Python机器学习--期末复习版

0、复习提示

考试分为四种题型：1选择题，2填空题，3简单题，4设计题

选择题和填空题考察基础知识

简答题的难度很不大，都是基本知识和原理的掌握

以上部分大家看我之前给你们的复习提纲就可以完全够用

设计题有两个，需要根据给出的案例进行分析，选择适当算法，并写出算法流程

设计题第一个是根据案例写出算法流程，不需要写代码。第二个是根据案例和给出的代码回答问题，涉及读代码

案例都是熟悉的案例，如果大家每个人都是自己独立认真完成的课程每个实验，设计题相当于白送分

所以大家好好看看做过的每个实验，对设计题会有很大帮助

卷面一共60分，其中选择题10分，填空题10分，简答题28分，设计题12分。实验+三级项目一共40分

1、机器学习的定义

机器学习是人工智能的一个分支。我们使用计算机设计一个系统，使它能够根据提供的训练数据按照一定的方式来学习;随着训练次数的增加，该系统可以在性能上不断学习和改进;通过参数优化的学习模型，能够用于预测相关问题的输出。

2、机器学习的发展历程

- 推理期（20世纪50-70年代初）

认为只要给机器赋予逻辑推理能力，机器就能具有智能
A. Newell 和 H. Simon 的“逻辑理论家”、“通用问题求解”程序，获得了1975年图灵奖

- 知识期（20世纪70年代中期）

认为要使机器具有智能，就必须设法使机器拥有知识
E.A. Feigenbaum 作为“知识工程”之父在 1994 年获得了图灵奖

- 学科形成（20世纪80年代）

20 世纪 80 年代是机器学习成为一个独立学科领域并开始快速发展、各种机器学习技术百花齐放
1980 年美国卡内基梅隆大学举行第一届机器学习研讨会
1990 年《机器学习:风范与方法》出版

- 繁荣期（20世纪80年代-至今）

20 世纪 90 年代后，统计学习方法占主导，代表为SVM
2006 至今，大数据分析的需求，神经网络又被重视，成为深度学习理论的基础

3、监督学习，半监督学习和无监督学习的特点

- **监督学习**（supervised learning）从给定的**有标注的训练数据集**中学习出一个函数（模型参数），当新的数据到来时可以根据这个函数预测结果。 常见任务包括**分类**与**回归。**
- **半监督学习**（Semi-supervised learning）：结合（少量的）标注训练数据和（大量的）未标注数据来进行数据的分类学习。

半监督学习可进一步划分为纯（pure）半监督学习和直推学习（transductive learning），前者假定训练数据中的未标记样本并非待测的数据，而后者则假定学习过程中所考虑的未标记样本恰是待预测数据，学习的目的就是在这些未标记样本上获得最优泛化性能。

- **无监督学习**（unsupervised learning）：**没有标注的训练数据集**，需要根据样本间的统计规律对样本集进行分析，常见任务如**聚类**等。

4、机器学习的步骤，每个步骤的主要内容

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：查准率、查全率、F1值、PR曲线、ROC曲线

5、数据清洗的内容和意义

数据清洗的内容：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，提供给机器学习的后续应用。

数据清洗的意义：保证数据的完整性、唯一性、合法性、权威性、一致性。

6、什么是数据采样

数据采样是指从原始数据集中选取一部分数据进行处理和分析，以得到更加准确和有价值的结果。

数据不平衡：指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。

解决数据不平衡的方法：过采样、欠采样

- 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。

7、什么是特征抽取，特征如何选择，如何编码

特征抽取：在数据集中找到有用的特征属性，通过组合现有特征创建新的特征子集。

特征选择：

- 过滤法（Filter）：其核心在于对特征进行排序——按照特征价值高低排序后，即可实现任意比例/数量的特征选择或剔除。显然，如何评估特征的价值高低从而实现排序是这里的关键环节。
- 包裹法(Wrapper)：包裹式特征选择法的特征选择过程与学习器相关，使用学习器的性能作为特征选择的评价准则，选择最有利于学习器性能的特征子集。常用的包裹式特征选择法有递归特征消除法RFE。
- 嵌入法(Embedded)：嵌入法是一种让算法自己决定使用哪些特征的方法，即特征选择和模型训练同时进行。
  在使用嵌入法时，我们先使用某些机器学习的算法或模型进行训练，得到各个特征的权值系数(0-1之间)。这些权值系数往往代表了特征对模型的贡献或者说重要性。

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。

- 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。
- 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有google的word2vec等。

8、分类算法有哪些?

