



使用scikit-learn构建机器学习模型

2020/5/26

目录

1	机器学习与Scikit-learn
2	贝叶斯分类模型
3	决策树分类模型
4	回归模型
5	聚类

1、机器学习

其本质是借助数学模型理解数据，即建立模型拟合旧观测数据，进而预测和解释新观测数据。机器学习可以弥补统计学习的不足，可以看作人工智能的研究方向。

- **有监督学习**:对数据的特征和标签(类型)之间的关联进行建模，用到新观测数据。
 - (1) 分类。类别标签是离散值 贝叶斯分类、逻辑回归、决策树、KNN
 - (2) 线性回归。类别标签是连续值 线性回归
- **无监督学习**: 对不带任何标签的数据特征进行建模与分类。让数据自己介绍自己，区分个体。
 - (1) 聚类。将数据聚集分成不同的类别
 - (2) 降维。降低数据特征维度，用简洁的方式表现数据



机器如何学习



数据预处理

数据清洗、数据集成、数据采样



特征工程

特征编码、特征选择、特征降维、规范化



数据建模

回归问题、分类问题、聚类问题、其他问题



结果评估

拟合度量、查准率、查全率、F1值、PR曲线、ROC曲线



数据清洗



□ 对各种脏数据进行对应方式的处理，得到标准、干净、连续的数据，提供给数据统计、数据挖掘等使用。

● 数据的完整性

例如人的属性中缺少性别、籍贯、年龄等；
解决方法：信息补全（使用身份证件号码推算性别、籍贯、出生日期、年龄等）；剔除；

● 数据的合法性

例如获取数据与常识不符，年龄大于150岁；
解决方法：**设置字段内容**（日期字段格式为“2010-10-10”）；**类型的合法规则**（性别 in [男、女、未知]）

● 数据的一致性

例如不同来源的不同指标，实际内涵是一样的，或是同一指标内涵不一致；
解决方法：建立数据体系，包含但不限于指标体系、维度、单位、频度等

● 数据的唯一性

例如不同来源的数据出现重复的情况等；
解决方法：按主键去重（用sql或者excel“去除重复记录”） / 按规则去重（如不同渠道来的客户数据，可以通过相同的关键信息进行匹配，合并去重）

● 数据的权威性

例如出现多个来源的数据，且数值不一样；
解决方法：为不同渠道设置权威级别，如：在家里，首先得相信媳妇说的。。。



数据采样



□ 数据不平衡 (imbalance)

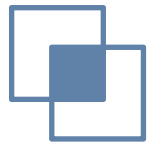
- ✓ **指数数据集的类别分布不均。**比如说一个二分类问题，100个训练样本，比较理想的情况是正类、负类样本的数量相差不多；而如果正类样本有99个、负类样本仅1个，就意味着存在类不平衡。
- ✓ 此时预测时就算全部为正，准确率也可以达到99%，这并**不能反映模型的好坏**。

面临不平衡数据集的时候，传统的机器学习模型的评价方法不能精确地衡量模型的性能

□ 解决方法

- ✓ **过采样** (Over-Sampling)
通过随机复制少数类来增加其中的实例数量，从而可增加样本中少数类的代表性。
- ✓ **欠采样** (Under-Sampling)
通过随机地消除占多数的类的样本来平衡类分布；直到多数类和少数类的实例实现平衡。



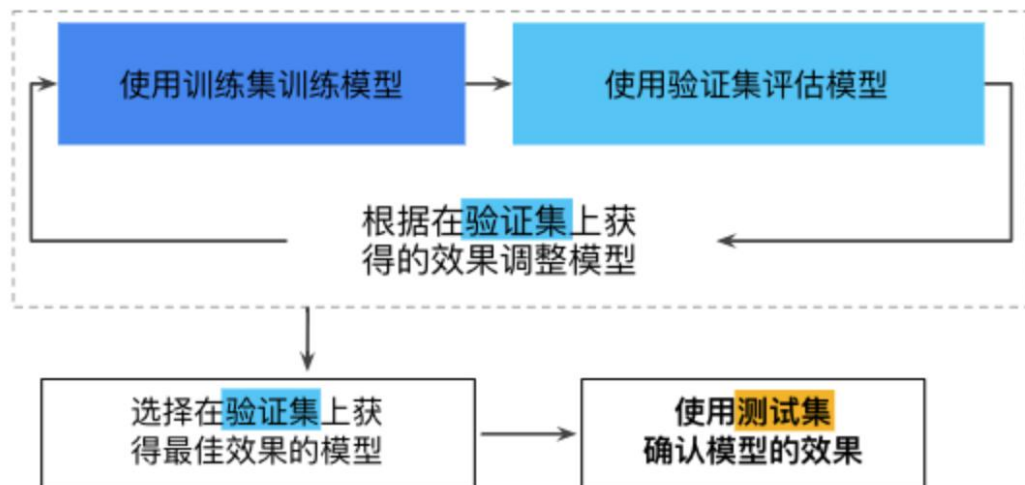


数据集拆分



□ 机器学习中将数据划分为3份

- ① **训练数据集 (train dataset)** : 用来构建机器学习模型
- ② **验证数据集 (validation dataset)** : 辅助构建模型, 用于在构建过程中评估模型, 提供无偏估计, 进而调整模型参数
- ③ **测试数据集 (test dataset)** : 用来评估训练好的最终模型的性能



□ 常用拆分方法

- ✓ **留出法 (Hold-Out)** : 直接将数据集划分为互斥的集合, 如通常选择 70% 数据作为训练集, 30% 作为测试集。需要注意的是保持划分后集合数据分布的一致性, 避免划分过程中引入额外的偏差而对最终结果产生影响。
- ✓ **K-折交叉验证法** : 将数据集划分为 k 个大小相似的互斥子集, 并且尽量保证每个子集数据分布的一致性。这样, 就可以获取 k 组训练 - 测试集, 从而进行 k 次训练和测试, k 通常取值为 10。





机器如何学习



数据预处理

数据清洗、数据采样、数据集拆分

特征工程

特征编码、特征选择、特征降维、规范化

数据建模

回归问题、分类问题、聚类问题、其他问题

结果评估

拟合度量、查准率、查全率、F1值、PR曲线、ROC曲线



特征编码



数据集中经常会出现字符串信息，例如男女、高中低等，这类信息不能直接用于算法计算，需要将这些**数据转化为数值形式**进行编码，便于后期进行建模。

	Direction	District	Elevator	Floor	Garden	Id	Layout	Price	Region	Renovation	Size	Year
0	东西	灯市口	NaN	6	锡拉胡同21号院	101102647043	3室1厅	780.0	东城	精装	75.0	1988
1	南北	东单	无电梯	6	东华门大街	101102650978	2室1厅	705.0	东城	精装	60.0	1988
2	南西	崇文门	有电梯	16	新世界中心	101102672743	3室1厅	1400.0	东城	其他	210.0	1996
3	南	崇文门	NaN	7	兴隆都市馨园	101102577410	1室1厅	420.0	东城	精装	39.0	2004

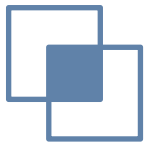
◆ **one-hot编码**：采用**N位状态寄存器**来对N个状态进行编码，每个状态都由他独立的寄存器位，并在任意时候只有一位有效。

- ✓ 图中的Elevator和Renovation都是定类型数据。除去缺失值，Elevator分类有电梯和无电梯两种，因此可用01和10表示。
- ✓ Renovation分为有精装，简装，毛坯和其它四种，可用0001/0010/0100/1000表示。

◆ **语义编码**：one-hot编码无法体现数据间的**语义关系**，对于一些有关联的文本信息来说，无法真正体现出数据关联。

- ✓ 对于这一类信息通常采用词嵌入（word embedding）的方式是比较好的选择，词嵌入信息可以编码语义信息，生成特征语义表示。目前在这一领域比较好的方法是基于google的word2vec方法。





特征选择



★ 这些属性特征都有用吗?

