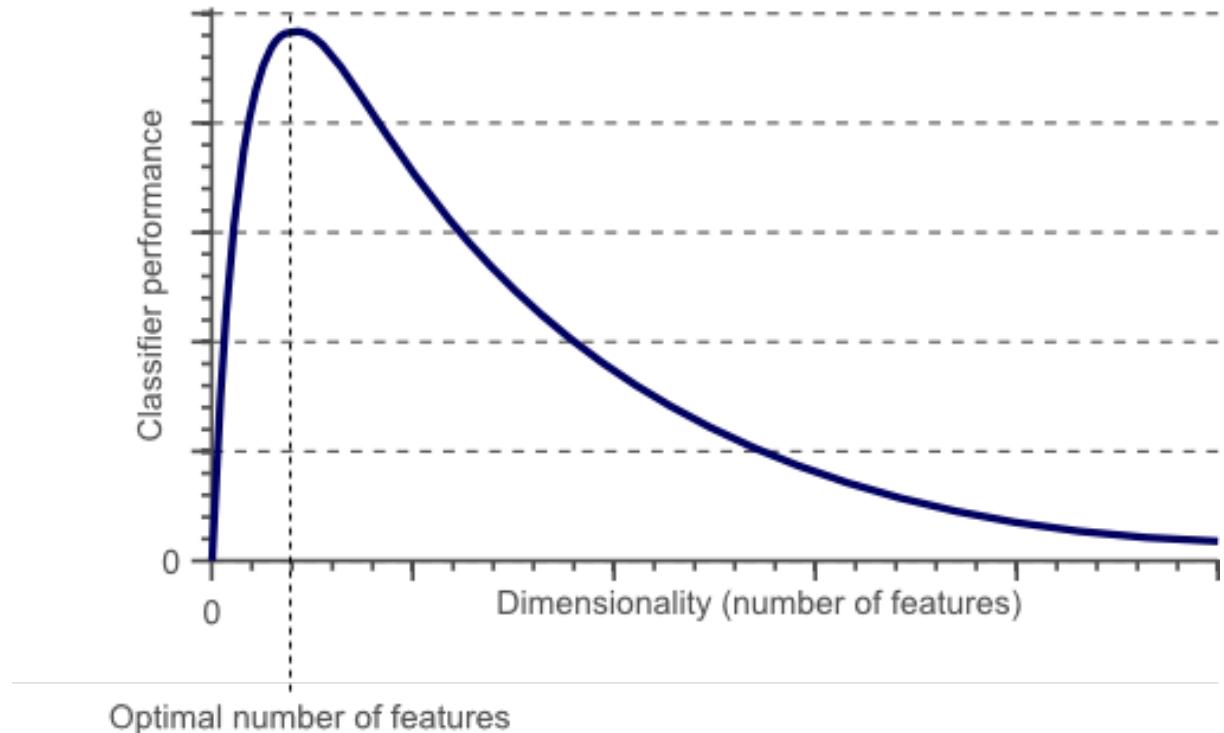
高维情况下，很多模型表现会急剧下降

猫狗识别的例子

模型性能在不同特征数量下的表现



我应该选择多少个特征呢？



<http://127.0.0.1:8888/notebooks/C3/Traffic-Train/traffic-sign.ipynb>

# 维度灾难：背景

人脸比对方法：<http://localhost:8888/notebooks/C3/face/demo.ipynb>

```
[ 6.45104647e-02,  2.84561459e-02, -7.34514371e-02, -1.72763225e-02,
  1.13835242e-02, -1.63212884e-02, -5.34696281e-02,  5.32028750e-02,
 -1.44749098e-02,  3.59034464e-02,  1.85653958e-02,  5.90951927e-02,
  1.47780543e-02, -2.17742771e-02,  1.50234271e-02,  2.69742161e-02,
  9.79379192e-03,  6.51236475e-02, -4.73314198e-03,  1.31187811e-02,
  1.93645861e-02,  7.57417455e-02, -2.74137277e-02,  1.94943510e-02,
 -5.75501844e-03,  7.68118026e-03, -1.67309050e-03, -4.68759052e-02,
 -7.83467572e-03,  1.86435506e-02,  6.26649186e-02, -2.79785935e-02,
  6.60447478e-02,  7.25395456e-02,  3.40714306e-02, -3.05157658e-02,
  2.18674112e-02, -3.11621651e-02,  6.31265417e-02,  1.90785695e-02,
 -3.54125351e-02, -8.98810178e-02, -8.88275821e-03,  8.27952754e-03,
 -7.42505789e-02,  6.32991940e-02, -5.85555136e-02, -3.11240144e-02,
  7.45614385e-03,  3.36035006e-02, -3.36385928e-02, -3.62228416e-02,
 -2.25111637e-02, -3.21121365e-02,  1.06237046e-02, -3.13168541e-02,
  2.29744464e-02, -7.73122311e-02, -2.71368353e-03, -2.71095615e-02,
 -3.94065753e-02,  1.24143705e-01, -3.52797844e-02,  3.50417607e-02,
 -2.03845110e-02, -8.30745846e-02, -1.34285409e-02, -1.15193380e-02,
  8.32579955e-02,  6.08327519e-03,  1.97861549e-02, -4.40394022e-02,
 -4.70835753e-02,  3.52543071e-02,  3.41317244e-02, -1.77111034e-03,
```

**D**

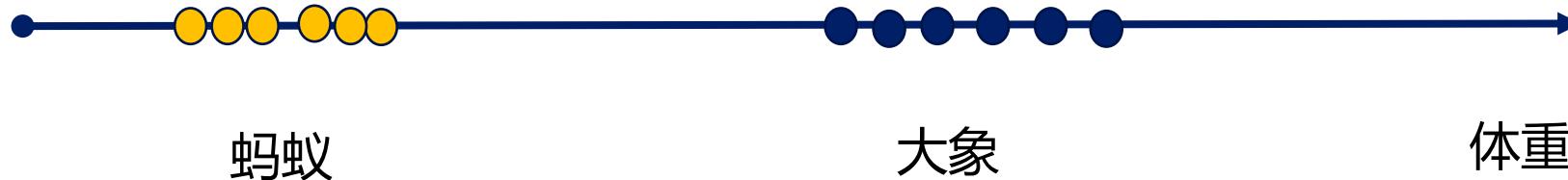
```
[ 7.10385218e-02, -3.22032999e-03, -1.24619424e-03,  1.91412363e-02,
 -5.42471372e-03,  4.97934222e-02, -4.62391647e-03,  2.44413707e-02,
 -4.37092967e-02,  1.28905633e-02,  4.58273627e-02,  1.60606683e-03,
 -3.24402517e-03, -3.11120655e-02,  5.20463549e-02,  2.78522763e-02,
  1.21331312e-01,  5.84986359e-02, -2.24856790e-02,  6.56152982e-03,
  2.72995494e-02,  2.95747090e-02,  6.11840189e-02, -5.89872478e-03,
 -1.83749925e-02,  1.91699155e-02,  2.81136055e-02,  1.05330022e-02,
  2.49693021e-02,  1.23054804e-02,  3.61388363e-02,  6.87964931e-02,
  1.34759527e-02,  4.35568206e-02,  2.19232421e-02, -7.88312331e-02,
 -6.39629960e-02, -6.75716326e-02, -5.15788188e-03,  1.35627938e-02,
  4.90935482e-02, -7.30361789e-02, -6.72618253e-03,  3.84790963e-03,
 -3.24839801e-02,  3.11814086e-03,  1.42189832e-02, -2.38011386e-02,
 -4.32611741e-02,  2.29026433e-02, -3.68273519e-02, -2.12987065e-02,
  4.82287668e-02, -1.86451513e-03,  2.75416374e-02, -2.81859729e-02,
  2.01441273e-02, -5.03548756e-02,  5.85663095e-02,  5.18572219e-02,
 -3.08055952e-02,  6.18212484e-02, -3.48523930e-02,  6.40194491e-02,
  2.44354028e-02, -1.25454785e-02, -1.40200751e-02, -9.67387408e-02,
  1.91903841e-02, -4.41153497e-02, -6.71385787e-03,  1.30572531e-03,
 -3.06758154e-02,  3.16419564e-02, -1.22185666e-02,  9.03905705e-02,
 -8.64211842e-03, -4.80578952e-02,  9.80059337e-03, -3.30699869e-02,
  5.56746162e-02, -5.78230619e-02, -1.10484585e-01,  1.57824636e-03,
```