朴素贝叶斯分类器

逻辑回归

决策树分类算法

随机森林分类算法

支持向量机SVM

K最近邻分类算法

K-Means聚类

9、决策树算法

9.1 决策树定义

决策树是一种描述对样本数据进行分类的树形结构模型，由节点和有向边组成，内部每个节点表示一个属性上的判断，每个分支表示一个判断结果的输出，每个叶节点表示一种分类结果。

9.2 决策树分类

给定包含N个样本的训练数据集$X=(x_1,y_1),(x_2,y_2)\ldots,(x_N,y_N)$，其中$x_i=x_i^1,x_i^2,\ldots,x_i^D$，即每个训练样本包含$D$个特征（如上图中的：色泽、根蒂、敲声等，都是特征）。决策树学习的目标就是根据训练数据集**构建一个决策树模型**，使它能够对实例进行正确的分类。

决策树学习的目的是为了产生一棵泛化能力强，即处理未见示例能力强的决策树。

9.3 决策过程

从根节点开始，测试待分类项中相应的特征属性，并按照其选择输出分值，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

9.4 决策树模型

决策过程中提出的**每个判定问题**都是对某个属性的“测试”

决策过程的**最终结论**对应了我们所希望的判定结果

每个测试的结果或是导出最终结论，或者导出进一步的判定问题，其考虑范围是在上次决策结果的限定范围之内

从根结点到每个叶结点的路径对应了一个判定测试序列

9.5 算法实现

Algorithm 1 决策树学习基本算法

输入:

- 训练集 $D=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_m,y_m)\}$;
- 属性集 $A=\{a_1,\ldots,a_d\}$.

过程: **函数** TreeGenerate(D,A)

生成结点 node;
if D 中样本全属于同一类别 C **then**
　将 node 标记为 C 类叶结点; **return**
end if
if $A=\emptyset$ **OR** D 中样本在 A 上取值相同 **then**
　将 node 标记叶结点，其类别标记为 D 中样本数最多的类; **return**
end if
从 A 中选择最优划分属性 a_*;
for a_* 的每一个值 a_*^v **do**
　为 node 生成每一个分枝; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
　if D_v 为空 **then**
　　将分枝结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
　else
　　以 TreeGenerate$(D_v, A-\{a_*\})$ 为分枝结点
　end if
end for

输出: 以 node 为根结点的一棵决策树

9.6 决策树学习的三个步骤

特征划分选择：信息增益、增益率、基尼系数

决策树生成：ID3、C4.5、CART

决策树剪枝：预剪枝、后剪枝

9.7 特征划分的选择

特征划分选择，即选择最优划分的特征属性。一般而言，随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”(purity)越来越高。

信息熵：是度量样本集合纯度最常用的一种指标，熵越高越混乱

信息增益：信息增益越大，则意味着使用属性 a 来进行划分所获得的“纯度提升”越大

信息增益率：处理选择取值较多的特征的问题，只有不同变量分裂出不同个数字节点的情况下才会起作用。

基尼指数：数据集 D 的纯度可用“基尼值”来度量，基尼值越小，数据集的纯度越高。应选择划分后使基尼指数最小的属性作为最优划分属性

9.8 决策树生成

ID3生成决策树的流程：

1.从根节点开始，对节点计算所有可能的特征的信息增益，选择信息增益最大的特征作为节点的特征，由该特征的不同取值建立子节点。
2.对子节点递归地调用以上方法，构建决策树；直到所有特征的信息增益均很小或没有特征可以选择为止。
3.最后得到一棵决策树。

ID3算法：以**信息增益**为准来选择分支

C4.5算法：以**信息增益率**为基准来选择分支

CART算法：使用**Gini指数**来选择分支的特征变量（二叉树）

9.9 决策树剪枝

为什么剪枝?

防止“过拟合”：避免因决策分支过多，以致于把训练集自身的一些特点当做所有数据都具有的一般性质而导致的过拟合

预剪枝：对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点记为叶结点。（降低过拟合风险、显著减少训练时间和测试时间开销、欠拟合风险）

后剪枝：先从训练集生成一棵完整的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来决策树泛化性能提升，则将该子树替换为叶结点。（欠拟合风险小，泛化性能优秀，训练时间开销大）

9.10 理想的决策树

叶子节点数最少、叶子节点深度最小

9.11 基于决策树算法实现鸢尾花数据集分类

- 数据预处理

函数名称：sklearn.model_selection.train_test_split(data, labels, test_size,random_state)

函数参数：data: 数据；labels:标签；test_size: 测试集比例；random_state：随机数种子；

函数返回值：返回划分完成的训练集和测试集的数据和标签

```
#导入鸢尾花数据集
data=datasets.load_iris()
feature=data.data
label=data.target

#查看数据集的规模
print('共包括',label.shape[0],'个样本，特征维度为',feature.shape[1])

#划分训练集和测试集：30%数据为测试集，其余70%为训练集
x_train,x_test,y_train,y_test=train_test_split(feature,label,test_size=0.3,random_state=33)
```