■ 例子 1: 两类问题: 男性、女性

数据特征: 身高、体重、音频、头发长短、
出生日期、家庭住址、籍贯、专业

■ 例子 2: 高维数据的稀疏情况

数据 1: 1, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0
数据 2: 1, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0, 0, 0, 0
数据 3: 1, 0, 1, 0, 0, 0, 5, 3, 0, 2, 0, 0, 0, 0

特征选择



◆ 过滤法(Filter)

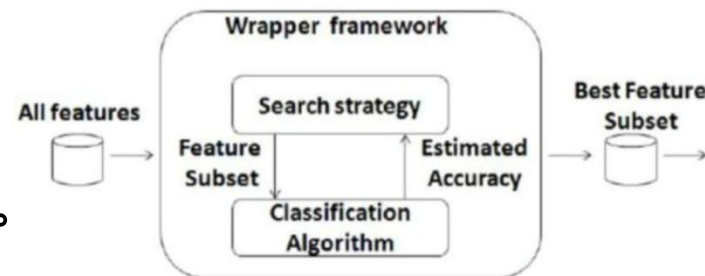
按照发散性或相关性对各特征进行评分, 设定阈值完成特征选择。

✓ 互信息: 指两个随机变量之间的关联程度, 即给定一个随机变量后, 另一个随机变量的确定性; 因而互信息取值最小为0, 意味着给定一个随机变量对确定一另一个随机变量没有关系, 越大表示另一个变量的确定性越高。

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

◆ 包裹法(Wrapper):

选定特定算法, 然后通过不断的启发式方法来搜索特征。



◆ 嵌入法(Embedded):

利用正则化的思想, 将部分特征属性的权重调整到0, 则这个特性相当于就是被舍弃了。常见的正则化有L1的Lasso, L2的Ridge, 还有一种综合L1和L2的这两个方法的Elastic Net方法。

$$\min_{w,b} \mathcal{L} = \frac{1}{n} \sum_{i=1}^n \underbrace{\ell(y_i, f(x_i))}_{\frac{1}{2} (f(x_i) - y_i)^2} + \lambda \Omega(w)$$

$\|w\|_1$
LASSO
Tibshirani, 1996

$\frac{1}{2} \|w\|_2^2$
Ridge Regression
Hoerl & Kennard, 1970

$\rho \|w\|_1 + (1 - \rho) \frac{1}{2} \|w\|_2^2$
Elastic Net
Zou & Hastie, 2005

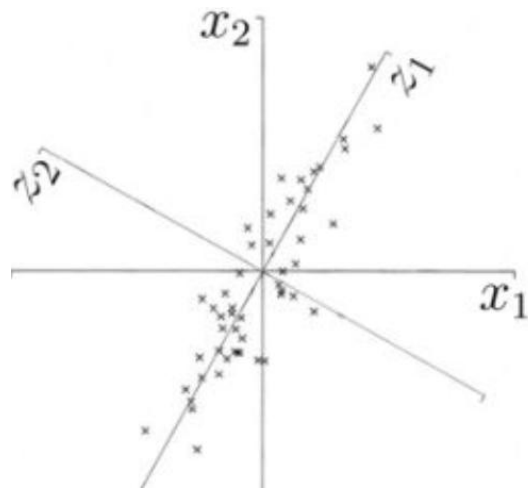
特征降维



特征选择完成后，可能由于特征矩阵过大，导致计算量大、训练时间长，因此**降低特征矩阵维度**也是必不可少的。

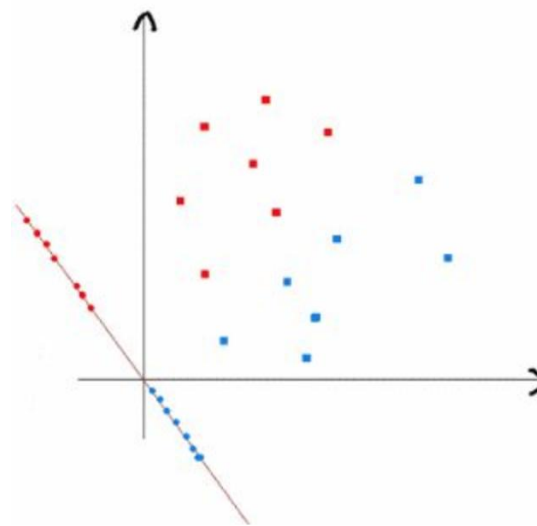
◆ 主成分分析(PCA)

将原始特征空间映射到彼此正交的特征向量空间，在非满秩的情况下使用SVD分解来构建特征向量。



◆ 线性判别分析(LDA)

给出一个标注了类别的数据集，投影到了一条直线之后，能够使得点尽可能的按类别区分开。



规范化

不同属性具有不同量级时会导致：①数量级的差异将导致量级较大的属性占据主导地位；②数量级的差异将导致迭代收敛速度减慢；③依赖于样本距离的算法对于数据的数量级非常敏感。

◆ 标准化

通过减去均值然后除以方差（或标准差），将数据按比例缩放，使之落入一个小的特定区间。

$$x = (x - \mu) /$$

适用于：如果数据的分布本身就服从正态分布，就可以用这个方法。

◆ 区间缩放

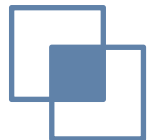
将属性缩放到一个指定的最大和最小值（通常是1-0）之间。

$$x = (x - \min) / (\max - \min)$$

◆ 归一化

将某一属性特征的模长转化成1。

$$x' = \frac{x}{\sqrt{\sum_j^m x[j]^2}}$$



机器如何学习



数据预处理

数据清洗、数据采样、数据集拆分



特征工程

特征编码、特征选择、特征降维、规范化



数据建模

回归问题、分类问题、聚类问题、其他问题



结果评估

拟合度量、查准率、查全率、F1值、PR曲线、ROC曲线





机器学习方法分类



分类问题

决策树

贝叶斯

支持向量机

逻辑回归

集成学习

回归问题

线性回归

岭回归

Lasso回归

聚类问题

K-means

高斯混合聚类

密度聚类

层次聚类

谱聚类

其他问题

隐马尔可夫模型

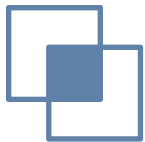
LDA主题模型

条件随机场

神经网络

深度学习





机器如何学习



数据预处理

数据清洗、数据采样、数据集拆分



特征工程

特征编码、特征选择、特征降维、规范化



数据建模

回归问题、分类问题、聚类问题、其他问题、开源框架



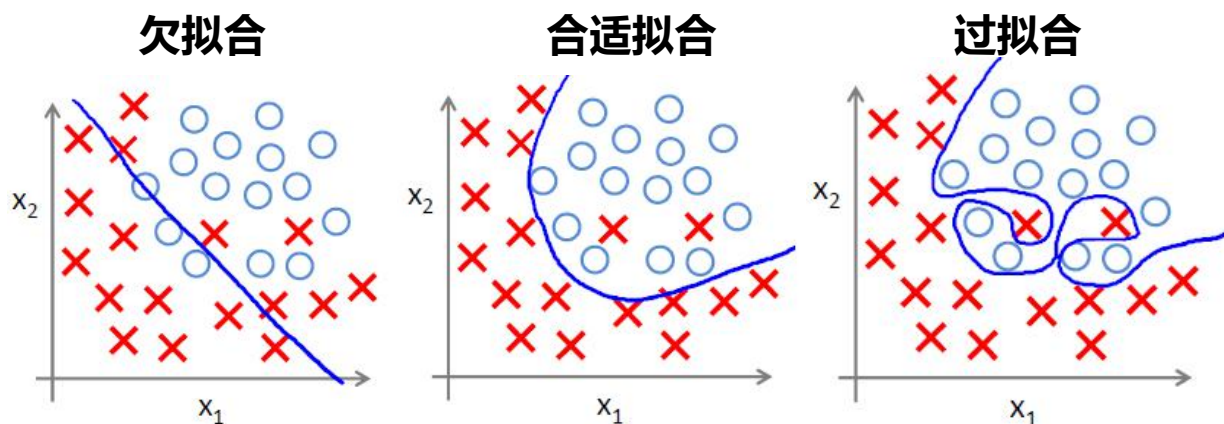
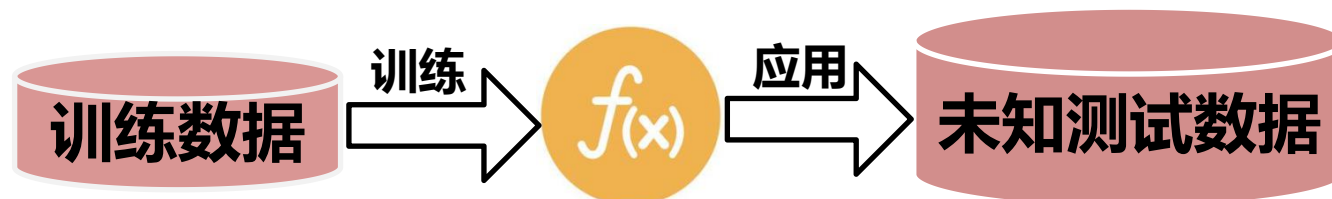
结果评估