1. 计算距离得到相似度 **D** →

2. 根据相似度 **D** 判断是哪个人

# 维度灾难：一维

$D$  = 体重

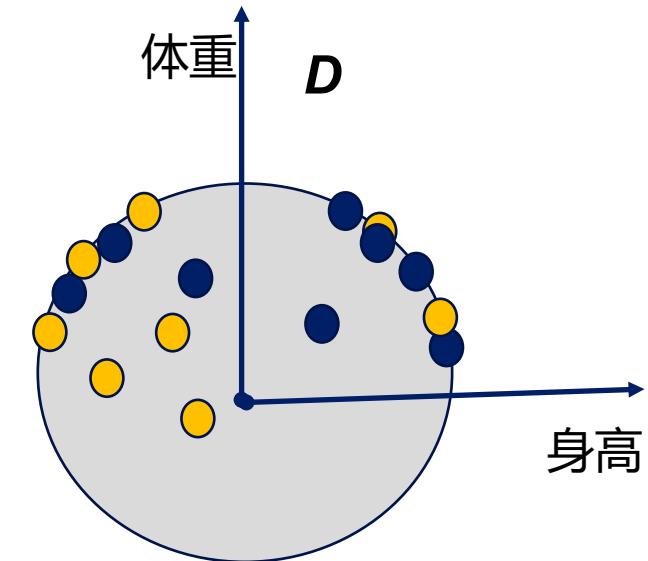


- 最好情况：  
 $D$ 对应到每一个个体，通过 $D$ 可以确定每一只蚂蚁或大象  
可能存在冲突
- 可接受情况：  
 $D$ 对应到不同的类别，通过 $D$ 范围确定类别  
几乎没有冲突

## 维度灾难：二维

$$D = \sqrt{\text{体重}^2 + \text{身高}^2} = \sqrt{\text{身高}^2 + \text{体重}^2} = \dots$$

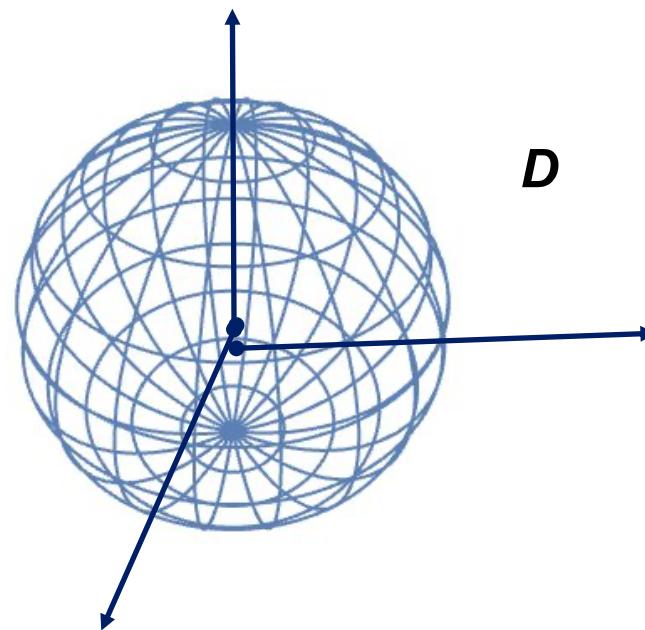
- 最好情况：  
 $D$ 对应到每一个个体  
存在较多冲突
- 可接受情况：  
 $D$ 对应到不同的类别，通过 $D$ 范围确定类别  
冲突并没有减少



## 维度灾难：三维

$$D = \sqrt{a^2 + b^2 + c^2} = \sqrt{c^2 + b^2 + a^2} = \dots$$

同样的道理，维度越高，数据碰撞概率越大



# 维度灾难

---

□ 维度越高，数据碰撞概率越高

□ 我应该选择在多少个特征呢？

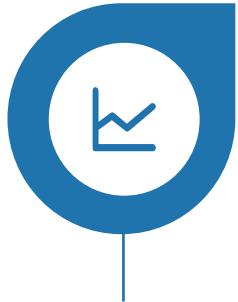
- 样本的数据分布
- 选取的特征辨识度
- 计算的方法
- .....