- 模型配置和训练

函数名称：sklearn.tree.DecisionTreeClassifier.fit(train_data, train_label)

函数功能：将训练数据传入模型中，并训练模型

函数参数：train_data: 训练集样本特征；train_labels:训练集样本标签；

函数返回值：返回该对象实例DecisionTreeClassifier

```
# 模型配置
model = tree.DecisionTreeClassifier() # 决策树分类器

# 模型训练
clf = model.fit(x_train, y_train)
```

- 可视化决策树模型

函数名称：sklearn.tree.export_text(model, feature_names)

函数功能：生成展示决策树规则的文本日志

函数参数：model: 决策树分类模型；feature_names:训练集样本特征名称(例如：萼片宽度等)；

函数返回值：决策树规则的文本描述

```
# 文本导出决策树的规则
r = export_text(clf, feature_names=data['feature_names'])
print(r)
```

- 模型预测

函数名称：sklearn.tree.DecisionTreeClassifier.predict(test_data)

函数功能：对测试数据进行预测

函数参数：test_data: 测试集样本特征；

函数返回值：返回预测结果，即预测的样本类别

```
28    # 模型预测
29    y_predict = clf.predict(x_test)
30    print(y_predict)
```

9.12 实验项目：肿瘤预测、顾客购买服装

```
# ID3 算法：
# 进行数据集分割。
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,
test_size=0.2, random_state=42)

# 配置决策树模型。
clf = DecisionTreeClassifier()

# 训练决策树模型。
clf.fit(X_train, y_train)

# 模型预测。
predictions = clf.predict(X_test)

# 计算ID3算法模型的精确度、召回率和F1分数
id3_precision = precision_score(y_test, id3_predictions)
id3_recall = recall_score(y_test, id3_predictions)
id3_f1 = f1_score(y_test, id3_predictions)

print("ID3算法的精确度：", id3_precision)
print("ID3算法的召回率：", id3_recall)
print("ID3算法的F1分数：", id3_f1)
```

```
# CART 算法：
# 配置CART算法的决策树模型
cart_model = DecisionTreeClassifier(criterion='gini')

# 训练CART算法的决策树模型
cart_model.fit(X_train, y_train)

# 计算CART算法模型的精确度、召回率和F1分数
cart_precision = precision_score(y_test, cart_predictions)
cart_recall = recall_score(y_test, cart_predictions)
cart_f1 = f1_score(y_test, cart_predictions)

print("CART算法的精确度：", cart_precision)
print("CART算法的召回率：", cart_recall)
print("CART算法的F1分数：", cart_f1)
```

10、贝叶斯分类算法

10.1 贝叶斯算法基本思路

对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。朴素贝叶斯分类器认为事件属性在概率分布上是独立的。

$$P(B \mid A) = \frac{P(A|B)P(B)}{P(A)}$$

10.2 算法过程

假设一个待分类样本$x = x_1, x_2, x_3, x_4 \ldots x_D$，$x_i$是$x$的属性，整个样本集$X$可以被分成$n$类，类别$y = y_1, y_2, y_3 \ldots y_n$，则预测$x$的所属类别为$y_k = argmax(P(y_k|x))(y_k \in y)$，根据贝叶斯定理可得：

$$P(y_k \mid x) = \frac{P(x|y_k)P(y_k)}{P(x)}$$

即：

$$P(\text{类别} \mid \text{特征}) = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})}$$

贝叶斯公式

$$P(c \mid \boldsymbol{x}) = \frac{P(c)P(\boldsymbol{x}|c)}{P(\boldsymbol{x})} = \frac{P(c)}{P(\boldsymbol{x})}\prod_{i=1}^{d} P(x_i \mid c)$$

$$P(\text{类别}|\text{特征}) = \frac{P(\text{特征}|\text{类别})P(\text{类别})}{P(\text{特征})}$$

先估算先验概率$P(c)$，为每个属性计算条件概率$P(x_i|c)$，计算后验概率$P(c|x)$

10.3 基于朴素贝叶斯实现印第安人数据集分类

- 数据预处理：读取数据

函数名称：pandas.read_csv(filepatah, names)

函数参数：filepath: csv文件路径；names: 列名数组；

函数返回值：返回DataFrame对象

```
#数据准备阶段
filename = 'work/Pima.csv'
#怀孕次数、葡萄糖、血压、皮层厚度、胰岛素、BMI体重指数、糖尿病谱系功能、年龄、类标变量 （0或1）
names = ['preg','plas','pres','skin','test','mass','pedi','age','class']

#数据导入
dataset = read_csv(filename,names = names)
dataset.head() #查看前5行数据
```