拟合度量、查准率、查全率、F1值、PR曲线、ROC曲线



模型选择

泛化误差：在“未来”样本上的误差
经验误差：在训练集上的误差



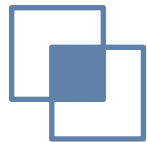
✓ AIC准则 (Akaike Information Criterion)

$$AIC = 2k - 2\ln(L)$$

✓ BIC准则 (Bayesian Information Criterion)

$$BIC = k\ln(n) - 2\ln(L)$$





性能评价指标-分类



准确率(Accuracy)是指在分类中，分类正确的记录个数占总记录个数的比。

$$accuracy = \frac{n_{correct}}{n_{total}}$$

召回率(Recall)也叫查全率，是指在分类中样本中的正例有多少被预测正确了。

通常，准确率高时，召回率偏低；召回率高时，准确率偏低。

1. 地震的预测

对于地震的预测，我们希望的是召回率非常高，也就是说每次地震我们都希望预测出来。这个时候我们可以牺牲准确率。情愿发出1000次警报，把10次地震都预测正确了；也不要预测100次对了8次漏了两次。

2. 嫌疑人定罪

基于不错怪一个好人的原则，对于嫌疑人的定罪我们希望是非常准确的。及时有时候放过了一些罪犯（召回率低），但也是值得的。





性能评价指标-分类

准确率(Accuracy): 分类正确的样本个数占所有样本个数的比例

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

平均准确率(Average per-class accuracy): 每个类别下的准确率的算术平均

$$average_accuracy = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}$$

精确率(Precision 查准率): 分类正确的正样本个数占分类器所有的正样本个数的比例

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

召回率(Recall 查全率): 分类正确的正样本个数占正样本个数的比例

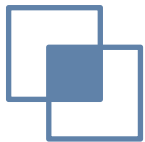
$$Recall = \frac{TP}{TP + FN}$$

分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

F1-Score: 精确率与召回率的调和平均值, 它的值更接近于Precision与Recall中较小的值

$$F1 = \frac{2 * precision * recall}{precision + recall}$$



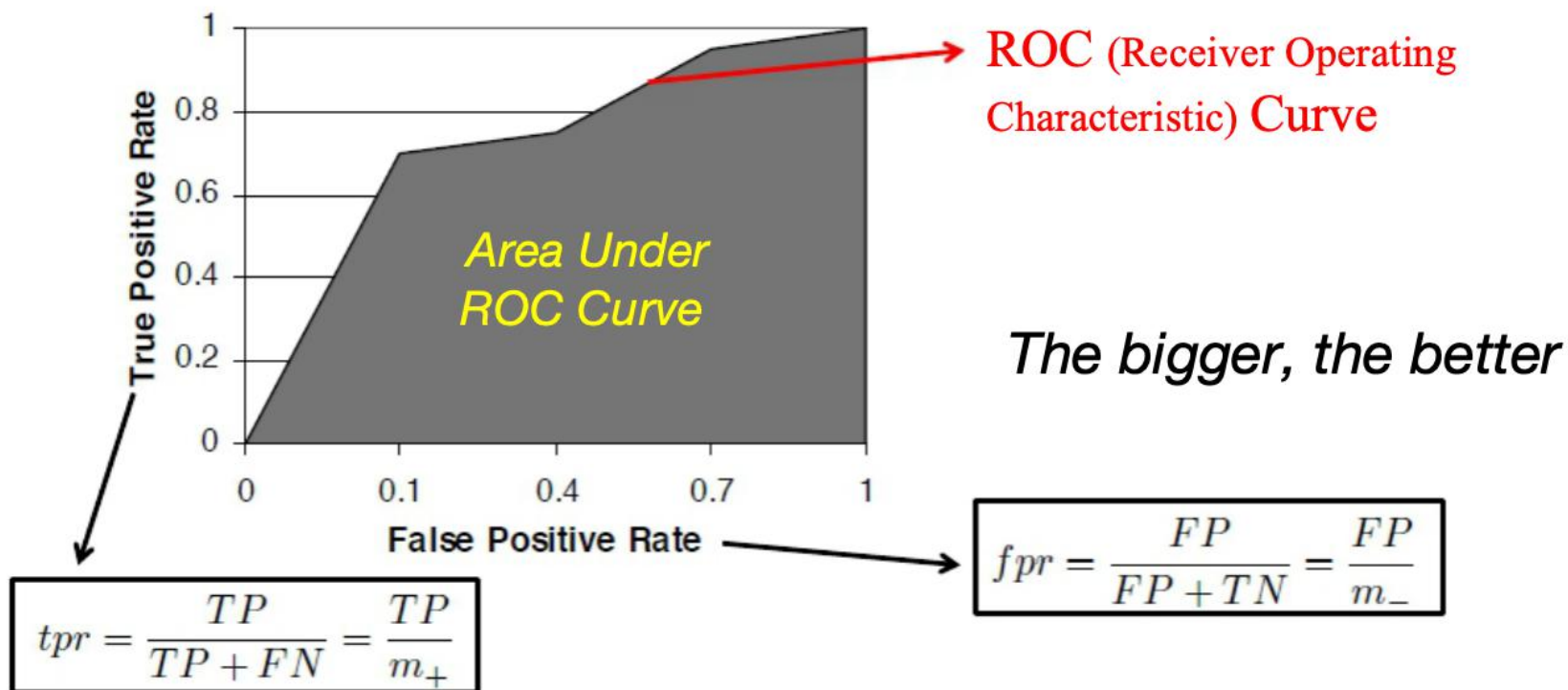
性能评价指标-分类



AUC(Area under the Curve(Receiver Operating Characteristic, ROC))

ROC: 纵轴: 真正例率TPR; 横轴: 假正例率FPR

AUC是ROC曲线下的面积。一般来说, 如果ROC是光滑的, 那么基本可以判断没有太大的overfitting, 这个时候调模型可以只看AUC, 面积越大一般认为模型越好。





性能评价指标-分类



PR曲线：根据学习器的预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测。

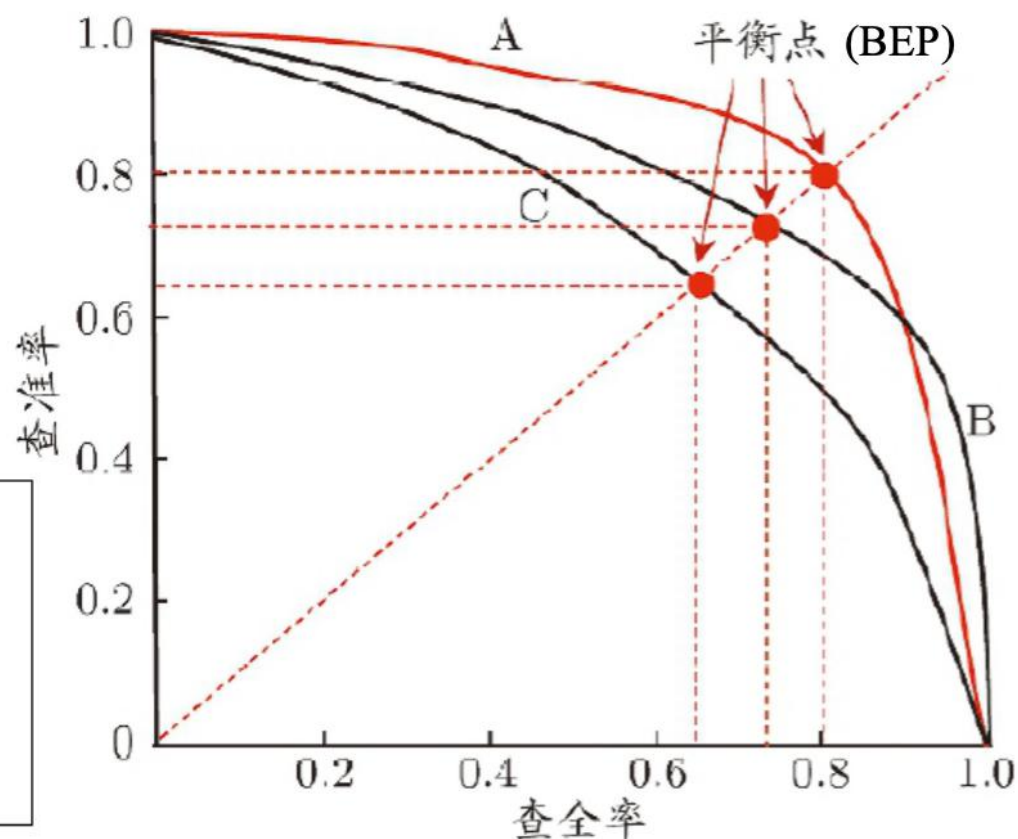
- ✓ 如果一个学习器的**PR曲线**包住了另一个，则可以认为A的性能优于C
- ✓ 如果有交叉，如A、B，综合考虑PR性能，引入**平衡点(BEP)**，基于BEP比较，A优于B