□ 如果把维度降到1维，会怎样？

<https://setosa.io/ev/principal-component-analysis/>

# 解决方法：特征工程

就是找到合适的特征组合



Filter过滤法

根据特征重要程度，过滤掉一些没用的特征，  
主要方法：方差、相关系数等。



Wrapper包装法

穷举全部特征组合，评估包括训练器的综合性能，得出最佳组合。



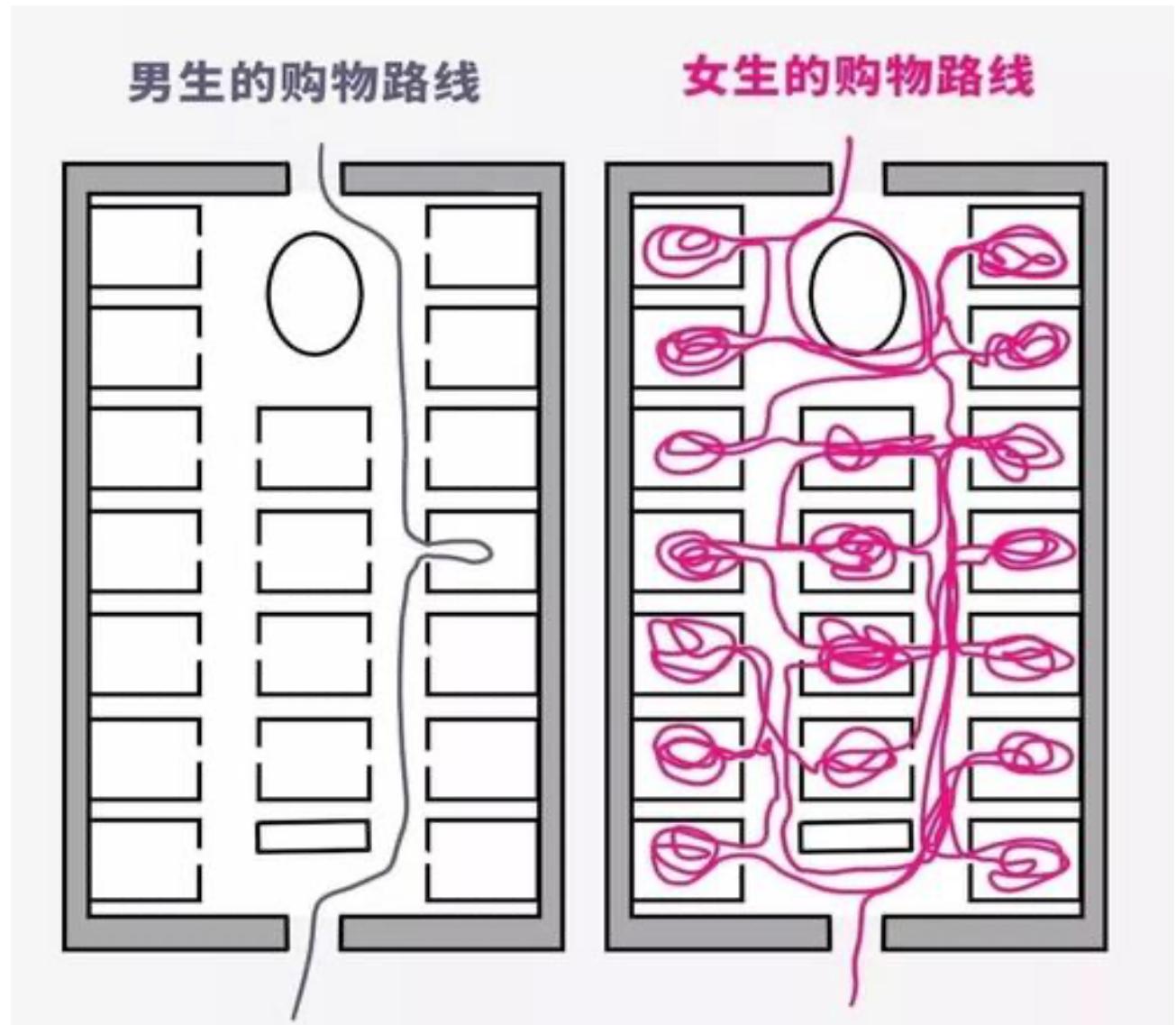
Embedded嵌入法

输入全部特征，在学习模型的过程中，挑选出那些对模型训练有重要性的特征。

# 举例

## □ 男女生逛商场

- 已有每个人的位置坐标数据；
- 数据量相同，都停留1个小时；
- 需要分析摊位的受欢迎程度；
- 优先选择哪个数据进行分析？



# 方差法

衡量数据的稳定性（变化程度）

$$Var(x) = \frac{\sum(x_i - \bar{x})^2}{n}$$

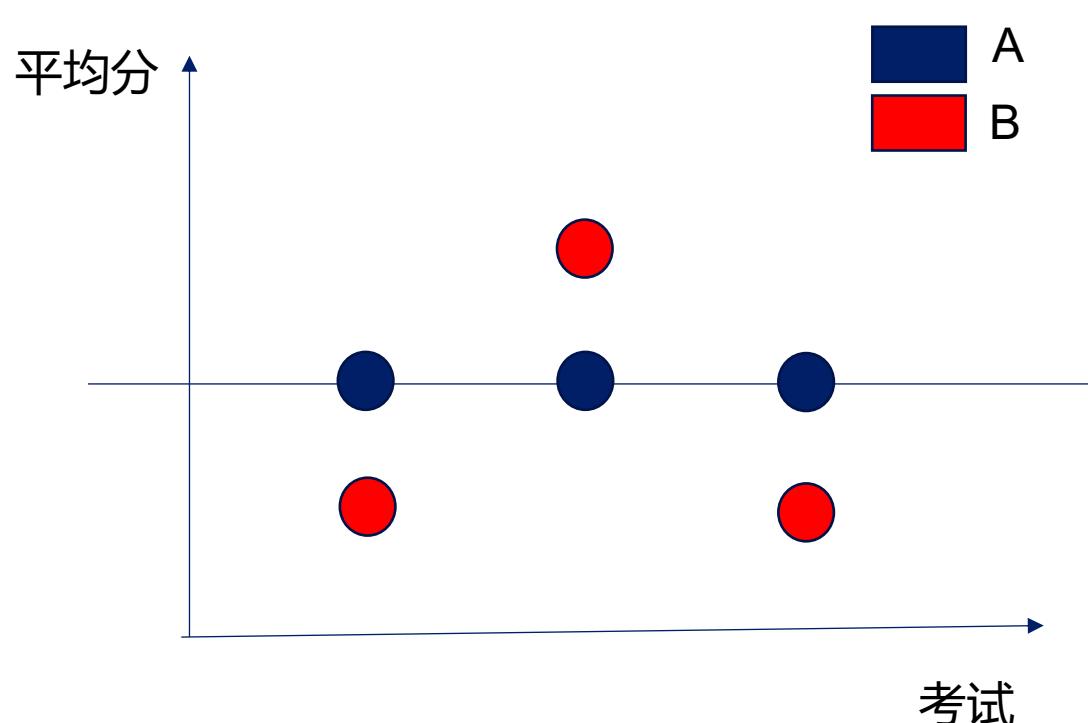
考试成绩

学生	考试1	考试2	考试3	平均分
A	90	90	90	90
B	85	90	95	90

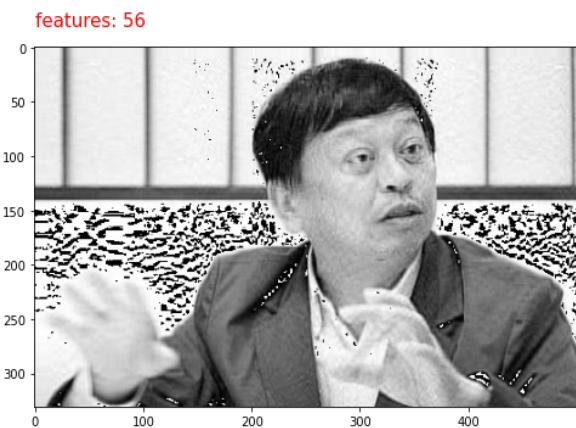
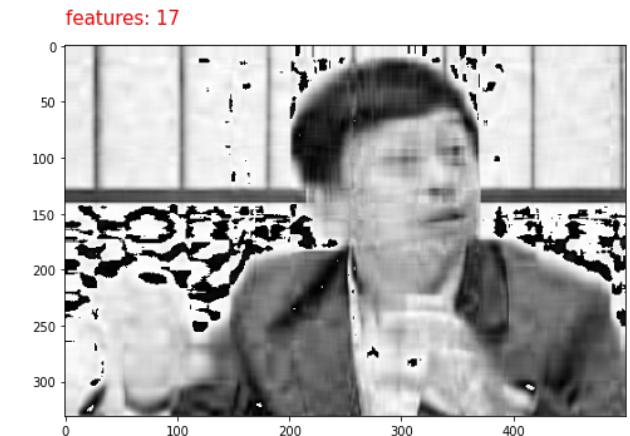