- 数据预处理：特征选择、数据集划分

函数名称：sklearn.model_selection.train_test_split(data, labels, test_size)

函数参数：data: 数据；labels:标签；test_size: 测试集比例；

函数返回值：返回划分完成的训练集和测试集的数据和标签

```
#数据的特征选择
array = dataset.values
x = array[:,0:8] #特征数据
y = array[:,8]   #目标数据

#数据划分：20%数据为测试集，其余80%为训练集
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

- 模型配置和训练

函数名称：sklearn.naive_bayes.GaussianNB.fit(train_data, train_label)

函数功能：将训练数据传入模型中，并训练模型

函数参数：train_data: 训练集样本特征；train_labels:训练集样本标签；

函数返回值：返回该对象实例GaussianNB

```
#模型配置
mode1 = GaussianNB()     #高斯朴素贝叶斯
mode2 = MultinomialNB() #多项式朴素贝叶斯
mode3 = BernoulliNB()    #伯努利朴素贝叶斯
```

- 模型评估

函数名称：sklearn.model_selection.cross_val_score(model, data, labels, cv, scoring)

函数功能：根据传入的数据和标签，对模型进行交叉验证

函数参数：model: 要评估的模型；data: 用于评估的样本特征；labels:用于评估的样本标签；

cv: 将模型拆分为cv组进行交叉验证；scoring: 可调用的评分函数, 例如准确率'accuracy'

函数返回值：评分结果

```
#模型评估
#cross_val_score交叉验证
sorcel = cross_val_score(mode1,x_train,y_train,cv=10,scoring='accuracy') #计算高斯朴素贝叶斯算法模型的准确率
sorce2 = cross_val_score(mode2,x_train,y_train,cv=10,scoring='accuracy') #计算多项式朴素贝叶斯算法模型的准确率
sorce3 = cross_val_score(mode3,x_train,y_train,cv=10,scoring='accuracy') #计算伯努利朴素贝叶斯算法模型的准确率

print("高斯朴素贝叶斯模型的准确率：",sorcel.mean())
print("多项式朴素贝叶斯模型的准确率：",sorce2.mean())
print("伯努利朴素贝叶斯模型的准确率：",sorce3.mean())
```

- 模型预测

函数名称：sklearn.naive_bayes.GaussianNB.predict(test_data)

函数功能：对测试数据进行预测

函数参数：test_data: 测试集样本特征；

函数返回值：返回预测结果，即预测的样本类别

```
#模型预测
y_predict1 = mode1.predict(x_test)
y_predict2 = mode2.predict(x_test)
y_predict3 = mode3.predict(x_test)
```

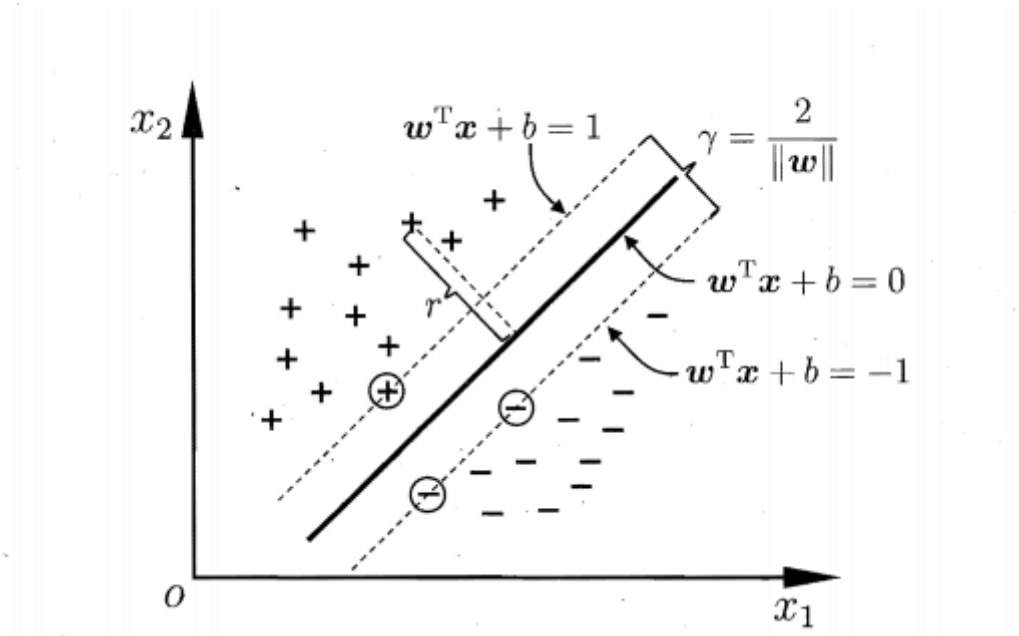
10.4 鸢尾花分类