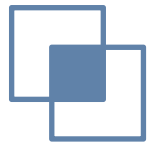
PR图：

- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C
- 学习器 A ?? 学习器 B

BEP：

- 学习器 A 优于 学习器 B
- 学习器 A 优于 学习器 C
- 学习器 B 优于 学习器 C





性能评价指标-分类



宏平均&微平均

多分类问题中，若能得到**多个混淆矩阵**，例如多次训练/测试的结果，多分类的两两混淆矩阵：

宏(macro-)查准率、查全率、F1

$$\text{macro-}P = \frac{1}{n} \sum_{i=1}^n P_i ,$$

$$\text{macro-}R = \frac{1}{n} \sum_{i=1}^n R_i ,$$

$$\text{macro-}F1 = \frac{2 \times \text{macro-}P \times \text{macro-}R}{\text{macro-}P + \text{macro-}R} .$$

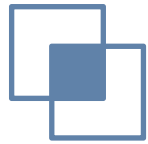
微(micro-)查准率、查全率、F1

$$\text{micro-}P = \frac{\overline{TP}}{\overline{TP} + \overline{FP}} ,$$

$$\text{micro-}R = \frac{\overline{TP}}{\overline{TP} + \overline{FN}} ,$$

$$\text{micro-}F1 = \frac{2 \times \text{micro-}P \times \text{micro-}R}{\text{micro-}P + \text{micro-}R} .$$





性能评价指标-分类



对数损失 (Log loss) 亦被称为逻辑回归损失 (Logistic regression loss) 或交叉熵损失 (Cross-entropy loss) 。

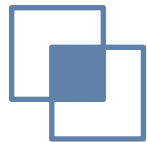
二分类问题: $y \in \{0, 1\}$ 且 $p = \Pr(y = 1)$ 则对每个样本的对数损失为

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

多分类问题: 设Y为指示矩阵, 即当样本i的分类为k, $y_{i,k} = 1$ 设P为估计的概率矩阵, $p_{i,k} = \Pr(t_{i,k} = 1)$ 则对每个样本的对数损失为

$$L_{\log}(Y_i, P_i) = -\log \Pr(Y_i|P_i) = \sum_{k=1}^K y_{i,k} \log p_{i,k}$$





性能评价指标-回归



平均绝对误差：平均绝对误差MAE (Mean Absolute Error) 又被称为l1范数损失 (l1-norm loss)

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} |y_i - \hat{y}_i|$$

平均平方误差：平均平方误差MSE (Mean Squared Error) 又被称为l2范数损失 (l2-norm loss) :

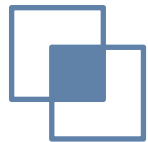
$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=1}^{n_{\text{samples}}} (y_i - \hat{y}_i)^2$$

均方根差RMSE:

R Squared:是将预测值跟只使用均值的情况下相比，看能好多少。

$$R^2 = 1 - \frac{(\sum_i (\hat{y}_i - \bar{y})^2) / m}{(\sum_i (y_i - \bar{y})^2) / m} = 1 - \frac{\text{MSE}(\hat{y}, y)}{\text{Var}(y)}$$





性能评价指标-聚类



外部指标对数据集 $D = \{x_1, x_2, \dots, x_m\}$, 假定通过聚类给出的簇划分为 $C = \{C_1, C_2, \dots, C_k\}$ 参考模型给出的簇划分为 $C^* = \{C_1^*, C_2^*, \dots, C_k^*\}$, 通过比对 C 和 C^* 来判定聚类结果的好坏。

Jaccard系数, FM 指数, Rand 指数, 纯度purity, 熵 entropy, 互信息, Adjusted Rand Index (ARI), F-measure, Probabilistic Rand Index (PRI)

内部指标对聚类数据结构上的描述, 类内距离小, 类间距离大较好。

DB 指数(Davies-Bouldin Index, 简称DBI): 衡量同一簇中数据的紧密性, 越小越好。

Dunn 指数(Dunn Index 简称DI): 衡量同一簇中数据的紧密性, 越大越好。

Silhouette: 衡量同一簇中数据的紧密性, 越大越好。

Modurity: 衡量模块性, 越大越好。



2、Scikit-Learn简介

Python有很多机器学习算法库：Scikit-learn、TensorFlow、Keras、Theano、Caffe、Pylearn2、Orange3、PyBrain

- Scikit-learn是一个基于NumPy, SciPy, Matplotlib的开源机器学习工具包，主要涵盖分类，回归和聚类算法，例如SVM，逻辑回归，朴素贝叶斯，随机森林，k-means等算法。
- Keras（深度学习）Keras是基于Theano的一个深度学习框架，它的设计参考了Torch，用Python语言编写，是一个高度模块化的神经网络库，支持GPU和CPU。
- TensorFlow是谷歌基于DistBelief进行研发的第二代人工智能学习系统，其命名来源于本身的运行原理。Tensor（张量）意味着N维数组，Flow（流）意味着基于数据流图的计算，TensorFlow为张量从流图的一端流动到另一端计算过程。TensorFlow可被用于语音识别或图像识别等多项机器学习和深度学习领

2、Scikit-Learn简介：datasets数据集

datasets模块常用数据集加载函数及其解释

- sklearn库的datasets模块集成了部分数据分析的经典数据集，可以使用这些数据集进行数据预处理，建模等操作，熟悉sklearn的数据处理流程和建模流程。
- datasets模块常用数据集的加载函数与解释如下表所示。
- 使用sklearn进行数据预处理会用到sklearn提供的统一接口——转换器（Transformer）。
- 加载后的数据集可以视为一个字典，几乎所有的sklearn数据集均可以使用data，target，feature_names，DESCR分别获取数据集的数据，标签，特征名称和描述信息。

数据集加载函数	数据集任务类型	数据集加载函数	数据集任务类型
load_boston	回归	load_breast_cancer	分类，聚类
fetch_california_housing	回归	load_iris	分类，聚类
load_digits	分类	load_wine	分类

2、Scikit-Learn简介: datasets数据集

#两种不同的数据集调用方法

```
import seaborn as sns
```

```
irisFd=sns.load_dataset('iris')
```

```
sns.pairplot(irisFd,hue='species',size=1.5)
```

#seaborn建立Pandas数据结构

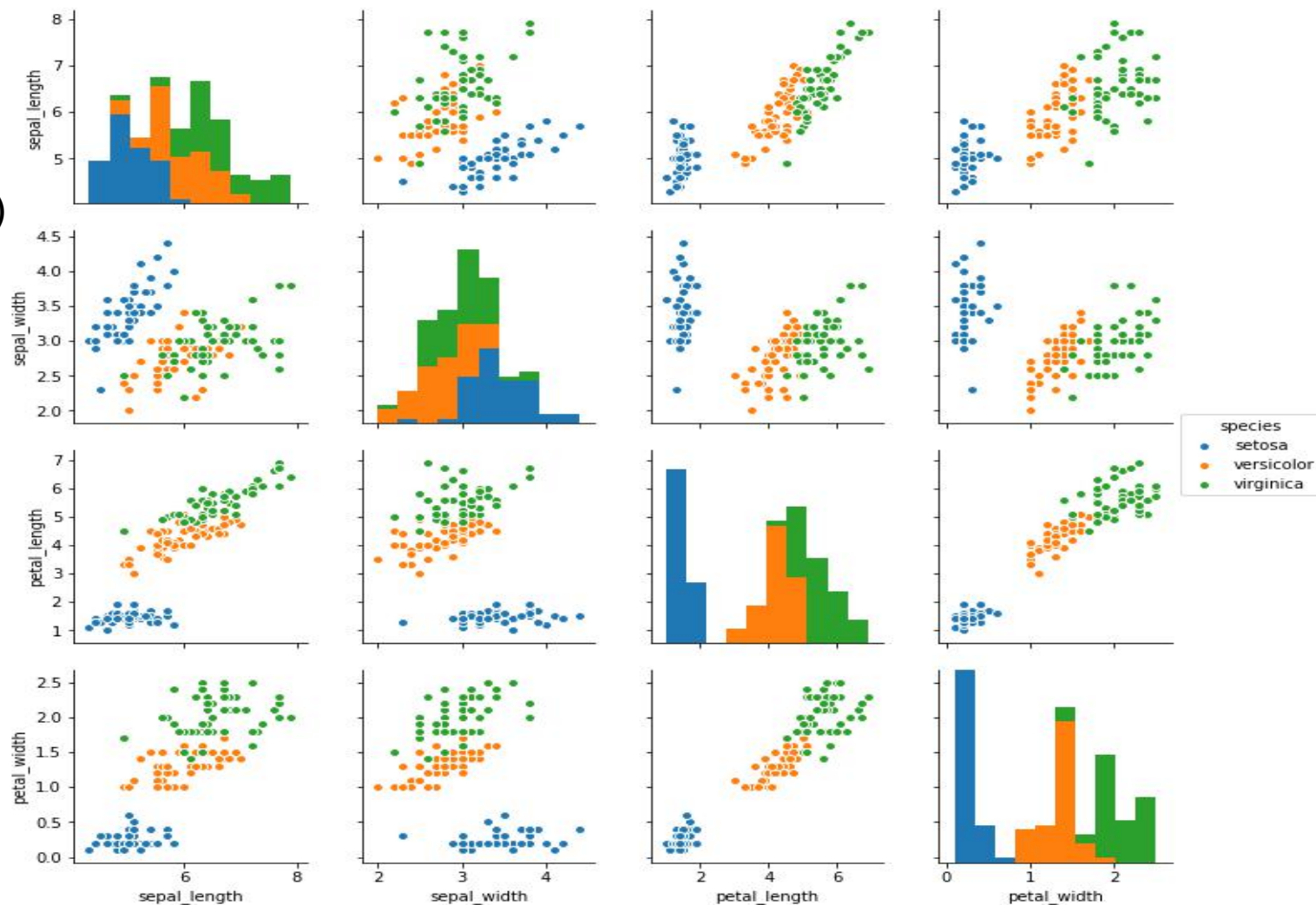
```
from sklearn.datasets import load_iris
```

```
iris = load_iris()
```

```
X = iris.data[:, :2] #获取花卉两列数据集
```

```
Y = iris.target
```