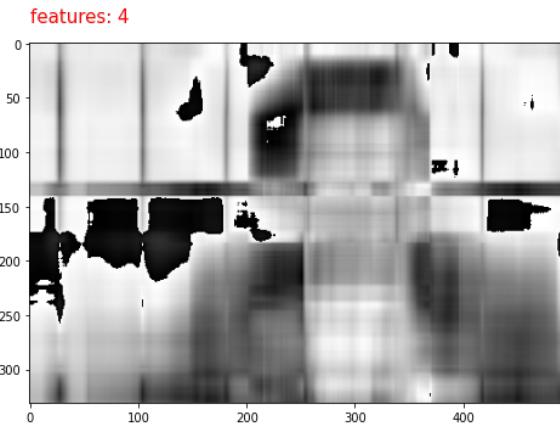
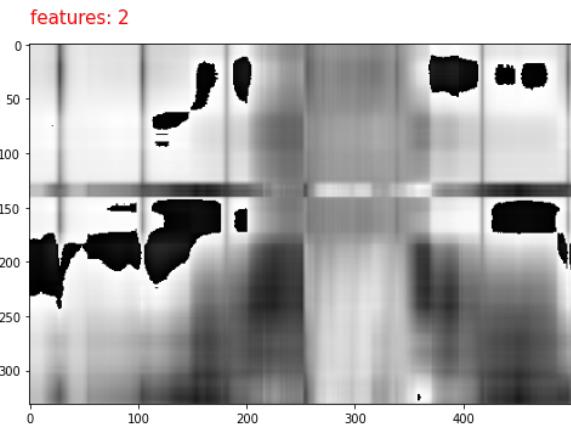
A的方差是 0

B的方差是  $50/3$

} 学生B包含的信息更多  
(特征发散)



# 降维后对比



# 根据方差选特征

对样本方差计算 : df.var()

QQ: 这样做是有问题吗 ?

	Height	Weight	Level
495	148	155	5
496	146	157	5
497	140	152	5
498	145	160	5
499	142	159	5

df.var()

Height 268.15  
Weight 1048.63

weight特征更发散，优先选择

	Height_scaled	Weight_scaled
495	-0.73	0.91
496	-0.80	0.95
497	-1.00	0.85
498	-0.83	1.00
499	-0.93	0.98

df.var()

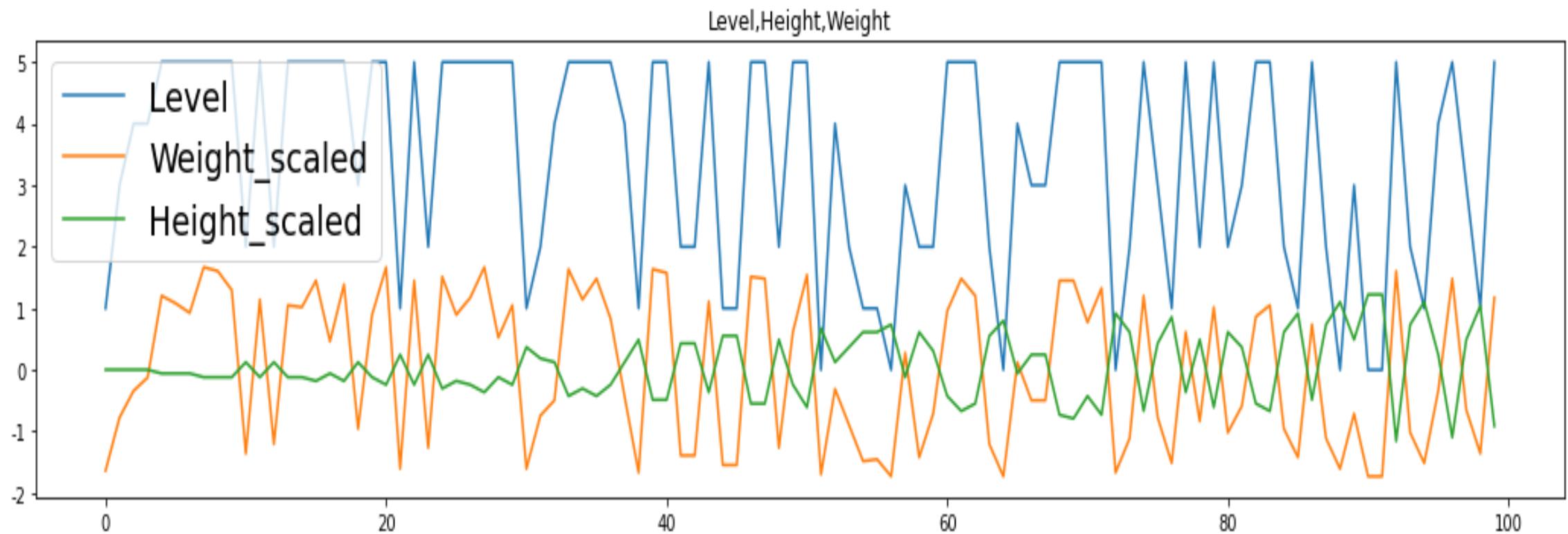
Height\_scaled 0.31  
Weight\_scaled 0.35

# 根据关联度选特征

1. 将Level , Height , Weight放到一张图上 (如下图) ;
2. Weight/Height关联度哪个大



如何度量这种关联程度 ?