```
classifier=GaussianNB()                 #搭建模型
classifier.fit(x_train,y_train)         #训练模型
prediction=classifier.predict(x_test)   #模型预测
```

11、支持向量机（SVM）

11.1 支持向量机概念

支持向量机 (Support Vector Machine ,SVM) 是一种有监督学习方法，就是寻找具有最大边缘（间隔）的超平面，位于间隔里面的平行超平面，都能实现对数据点的线性可分。同时，面对近似线性分的情况，需要适当放宽这个间隔，引入**软间隔和松弛因子**。而面对更复杂的低维线性不可分的情况，通过使用**核函数**将数据点映射到高维，进行寻找超平面进行划分。



11.2 线性可分SVM（硬间隔）

应选择"**正中间**"，容忍性好，鲁棒性高，泛化能力最强。

超平面方程：$f(x)=\omega^{T}x+b=0$

最大间隔：寻找参数 ω 和 b，使得 γ 最大。

线性可分：构造拉格朗日函数、序列最小最优化算法-SMO

11.3 线性近似可分SVM（软间隔）

由于噪声存在或数据本身分布存在偏差，现实中大多数情况下很难实现二类问题的完美划分，需要引入软间隔与松弛变量、Hinge Loss、惩罚因子。

11.4 线性不可分SVM

对于一维线性不可分，映射到二维空间就可以获得一条直线，仍不可分变换坐标空间，二维不可分则映射到三维空间找平面

将数据点从原始空间映射到特征空间

核函数映射：映射后的特征空间数据点内积的计算等价于低维空间数据点在映射函数对应的和函数中的计算

11.5 鸢尾花SVM分类

C越大，相当于惩罚松弛变量，训练集测试准确率高，但泛化能力弱；C越小，对误分类惩罚减小，允许容错，泛化能力较强

gamma值越大，造成过拟合，导致测试集拟合变差；gamma值越小，泛化能力越好，但会出现欠拟合

Kernel='rbf'高斯核函数，'linear'——线性核函数 'poly'——多项式核函数 'rbf'——径像核函数/高斯核函数 'sigmod'——多层感知机核函数 'precomputed'——核矩阵。

```
classifier=SVC(kernel='linear',C=10)       #配置模型，线性核函数、惩罚系数为10的SVM模型
classifier.fit(x_train,y_train)            #训练模型
prediction=classifier.predict(x_test)      #模型预测
#查看分类效果
print('分类精度为：',classifier.score(x_test,y_test)*100,'%')  #模型评估
print('评价指标：\n',classification_report(y_test,prediction))