#X Y 为Numpy数据结构



2、Scikit-Learn简介：数据集分割进行训练和测试

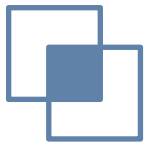
模型构建实例

```
from sklearn.datasets import load_iris    #导入数据集
from sklearn.model_selection import train_test_split #导入自动数据分割函数
from sklearn.naive_bayes import GaussianNB    #导入模型函数
from sklearn.metrics import accuracy_score    #导入正确率评价函数

iris = load_iris()
Xtrain, Xtest, ytrain, ytest = train_test_split(iris.data, iris.target, random_state=1)
model = GaussianNB()                # 2. 初始化贝叶斯分类模型
model.fit(Xtrain, ytrain)            # 3. 训练模型
y_model = model.predict(Xtest)       # 4. 预测
accuracy_score(ytest, y_model)      #0.9736842105263158 模型分类准确率
```

目录

1	机器学习与scikit-learn
2	贝叶斯分类模型
3	决策树分类模型
4	回归模型
5	聚类



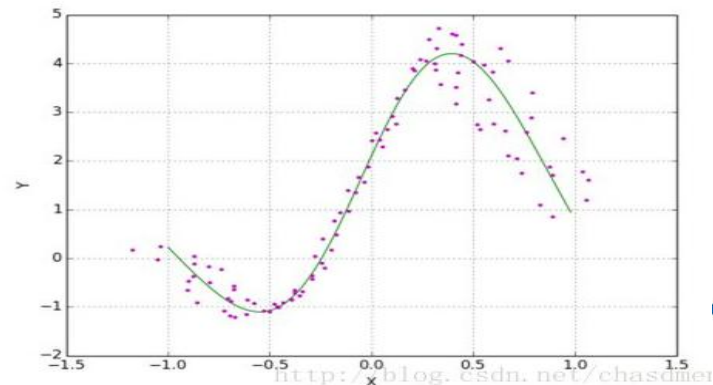
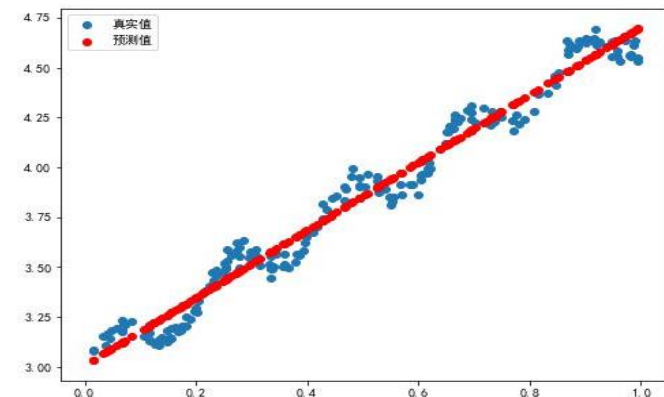
回归问题

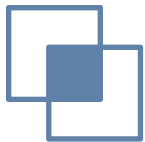


回归分析用于预测输入变量（自变量）和输出变量（因变量）之间的关系，特别是当输入变量的值发生变化时，输出变量值随之发生变化。直观来说回归问题等价于**函数拟合**，选择一条函数曲线使其很好地拟合已知数据且很好地预测未知数据。



为什么叫回归？：达尔文表兄弟Francis Galton发明的。Galton于1877年完成了第一次回归预测，目的是根据上一代豌豆种子（双亲）的尺寸来预测下一代豌豆种子（孩子）的尺寸。他注意到双亲高的，孩子也倾向于比平均高，但尚不及双亲，孩子的高度会向着平均身高回退（回归）。





线性回归



线性回归算法假设特征和结果满足线性关系。这就意味着可以将输入项分别乘以一些常量，再将结果加起来得到输出。

线性回归算法流程

- ① 选择拟合函数形式

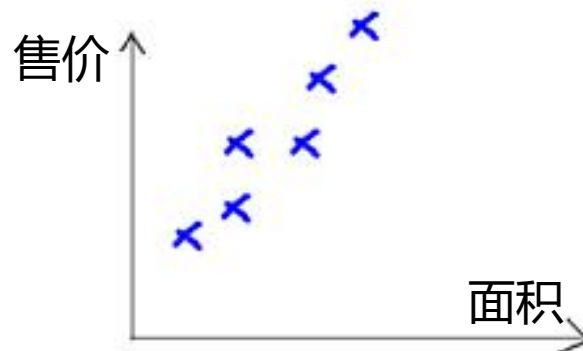
$$h_{\theta}(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$$

- ② 确定损失函数形式

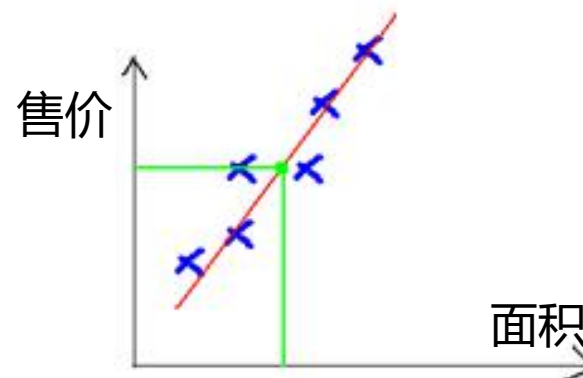
$$\min_{\theta} J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- ③ 训练算法，找到回归系数
如最小二乘、梯度下降等

- ④ 使用算法进行数据预测
 $y = 10 * x + 3$



以面积为X轴，售价为Y轴建立房屋销售数据的特征空间表示图。



用一条曲线去尽量准的拟合这些数据，然后如果有新的输入过来，我们可以在将曲线上这个点对应的值返回。



使用sklearn估计器构建回归模型

sklearn库常用回归算法函数

- sklearn内部提供了不少回归算法，常用的函数如下表所示。
- 可以利用预测结果和真实结果画出折线图作对比，以便更直观看线性回归模型效果。

模块名称	函数名称	算法名称
linear_model	LinearRegression	线性回归
svm	SVR	支持向量回归
neighbors	KNeighborsRegressor	最近邻回归
tree	DecisionTreeRegressor	回归决策树
ensemble	RandomForestRegressor	随机森林回归
ensemble	GradientBoostingRegressor	梯度提升回归树