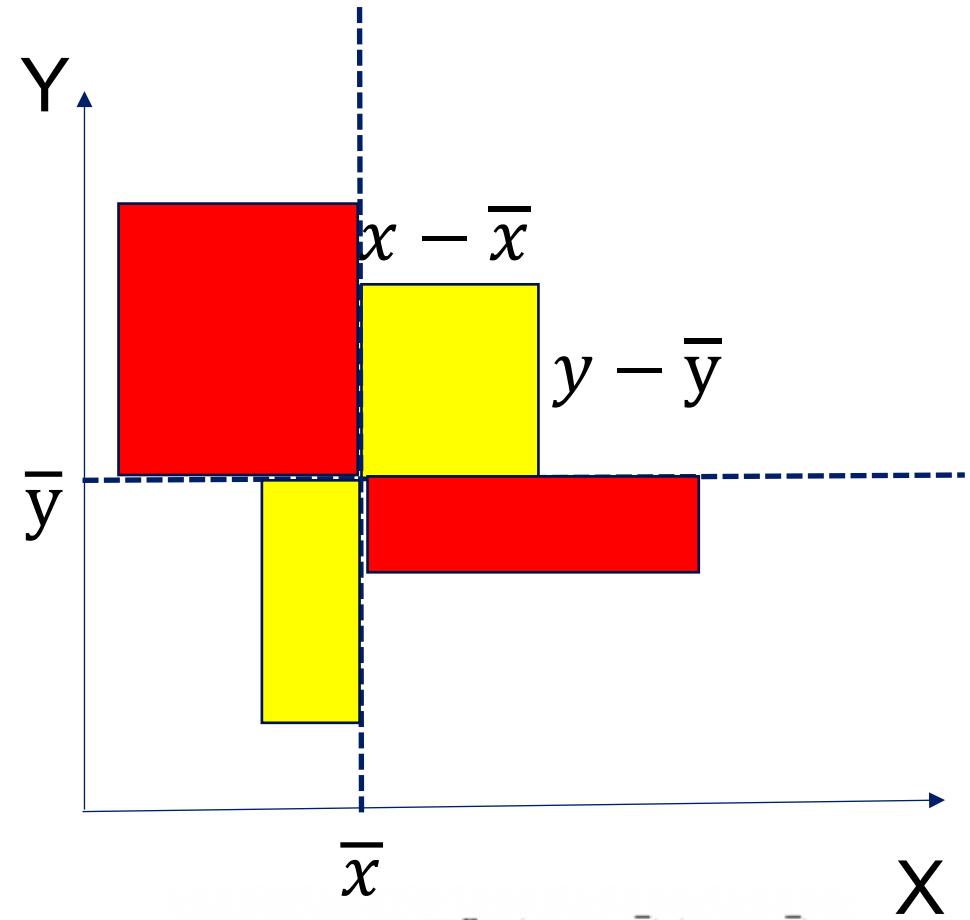
# 协方差

## 衡量关联度

- > 0 : X,Y 变动方向相同，正相关
- < 0 : X,Y 变动方向不同，负相关
- = 0 : X,Y 独立变化，不相关

绝对值大有两个原因

- ①: 共振
- ②: X,Y自身振动大



$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{n - 1}$$

# 相关系数

---

## 衡量两个变量的相关度

从协方差中剔除X,Y自身波动带来的影响

$$\frac{\text{协方差}}{\text{X的标准差} * \text{Y的标准差}} = \rho_{x,y} = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \text{Var}(Y)}}$$



大于0  
正相关

等于0  
不相干

小于0  
负相关

# 根据相关系数选特征

---

相关系数法： df.corr()

	<b>Height</b>	<b>Weight</b>	<b>Level</b>
<b>Height</b>	1.00	0.00	-0.42
<b>Weight</b>	0.00	1.00	0.80
<b>Level</b>	-0.42	0.80	1.00

取最大，取特征Weight

# 卡方检验

---

## 基本思路

1. 首先假设A、B不相关；
2. 然后根据A和B的联合概率分布，计算卡方值；
3. 卡方值越大，越倾向于推翻原假设，也就是A、B越相关；

举例：家庭收入是否会影响孩子考上985高校？

年收入区间	0 ~ 10万	10万 ~ 15万	15 ~ 20万	20万以上
家庭数量	100	100	100	100
985高校	10	11	15	13
普通高校	90	89	85	87

# 卡方分布

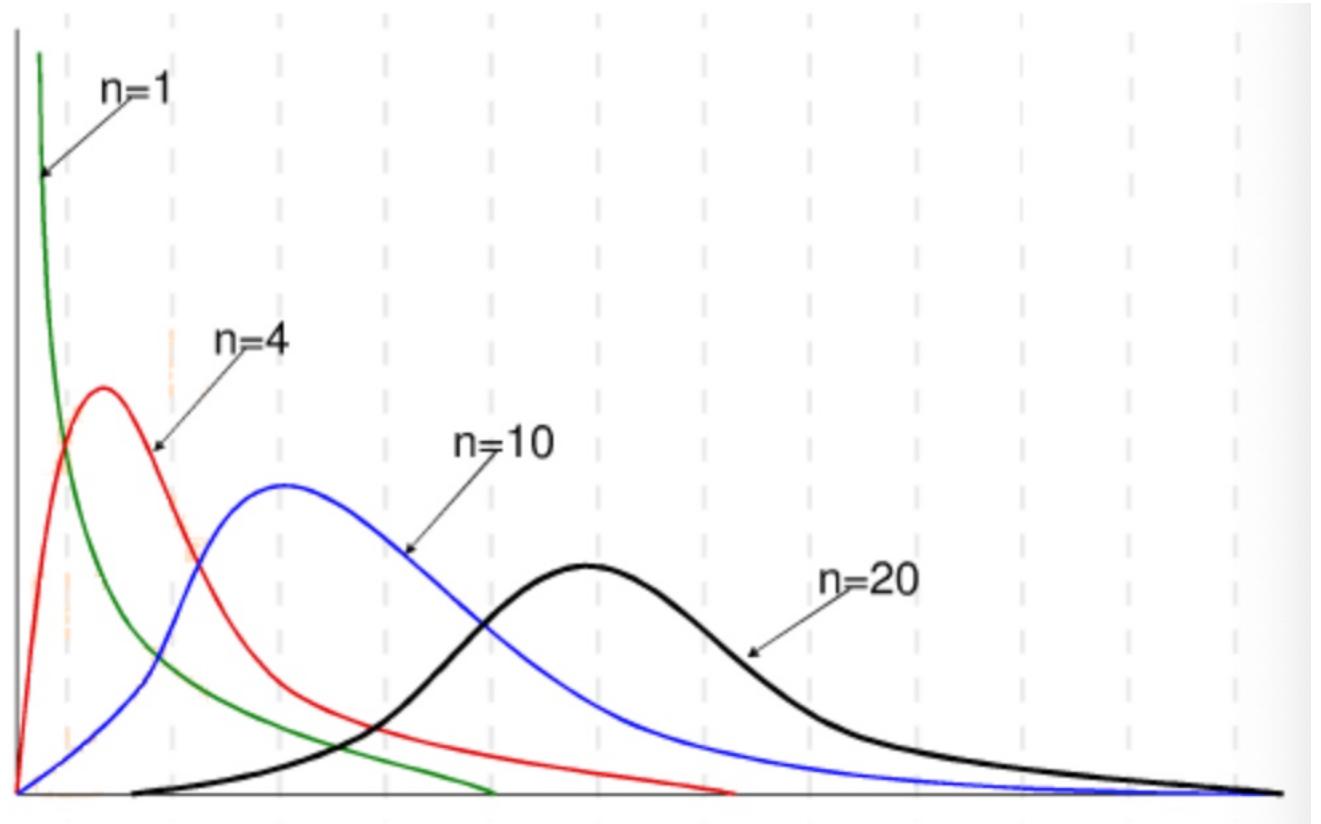
$$\text{公式 : } X = \sum_{i=1}^n X_i^2$$

$\chi_i$ :