#其他评估方法：采用准确率(accuracy)作为评估函数：预测结果正确的数量占样本总数，
(TP+TN)/(TP+TN+FP+FN)
from sklearn.metrics import accuracy_score # 导入准确率评价指标
print('Accuracy:%s' % accuracy_score(y_test, prediction))
```

12、逻辑回归

13、线性回归、最小二乘法、梯度下降法

13.1 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，再与偏置项相加得到输出。**一元线性回归**指的是分析只有一个自变量x与因变量y线性相关关系的方法。

线性回归的本质就是一个全连接层（fully-connected layer）

算法流程：

一元线性回归算法流程

① 选择拟合函数形式

$$h(x) = \theta x$$

② 确定损失函数形式

$$\min_{\theta} J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - h(x_i))^2$$

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

损失函数：

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）

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}|$$

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）：

$$MSE(y,\hat{y}) = \frac{1}{n}\sum_{1}^{n}(y_i - \hat{y})^2$$

均方根差RMSE：是MSE的算术平方根

13.2 最小二乘法

通过最小化误差的平方和，使得拟合对象无限接近目标对象。

$$\min_{\theta} J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - h(x_i))^2$$

主要思想：求解未知参数，是的理论值和观测值之差（残差）的平方和最小。

缺点：最小二乘法主要针对于线性函数，有全局最优解且是闭式解，针对更加复杂的函数难起作用

13.3 梯度下降法

可看成是更简单的一种最小二乘法最后一步解方程的方法，梯度下降法是用来计算函数最小值的。

$$\theta_0 := \theta_0 - \alpha\frac{\partial J(\theta_0)}{\partial \theta_0}$$

根据计算一次目标函数梯度的样本数量可分为批量梯度下降（Batch graduebt descent, BGD），随机梯度下降（Stochatic gradient decent， SGD），小批量梯度下降(mini-batch gradient descent)。

BGD：所有样本都用上，一次求一个梯度。优点：能够收敛，不需要逐渐减小学习速率。缺点：由于每一步都要使用所有数据,因此随着数据集的增大,运行速度会越来越慢.

SGD：一次只抽取一个随机样本进行目标函数梯度计算。优点：收敛非常快。缺点：因为是随机抽取样本，因此误差是不可避免的，且每次迭代的梯度受抽样的影响比较大。

梯度下降法的挑战：

难以选择合适的学习速率：如果降低学习率，其周期是人为事先设定的，所以它不能很好地适应数据内在的规律；我们对特征向量中的所有的特征都采用了相同的学习率，如果训练数据十分稀疏并且不同特征的变化频率差别很大，这时候对变化频率慢得特征采用大的学习率而对变化频率快的特征采用小的学习率是更好的选择；梯度下降方法难以逃脱"鞍点"

梯度下降法的改进：

Momentum：若当前的梯度方向与累积的历史梯度方向一致，则当前的梯度会被加强，反之减弱。

AdaGrad：每一次更新参数时（一次迭代），不同的参数使用不同的学习率。

Adam：动态调整每个参数的学习率。其优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳。

13.4 线性回归的改进

过拟合是指模型学习的参数过多，导致拟合的函数完美的预测训练集，但对新数据的测试集预测结果差。

添加正则项：

根据所加正则项的不同，可分为L1正则化（ LASSO回归）与L2正则化（岭回归）。

L1正则项：$\min_{\theta} J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - h(x_i))^2 + \lambda||\theta||_1$

$$||\theta||_1 = \sum_{i=1}^{m}|\theta_i|$$

L2正则项：

$$\min_{\theta} J(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(y_i - h(x_i))^2 + \lambda||\theta||_2^2$$

$$||\theta||_2 = \sqrt{\sum_{i=1}^{m}\theta_i^2}$$

13.5 波士顿房价预测

```
# 数据集划分
# 数据集划分
from sklearn.model_selection import train_test_split  #导入数据划分包
# 把X、y转化为数组形式，以便于计算
X = np.array(X.values)
y = np.array(y.values)
# 以25%的数据构建测试样本，剩余作为训练样本
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
# 模型训练
from sklearn.linear_model import LinearRegression  #使用LinearRegression库
lr=LinearRegression()   #设定回归算法
lr.fit(X_train,y_train) #使用训练数据进行参数求解
```

```
# 模型预测
y_hat = lr.predict(X_test) #对测试集的预测
y_hat[0:9]  #打印前10个预测值
```

14、K-means算法

14.1 聚类问题

无监督学习是从无标注的数据中学习数据的统计规律或者内在结构的机器学习方法，主要包括：聚类、降维等。

聚类是将给定的样本，依据它们特征的**相似度**或**距离**，将样本集合中相似的样本分配到相同的类，不相似的样本分配到不同的类。

同一簇的样本尽可能彼此相似，不同簇的样本尽可能不同。换言之，聚类结果的“簇内相似度”高，且“簇间相似度”低，这样的聚类效果较好。

聚类性能度量：

外部指标：将聚类结果与某个“参考模型” 进行比较。 (Jaccard系数、FM指数、Rand指数在[0,1]区间内越大越好)

内部指标：直接考察聚类结果而不使用任何参考模型。

14.2 K-Means聚类

K-Means聚类(K均值聚类)：随机选取k个点作为初始的聚类中心点，根据每个样本到聚类中心点的距离，把样本归类到相距它最近的聚类中心代表的类中，再计算样本均值，若相邻的两个聚类中心无变化结束迭代，否则改过程重复进行。

算法流程：

- 初始化聚类中心点：随机选择 k 个初始点，作为k个簇的初始聚类中心点
- 对样本进行聚类：根据每个样本x_i到k个类中心的距离，将每个样本划分到距离其最近的类中 心的类中，得到聚类结果
- 计算新的聚类中心点：计算当前各个类中的样本的均值，并将其作为新的聚类中心点
- 判断聚类中心点是否发生变化（或变化小于某个给定的阈值），若改变，则重复执行上述步骤2、3；若不变，则输出聚类结果为最终分类结果。

优点
迭代式算法：实现简单，方便理解；
缺点
K均值聚类属于启发式方法，不能保证收敛到全局最优，初始中心的选择会直接影响到聚类结果。
类别数k需要预先指定，而在实际应用中最优的k值是不知道的。

14.3 基于K-Means聚类算法实现鸢尾花数据及分类