1、线性回归简介

线性回归：是解决回归任务的起点-----朴素贝叶斯是分类任务的起点。

- 分类：用于预测有限的离散值，如是否得了癌症（0，1），或手写数字的判断，是0,1,2,3,4,5,6,7,8还是9等。

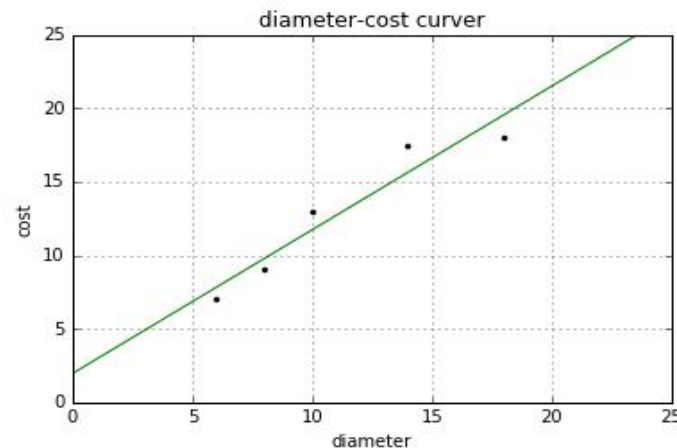
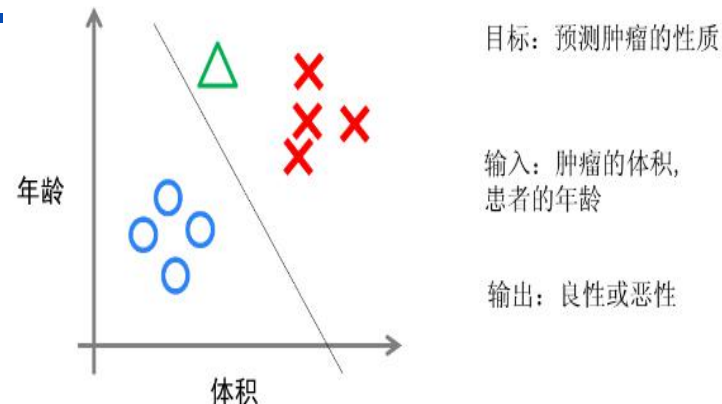
分类中，预测的可能的结果是有限的，且提前给定的。

- 回归：用于预测实数值，如给定了房子的面积，地段，和房间数，预测房子的价格。
- 假定f函数的数学形式已知，其中若干个参数未知，要通过自变量和因变量的观察值去估计未知的参数值
这种情况叫做“线性回归”。

其中应用最广泛的是f为线性函数的假设： $f(x_1, x_2, \dots, x_n) = a_0 + a_1x_1 + a_2x_2 + \dots + a_nx_n$

自变量只有一个时，叫做一元线性回归。 $f(x) = ax + b$

b记为直线截距，a记为斜率。如何计算得到？？？



1、线性回归简介

$y=ax+b$ 参数的计算依据, 利用现有的训练集 (x,y) 来判定未知参数 (a,b) 的值, 使其让 y^{\wedge} 的值更接近实际值 y ?

将横坐标作为 x 轴, 纵坐标作为 y 轴, 每一个点为 $(x^{(i)}, y^{(i)})$, 那么我们期望寻找的直线就是 $y^{\wedge}=ax+b$, 当给出一个新的点 $x^{(j)}$ 的时候, 我们希望预测的 $y^{\wedge(j)}=ax^{(j)}+b$

简单线性回归:

假设我们找到了最佳拟合的直线方程:

$$y = ax + b$$

则对于每一个样本点 $x^{(i)}$

根据我们的直线方程, 预测值为:

$$\hat{y}^{(i)} = ax^{(i)} + b$$

真值为: $y^{(i)}$

我们希望 $y^{(i)}$ 和 $\hat{y}^{(i)}$ 的差距尽量小

表达 $y^{(i)}$ 和 $\hat{y}^{(i)}$ 的差距:

~~$y^{(i)} - \hat{y}^{(i)}$~~ 两数相减有正有负, m 个样本相加有可能为零

~~$|y^{(i)} - \hat{y}^{(i)}|$~~ 绝对值后面计算 a, b 的时候, 我们要求极值, 但是 $y = |x|$ 这种函数不是一个处处可导的连续函数

$(y^{(i)} - \hat{y}^{(i)})^2$ ✓

考虑所有样本: $\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$ ✓

1、线性回归简介

简单线性回归:

目标: 使 $\sum_{i=1}^m (y^{(i)} - \hat{y}^{(i)})^2$ 尽可能小

$$\hat{y}^{(i)} = ax^{(i)} + b$$

目标: 找到a和b, 使得 $\sum_{i=1}^m (y^{(i)} - ax^{(i)} - b)^2$ 尽可能小

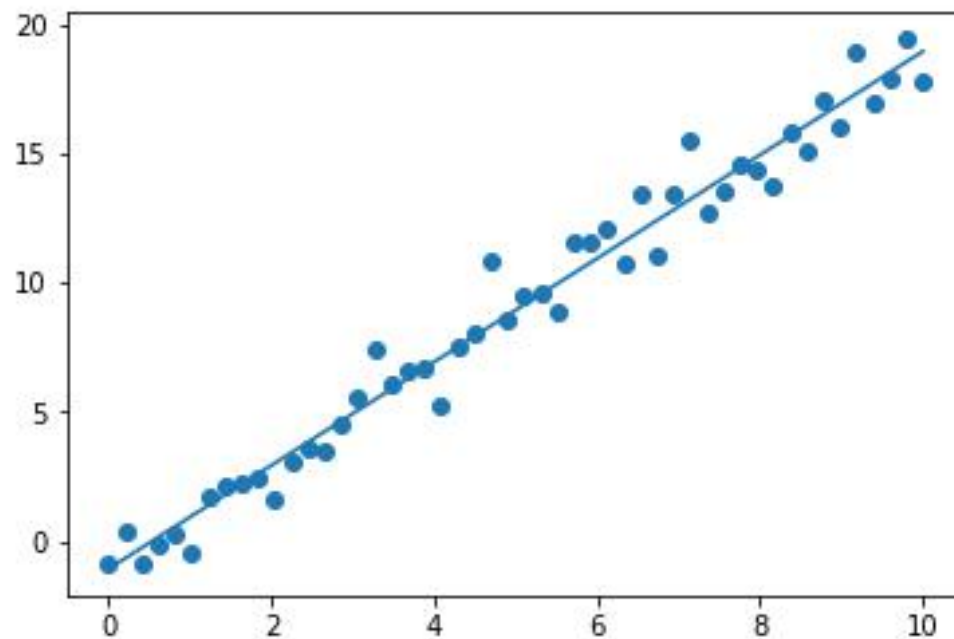
典型的最小二乘法问题: 最小化误差的平方

$$a = \frac{\sum_{i=1}^m (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^m (x^{(i)} - \bar{x})^2} \quad b = \bar{y} - a\bar{x}$$

不用背, 直接可用

1、线性回归简介

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
rng = np.random.RandomState(42)
x = np.linspace(0,10,50)
y = 2 * x - 1 + rng.randn(50) #正态分布均值为0 方差为1的50个点
model = LinearRegression()#fit_intercept=True
model.fit(x.reshape(x.size,1),y) #需要二维数据,
xfit=np.linspace(0,10,1000)
yfit=model.predict(xfit.reshape(xfit.size,1))
plt.scatter(x,y)
plt.plot(xfit,yfit)
plt.show()
print(model.coef_) #斜率 [2.00162915]
print(model.intercept_) #截距-1.0655711700789823
```



#拟合结果与实际很接近。除了简单的直线拟合，还可以处理多维度的线性回归。从几何角度来看，可以拟合三维空间的一个平面，或更高维度的超平面。即为多元线性回归。

1、线性回归简介-一元线性回归

使用线性回归预测**Pizza**的价格，由于直径大小不同的**Pizza**，价格也是不同的。这是一个非常经典的案例，主要包括两个特征——**Pizza**直径（单位：英寸）和**Pizza**价格（单位：美元）假设读者现在去到一家西餐厅，看到**Pizza**的菜单，现在需要通过机器学习的方法构造一个一元线性回归模型，通过分析匹萨直径与价格的数据的线性关系，来预测任意直径匹萨的价格。数据集共十行，包括两个特征，如下表1所示。

```
from sklearn.linear_model import LinearRegression
#数据集 直径、价格
x = [[5],[6],[7],[8],[10],[11],[13],[14],[16],[18]]
y = [6,7.5,8.6,9,12,13.6,15.8,18.5,19.2,20]
clf = LinearRegression()
clf.fit(x,y)
pre = clf.predict([[12]])
print("The price of 12 is {}".format(pre))
```

可以发现直径为12英寸的**Pizza**价格为14.42美元。同时它生成了一个一元线性回归模型，即： $y = a \cdot x + b$ 。其中， y 表示响应变量的预测值，这个示例为**Pizza**的价格预测值； x 为因变量，表示**Pizza**的直径。

表 10.1 Pizza 数据集

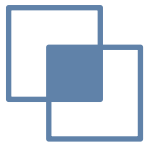
样本序号	直径（英寸）	价格（美元）
1	5	6
2	6	7.5
3	7	8.6
4	8	9
5	10	12
6	11	13.6
7	13	15.8
8	14	18.5
9	16	19.2
10	18	20

1、线性回归简介-多元线性回归

多元回归-披萨价格和多个因素有关