- 服从标准正态分布
- 独立随机变量

N :

- 自由度



# 卡方检验

---

H0 : 收入和上985无关

H1 : 收入和上985有关

如果H0成立，上985和上普通高校的比例应该是不变的

- 上985高校的比例 = 上985人数/总人数 = 12.25%
- 上普通高校比例 = 上普通高校人数/总人数 = 87.75%

## 上985高校的理论分布

年收入区间	0 ~ 10万	10万 ~ 15万	15 ~ 20万	20万以上
家庭数量	100	100	100	100
985高校	12.25	12.25	12.25	12.25
普通高校	87.75	87.75	87.75	87.75

# 卡方检验

$$\text{计算卡方值: } X^2 = \sum \frac{(\text{观察值} - \text{理论值})^2}{\text{理论值}}$$

	年收入区间	0~10万	10万~15万	15~20万	20万以上
观察值	家庭数量	100	100	100	100
	985高校	10	11	15	13
	普通高校	90	89	85	87
理论值	985高校	12.25	12.25	12.25	12.25
	普通高校	87.75	87.75	87.75	87.75
$\Delta$	985高校	0.41	0.13	0.62	0.03
	普通高校	0.06	0.02	0.09	0.01
$X^2$	$\text{sum}(\Delta) = 1.37$				
p_value	0.71. ( <a href="https://mlln.cn/2020/10/10/P值显著性计算器/">https://mlln.cn/2020/10/10/P值显著性计算器/</a> )				

结论：H0不成立，家庭收入与孩子上985没关系

# 卡方检验

样本中P值：身高: 0.95 , 体重 : << 0.05

```
1 from sklearn.feature_selection import SelectKBest
2 from sklearn.feature_selection import chi2
3
4 data = []
5 target = []
6 for h,w,t in zip(df2['Height'].tolist(),df2['Weight'].tolist(),df1['Level'].tolist()):
7     data.append([h,w])
8     target.append(t)
9 data = np.array(data)
10 target = np.array(target)
11 model = SelectKBest(chi2, k=2)
12 new_features = model.fit_transform(data,target)
13 print("Height:%.8f"%model.pvalues_[0],"Weight:%.12f"%model.pvalues_[1])
```

Height:0.94938629 Weight:0.00000000026

# 小结

---

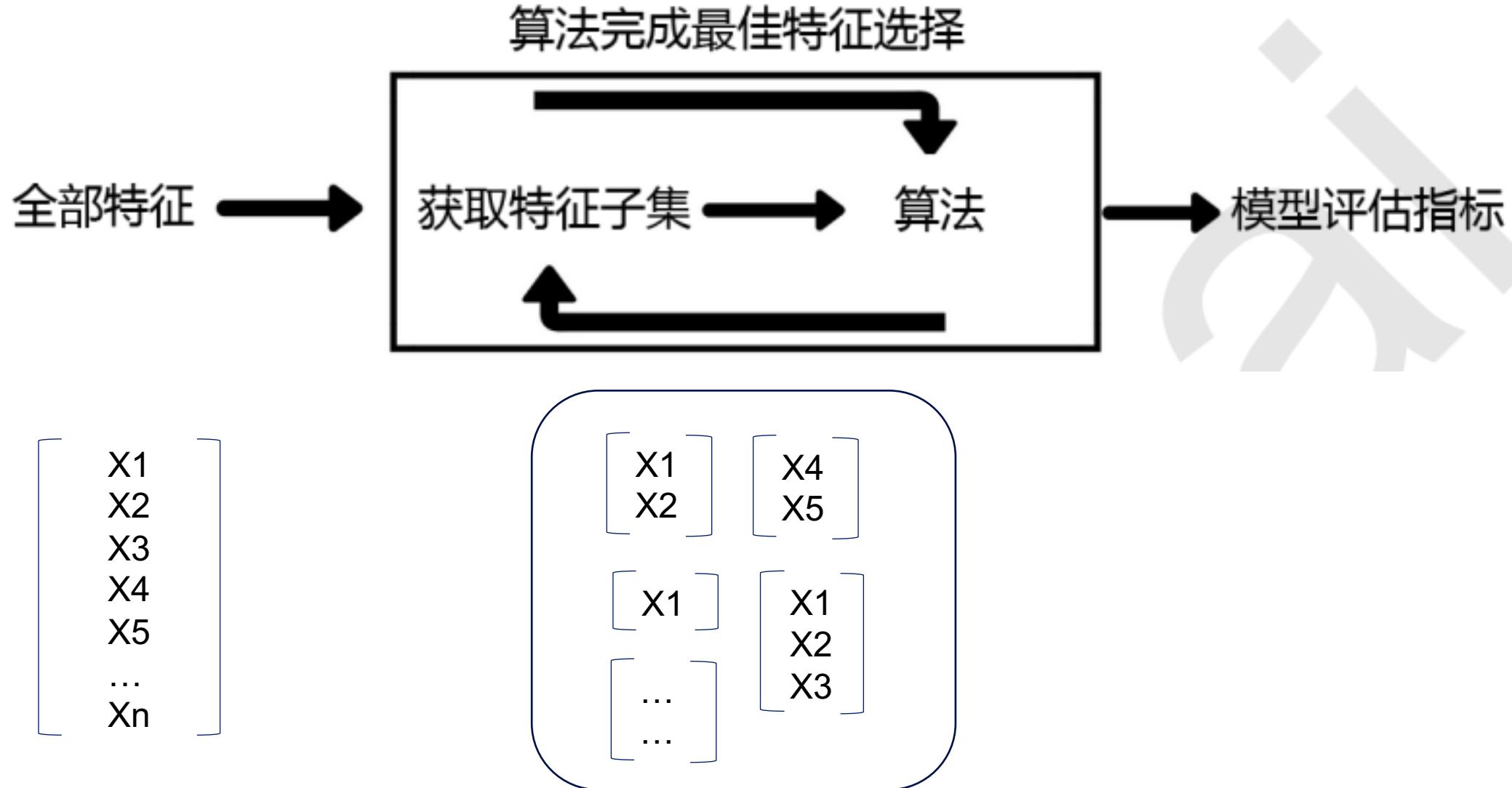
□ 维度灾难

□ 方差法

□ 关联系数法

□ 卡方检验法

# Wrapper包装法



# Embedded

---

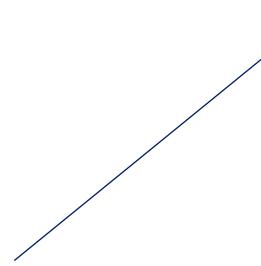
## 举例：

- 现有一个task，领导要求评估effort；
- 评估方法：总effort = 研发人数\*develop 时间 + 测试人数\*test时间；
- 实际上，项目不同，需要的RD或QA的数量也不同
- 如何设计改进评估方法，弹性更好，效率更高？