```
 9  # 加载数据集
10  iris = datasets.load_iris()
11
12  # 划分数据集
13  iris_X_train , iris_X_test , iris_y_train , iris_y_test = train_test_split(iris.data,iris.target,test_size=0.2)
14
15  # KMeans算法模型，包含3类标签
16  kmeans  = KMeans(n_clusters = 3)
17
18  # 模型训练
19  kmeans_fit=kmeans.fit(iris_X_train)
20
21  # 模型预测
22  y_predict=kmeans.predict(iris_X_train)
23  score=metrics.accuracy_score(iris_y_train,kmeans.predict(iris_X_train))
24  print('准确率:{0:f}'.format(score))  #显示准确率
 9  # 加载数据集
10  iris = datasets.load_iris()
11
12  # 划分数据集
13  iris_X_train , iris_X_test , iris_y_train , iris_y_test = train_test_split(iris.data,iris.target,test_size=0.2)
14
15  # KMeans算法模型，包含3类标签
16  kmeans  = KMeans(n_clusters = 3)
17
18  # 模型训练
19  kmeans_fit=kmeans.fit(iris_X_train)
20
21  # 模型预测
22  y_predict=kmeans.predict(iris_X_train)
23  score=metrics.accuracy_score(iris_y_train,kmeans.predict(iris_X_train))
24  print('准确率:{0:f}'.format(score))  #显示准确率
```

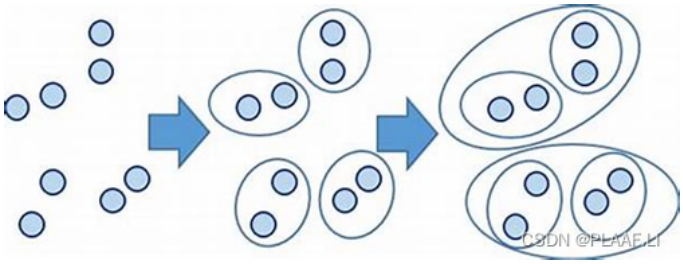
15、层次聚类

层次聚类，试图在不同层次上对数据集进行划分，从而形成树形的聚类结构，数据集的划分采用"自底向上"的聚合策略，也可采用"自顶向下"的分拆策略。因此可分为**聚合层次聚类**与**划分层次聚类**。

聚合层次聚类，采用自底向上的策略。开始时，每个样本对象自己就是一个类，称为 **原子聚类**，然后根据这些样本之间的 相似性，将这些样本对象（原子聚类）进行 **合并**。聚合层次聚类的代表算法为AGNES。

划分层次聚类，采用自顶向下的策略，它首先将所有对象置于同一个簇中，然后逐渐细分为越来越小的簇，直到每个对象自成一簇，或者达到了某个终止条件。**该种方法一般较少使用。**

AGNES(Agglomerative Nesting)是一种自底向上聚合策略的层次聚类算法。它先将数据集中的每一个样本看作一个初始聚类，然后在算法运行的每一步找出距离最近的两个聚类簇进行合并，该过程不断重复，直至达到预设的聚类簇的个数。



AGNES算法流程：

1. 给定包含 n 个对象的数据集 D，聚类簇距离度量函数 d，聚类簇数为 k。
2. 计算所有样本间的距离。
   使用度量函数 d 计算所有样本间的距离。
3. 更新聚类簇及样本间距离。
   将距离最近的两个聚类簇进行合并，计算合并后的聚类簇间的距离。
4. 重复上述步骤 3，直至聚类簇数为设定的参数 k。

代码实现

模型名称：sklearn.cluster.AgglomerativeClustering

模型功能：聚合层次聚类的模型实现，用于自底向上对样本进行聚类。0

模型参数：n_clusters: 聚类簇数；

linkage:指定簇之间距离的度量方式：single(最小距离)、complete(最大距离)、average(平均距离)；

```
# 聚类样本
samples = np.array([[5.1, 3.5, 1.5, 0.2], [4.9, 3.0, 1.4, 0.2],[5.4, 3.9, 1.7, 0.4],
                    [6.4, 3.2, 4.5, 1.5],[5.5, 2.3, 4.0, 1.3], [4.9, 2.4, 3.3, 1.0 ],
                    [6.3, 3.3, 6.0, 2.5], [7.3, 2.9, 6.3, 1.8],[6.7, 2.5, 5.8, 1.8]])

# 模型定义：单连接模型、全连接模型、平均连接模型
model1 = AgglomerativeClustering(n_clusters=3, linkage='single')
model2 = AgglomerativeClustering(n_clusters=3, linkage='complete')
model3 = AgglomerativeClustering(n_clusters=3, linkage='average')