```
from sklearn.linear_model import LinearRegression
X = [[6, 2], [8, 1], [10, 0], [14, 2], [18, 0]]
y = [7, 9, 13, 17.5, 18]
model = LinearRegression()
model.fit(X, y)
X_test = [[8, 2], [9, 0], [11, 2], [16, 2], [12, 0]]
y_test = [11, 8.5, 15, 18, 11]
predictions = model.predict(X_test)
for i, prediction in enumerate(predictions):
    print('Predicted:{}, \tTarget: {}'.format(round(prediction),
y_test[i]))
print(model.score(X_test, y_test))
```

```
In [40]: runfile('C:/Users/HP/Desktop/un
Predicted:10.0,      Target: 11
Predicted:10.0,      Target: 8.5
Predicted:13.0,      Target: 15
Predicted:18.0,      Target: 18
Predicted:13.0,      Target: 11
0.7701677731318468
```





线性回归扩展



线性回归扩展算法用简单的基函数 $\Phi(x)$ 替换输入变量 x 。这样我们就把线性拟合形式扩展到了固定非线性函数的线性组合。

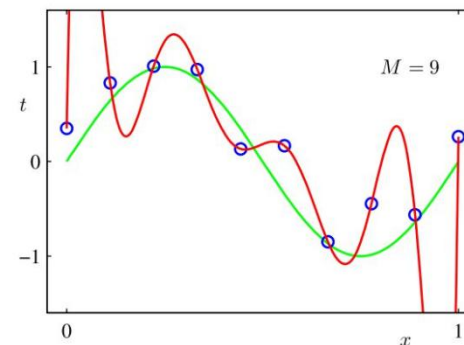
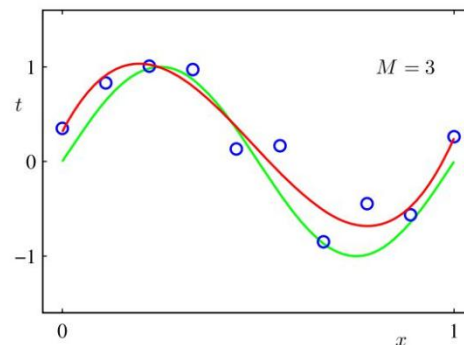
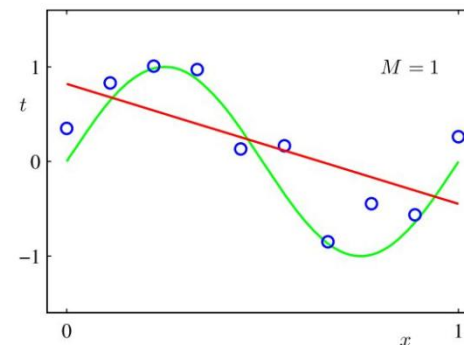
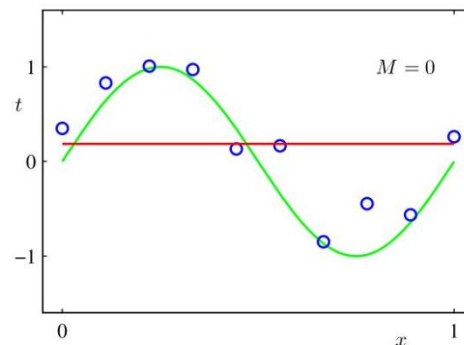
$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1x_1 + \dots + w_Dx_D$$


$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

多项式拟合：取 $\phi_j(x) = x^j$

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$



2、多项式回归

我们假设解释变量和响应变量的关系是线性的。真实情况未必如此。下面我们用多项式回归，一种特殊的多元线性回归方法，增加了指数项。现实世界中的曲线关系都是通过增加多项式实现的，其实现方式和多元线性回归类似。

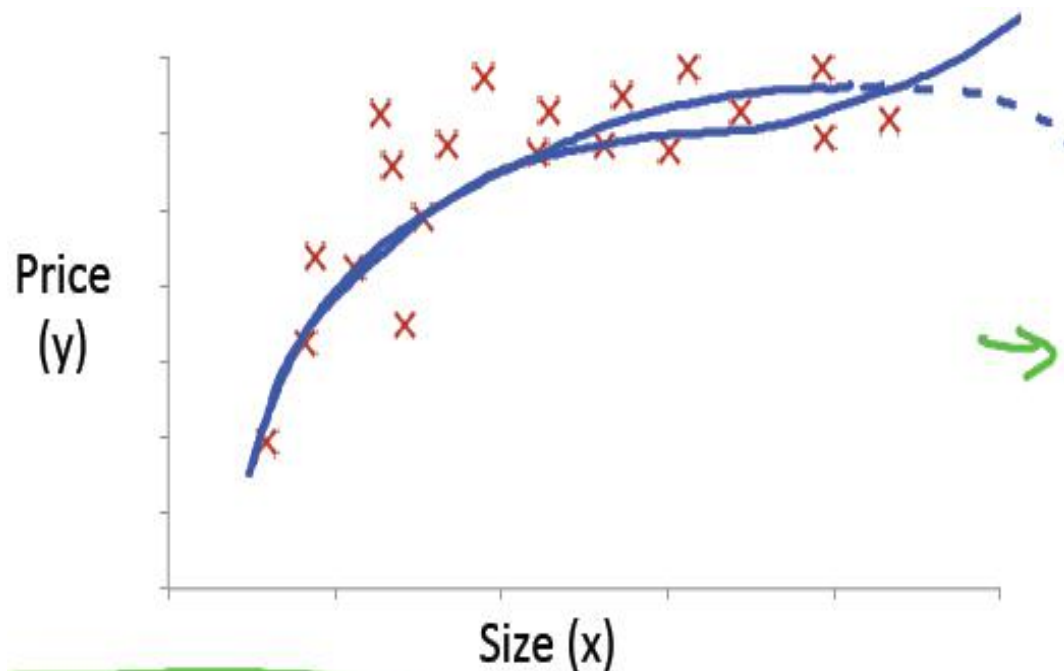
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

$$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

$$\rightarrow x_1 = (\text{size})$$

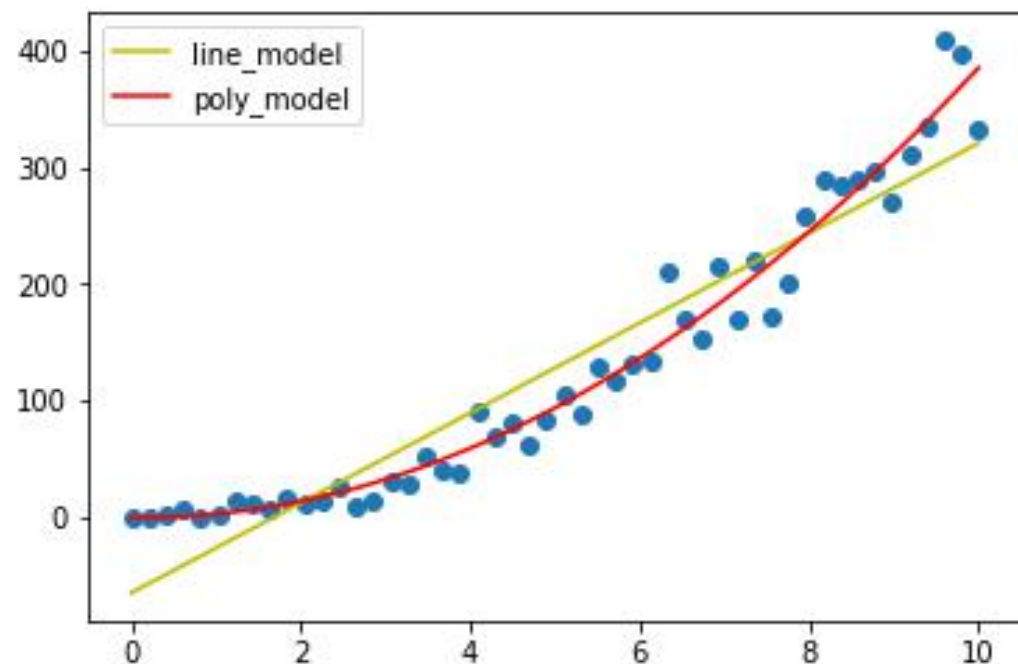
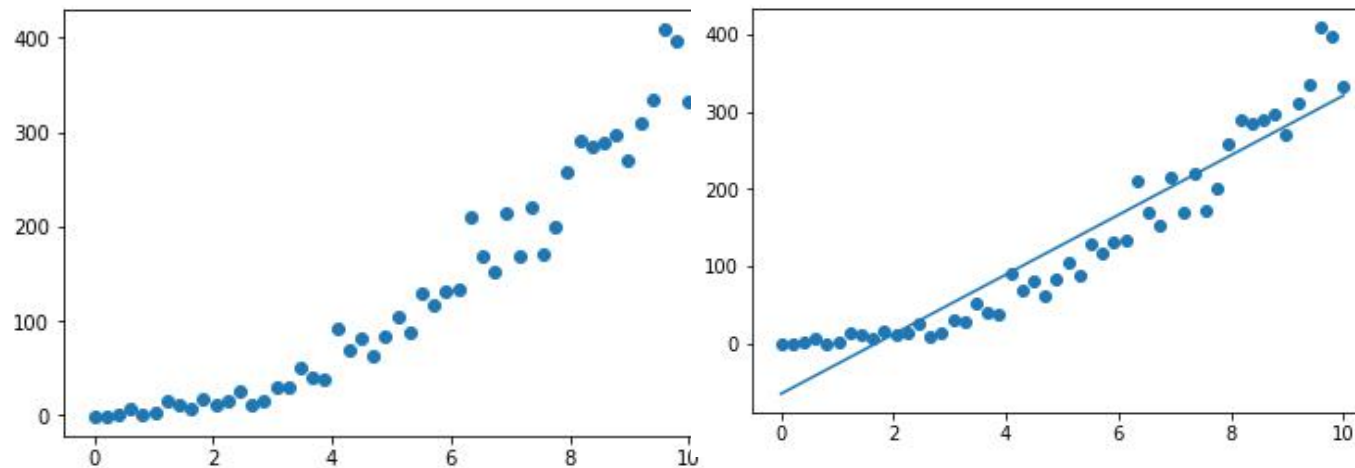
$$\rightarrow x_2 = (\text{size})^2$$

$$\rightarrow x_3 = (\text{size})^3$$



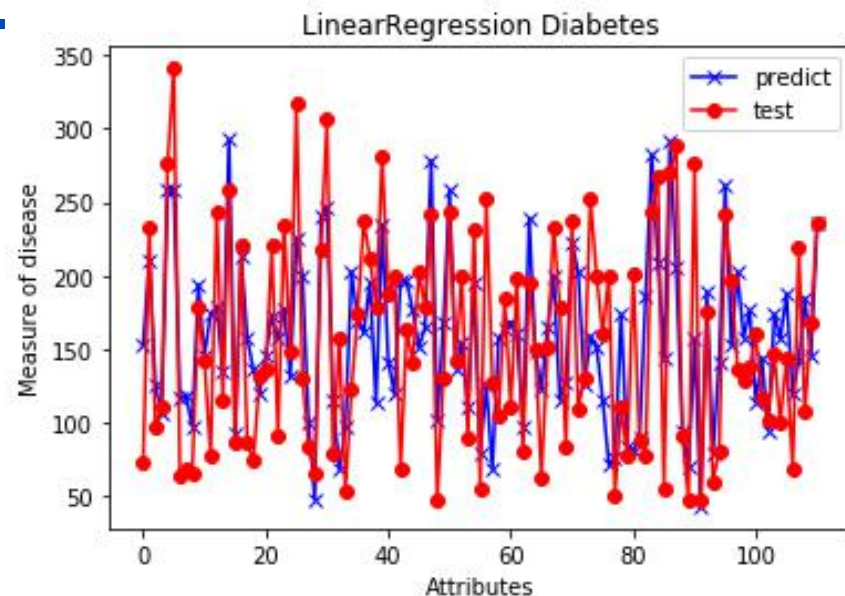
2、多项式回归