# Embedded

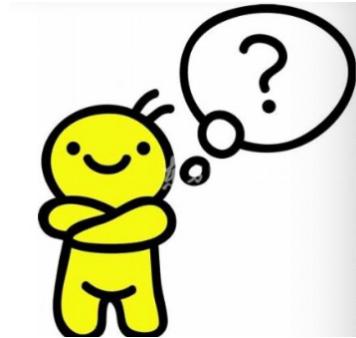
---

- 总effort = 研发人数\*develop 时间 + 测试人数\*test时间；
- 原来算法不变，同时要求研发人数 + 测试人数 最小；



在学习模型的过程中，利用正则化思想，将部分特征属性的权重变成零。常见的正则化有L1的Lasso，L2的Ridge和混合的Elastic Net。

什么是正则化，Ridge, LASSO, Elastic Net ?



# 正则化

举例：考试成绩

学生	考试1	考试2	考试3	平均分
A	90	90	90	90
B	85	90	95	90

学生A的成绩更稳定

模型  $Y = W_1 * X_1 + W_2 * X_2 + W_3 * X_3$

模型	$W_1$	$W_2$	$W_3$	准确率
A	90	90	90	95%
B	85	90	95	95%

模型A的表现更稳定

使用正则项( $W_1^n + W_2^n + W_3^n$ 表示稳定性)

- $\sum W_i^n$  越小越稳定 (n为自然数)
- n=1 称L1正则
- n=2 称L2正则

# 线性回归系列

---

线性回归模型  正则项

(Ridge , LASSO , Elastic Net)

# 总结：三化的常用场景

1 归一化

数据探索阶段

消除数据在  
量级上差别

2 标准化

特征工程阶段

让数据特征  
更容易显现

3 正则化

模型训练阶段

让数据拟  
合更稳定

# 小结

---



- 维度灾难
- 特征提取方法
- 方差、协方差与相关系数
- 正则化思想
- 线性模型的扩展

# 模型介绍

---

- 线性回归系列

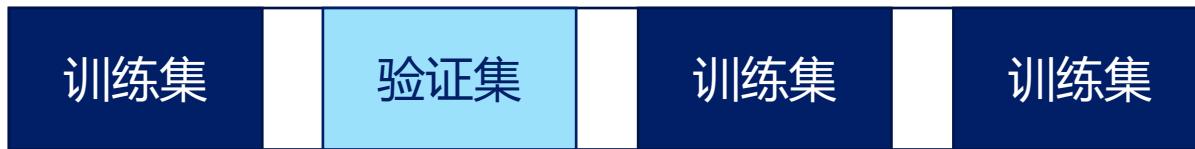
- Linear
- Ridge
- LASSO
- Elastic Net
- 多项式回归

- 决策树系列

- Decision tree
- Random forest
- Bagging
- Adaboost
- GBDT
- XGBoost

# K-Fold 交叉验证

- 训练集 = 训练集 + 验证集
- 测试集 = 测试集



80%

90%

95%

85%

87.5%

# 网格搜索调参

```
from sklearn.model_selection import GridSearchCV

def GridTrain(name,template,param_dict):
    estimator = GridSearchCV(template,param_grid=param_dict, cv=4)
    estimator.fit(X,Y)
    score = estimator.score(X_test, Y_test)
    model = estimator.best_estimator_
    plt.text(-2.5,2,'model:%s\nscore: %.2f\n%s'%(name,score,model),fontdict={'size': 15, 'color': 'red'})
    plot_predict_curve(estimator,X_test, Y_test)

%matplotlib inline
plt.style.use({'figure.figsize':(12, 12)})

estimator = SVC()
fig,ax = plt.subplots(1,1,figsize=(12, 12))
param_dict = {"C":[3,9,15,20,25,30,35]}
GridTrain("SVM",estimator,param_dict)
plt.show()

dt = DecisionTreeClassifier(criterion='entropy',max_depth=15,min_samples_split=5)
fig,ax = plt.subplots(1,1,figsize=(12, 12))
param_dict = {"max_depth":[5,9,15,20,25,30,35]}
GridTrain("DT",dt,param_dict)
plt.show()

rf = RandomForestClassifier(max_depth=15, n_estimators=10,criterion='entropy',min_samples_split=5)
fig,ax = plt.subplots(1,1,figsize=(12, 12))
param_dict = {"max_depth":[5,9,15,20,25,30,35]}
GridTrain("rf",rf,param_dict)
plt.show()
```

## Evaluate # Regression

---

Error = | Real – Predict | 

$$RMSE(X, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2}$$

$$MSE = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

$$MAE(X, h) = \frac{1}{m} \sum_{i=1}^m |h(x_i) - y_i|$$

# Evaluate # Classify



## Confusion Matrix

	Predicted (Positive)	Predicted (Negative)
Actual (Positive)	TP	FN
Actual (Negative)	FP	TN