```
import matplotlib.pyplot as plt
import numpy as np
rng = np.random.RandomState(42)
x = np.linspace(0,10,50)
y = (2 * x + rng.randn(50))**2-1
x=x.reshape(x.size,1)
from sklearn.linear_model import LinearRegression
model = LinearRegression()model.fit(x,y) #需要二维
xfit=np.linspace(0,10,1000).reshape(1000,1)
yfit=model.predict(xfit)
plt.scatter(x,y)
plt.plot(xfit,yfit,color='y',label='line_model')
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
poly_model = make_pipeline(PolynomialFeatures(2),
                           LinearRegression())
poly_model.fit(x,y) #二次多项式训练
y_pre=poly_model.predict(xfit)
plt.plot(xfit,y_pre,color='r',label='poly_model')
plt.legend()
plt.show()
```



2、多元/多项式线性回归预测糖尿病数据

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score
from sklearn import linear_model
from sklearn import datasets
diabetes = datasets.load_diabetes()    #载入糖尿病数据
X=diabetes.data
Y=diabetes.target
xtrain,xtest,ytrain,ytest=train_test_split(X,Y,random_state=2)
#回归训练及预测
clf = linear_model.LinearRegression()
clf.fit(xtrain,ytrain) #注: 训练数据集
y_pre=clf.predict(xtest)
print('Coefficients :\n',clf.coef_)#斜率  clf.intercept_ #截距
print("variance score: %.2f" % clf.score(xtest, ytest)) #评分函数
plt.plot(range(len(y_pre)),y_pre,color='blue',marker='x',label="predict")
plt.plot(range(len(ytest)),ytest,color='red',marker='o',label="test")
```



Coefficients :

```
[ -36.49214644 -194.10737013  513.87510687
 355.03926144 -890.97772785
  591.66754489  155.4763403   146.44553573
 846.83812282   54.30338383]
```

variance score: 0.44 拟合度很低, 不能满足要求
下面使用多项式回归提升模型效果

2、多元/多项式线性回归预测糖尿病数据

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
poly_model = make_pipeline(PolynomialFeatures(2),
                           LinearRegression())
```

```
diabetes = datasets.load_diabetes()
```

```
Y=diabetes.target
```

```
X=diabetes.data[:,0:7] #取6个特征
```

```
xtrain = X[:-20,:] #训练样本
```

```
xtest = X[-20:,:] #测试样本 后20行
```

```
ytrain = Y[:-20] #训练标记
```

```
ytest = Y[-20:] #预测对比标记 手动标记样本
```

```
poly_model.fit(xtrain, ytrain)#二次多项式训练
```

```
clf = linear_model.LinearRegression()
```

```
clf.fit(xtrain,ytrain) #注: 训练数据集
```

```
y_pre=clf.predict(xtest)
```

```
y_pre_n=poly_model.predict(xtest)
```

```
print(clf.score(xtest, ytest)) #评分函数
```

```
print(poly_model.score(xtest, ytest)) #评分函数
```

```
plt.plot(range(len(y_pre)),y_pre,color='blue',marker='x',label="pre_line")
```

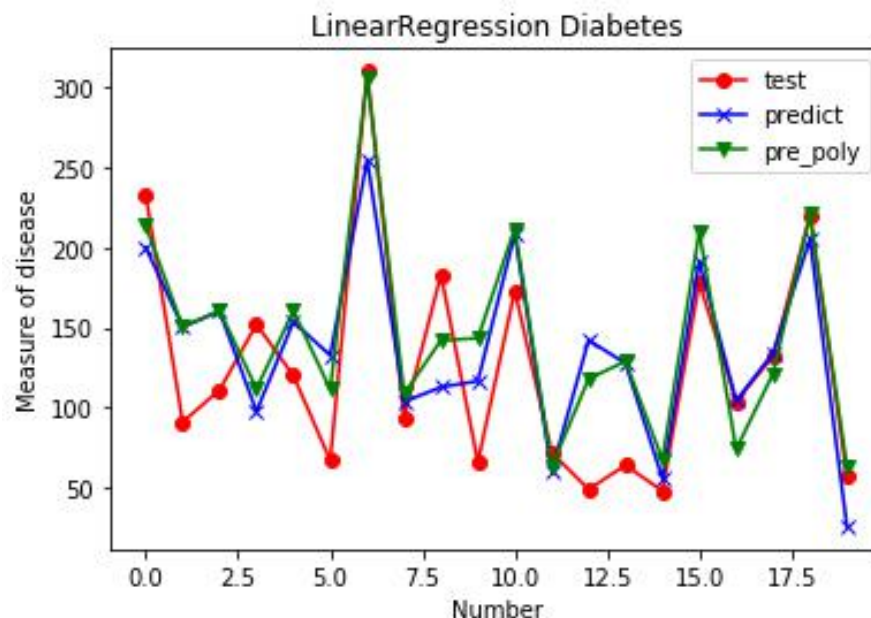
```
plt.plot(range(len(ytest)),ytest,color='red',marker='o',label="test")
```

```
plt.plot(range(len(y_pre_n)),y_pre_n,color='green',marker='v',label="pre_poly")
```

多项式特征维度取6，次数取2

0.5695333747531193 次数1

0.664006082955419 次数2





Thank you!