$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

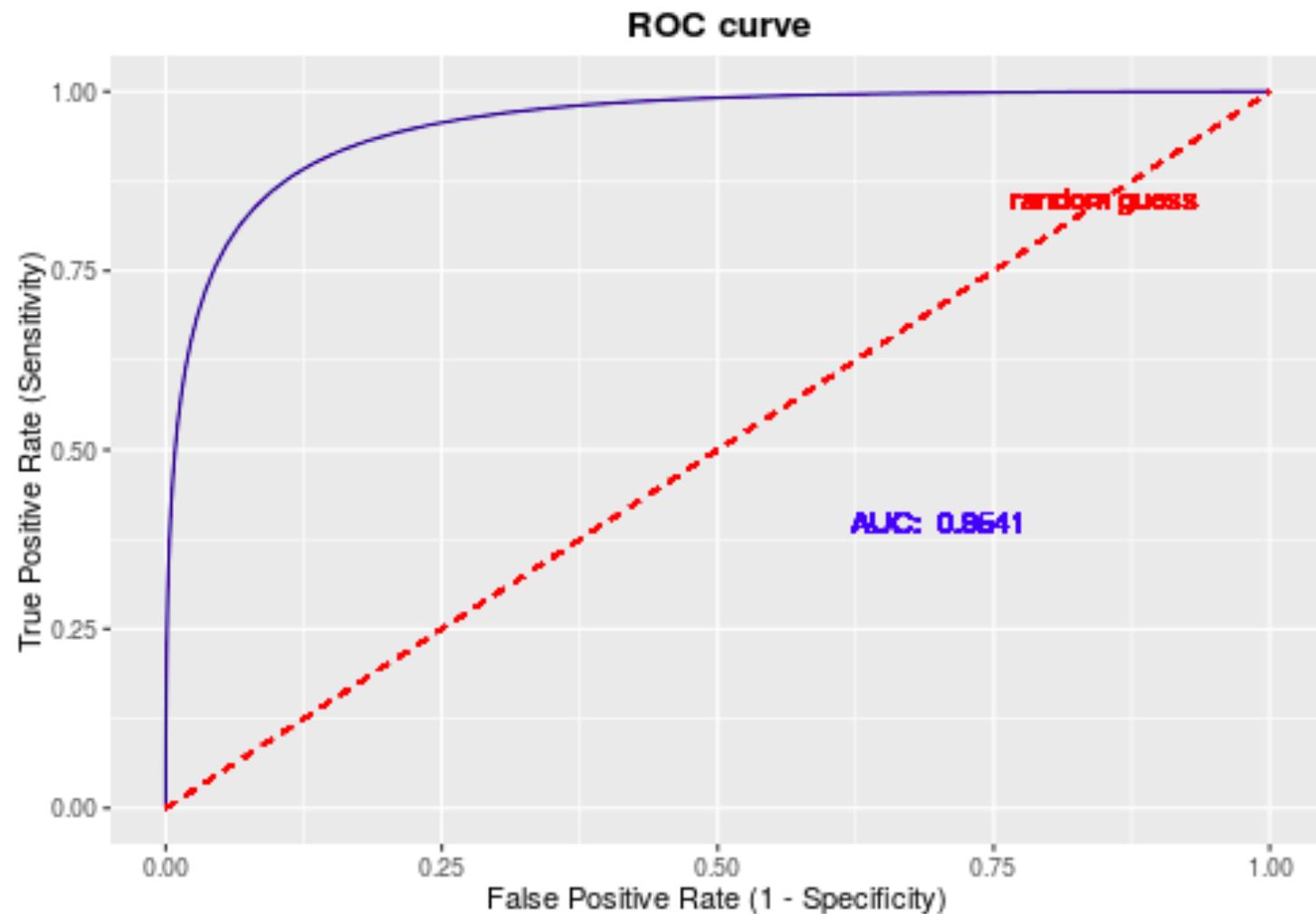
$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

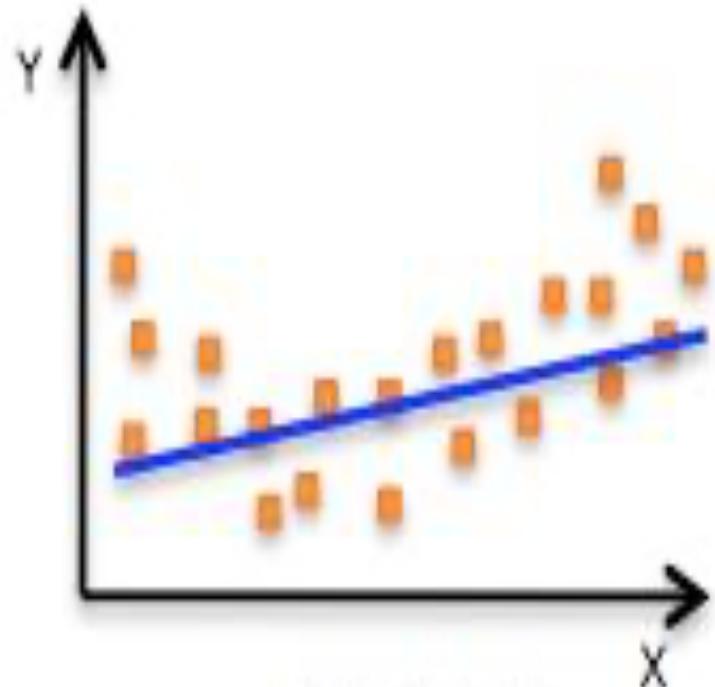
# ROC ?

---

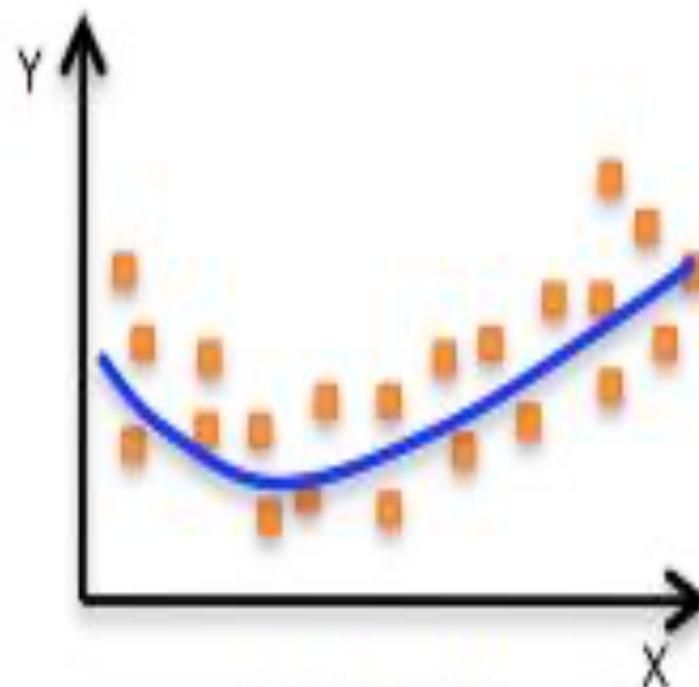


# Underfitting/Overfitting

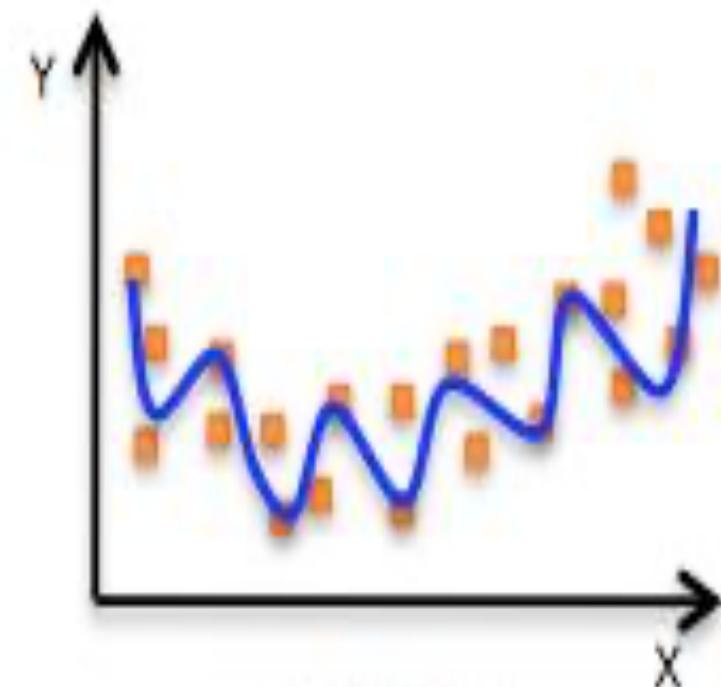
---



Underfitting



Just right!



overfitting

# Thanks

2020-8-15