# 模型训练和预测
labels = model1.fit_predict(samples)
print(labels)
```

16、密度聚类

密度聚类依据样本分布的紧密程度来确定聚类结构。从样本密度出发考虑样本间的可连接性，然后基于可连接性样本不断扩展聚类的簇实现聚类的目的。

16.1 密度聚类算法：DBSCAN

DBSCAN的核心思想是先找到密度较高的点，然后把与其相近的高密度点连成一片，进而生成各种簇。

基于一组"邻域"参数 (ϵ,MinPts) 来刻画样本分布的紧密程度。

算法流程：

给定包含 n 个对象的数据集 D ，邻域为 ϵ，密度阈值为 $MinPts$。

首先找到所有的核心对象：根据 $(\epsilon,MinPts)$ 对 n 个对象进行搜索，寻找所有的核心对象，构成核心对象集合。

根据上述的核心对象寻找 D 中所有密度相连的样本，构成簇，若上述核心对象已被访问，则剔除出去。

重复上述过程，直至核心对象集合为空。

优点
不需要指定类簇数目，只需要邻域半径及密度阈值；
可以发现任意形状的类簇，对噪音点不敏感；
缺点
不适合密度差异很大的情形；
复杂度高，为降低复杂性若提前建立距离索引则又会占存储空间；

16.2 基于DBSCAN聚类算法实现鸢尾花数据集分类

```
# 加载数据集
iris = datasets.load_iris()

# 划分数据集
iris_X_train , iris_X_test , iris_y_train , iris_y_test = train_test_split(iris.data,iris.target,test_size=0.2)

# DBSCAN算法模型, eps指定为0.3, 不用指定类别个数
db = DBSCAN(eps=0.3)

# 模型训练和预测
y_predict=db.fit_predict(iris_X_train)
```

17、集成学习AdaBoost

17.1 集成学习

集成学习是这样一个过程，按照某种算法生成多个模型，再将这些模型按照某种方法组合在一起来解决某个智能计算问题。集成学习主要用来提高模型（分类，预测，函数估计等）的性能，或者用来降低模型选择不当的可能性。

随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋向于0

注意：

1、基学习器的误差相互独立

2、事实上，个体学习器的"准确性"和"多样性"本身就存在冲突。一般的，准确性很高之后，要增加多样性就得牺牲准确性

3、如何产生"好而不同"的个体学习器是集成学习研究的核心

4、集成学习大致可分为两大类。第一类：个体学习器间存在强依赖关系、必须串行生成的序列化方法，代表是Boosting；第二类：个体学习器间不存在强依赖关系、可同时生成的并行化方法，代表是Bagging与随机森林。

17.2 Boosting AdaBoost

Boosting的主要思想：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器数目达到事先指定的值T，最终将这T个基学习器进行加权结合。

AdaBoost通过改变数据分布学习多个基本分类器。这些数据分布由上一个基本分类器分类出来的结果决定，确定下一步输入样本的权值。将修改过权值的新数据集送给下层分类器进行训练，最后将每次训练得到的分类器最后融合起来，作为最后的决策分类器。

18、性能指标

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。

$$accuracy = \frac{\text{预测正确的个数}}{\text{总数}}$$

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。

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)

准确率(Accuracy)：分类正确的样本个数占所有样本个数的比例

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

精确率(Precision)：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

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

召回率(Recall)：分类正确的正样本个数占正样本个数的比例

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

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

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

PR曲线：P指精确率，R指查全率，PR曲线用来描述模型的优劣。

19、人工神经网络

19.1 神经网络

神经网络是由具有适应性的简单单元（神经元模型）组成的广泛并行互联的网络, 它的组织能够模拟生物神经系统对真实世界物体所作出的反应。

机器学习中的神经网络通常是指"神经网络学习" 或者机器学习与神经网络两个学科的交叉部分。

19.2 MP神经元模型

输入：来自其他 n 个神经元传递过来的输入信号

处理：输入信号通过带权重的连接进行传递, 神经元接受到总输入值将与神经元的阈值进行比较

输出：通过激活函数的处理以得到输出

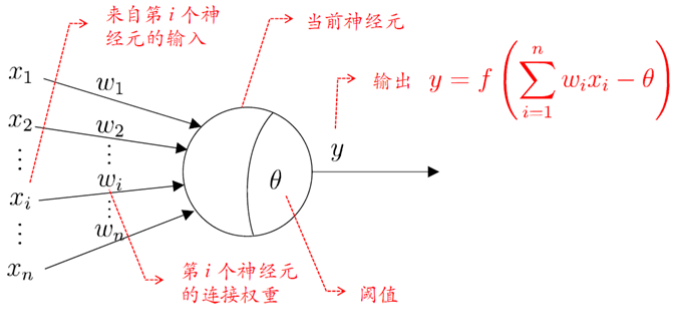
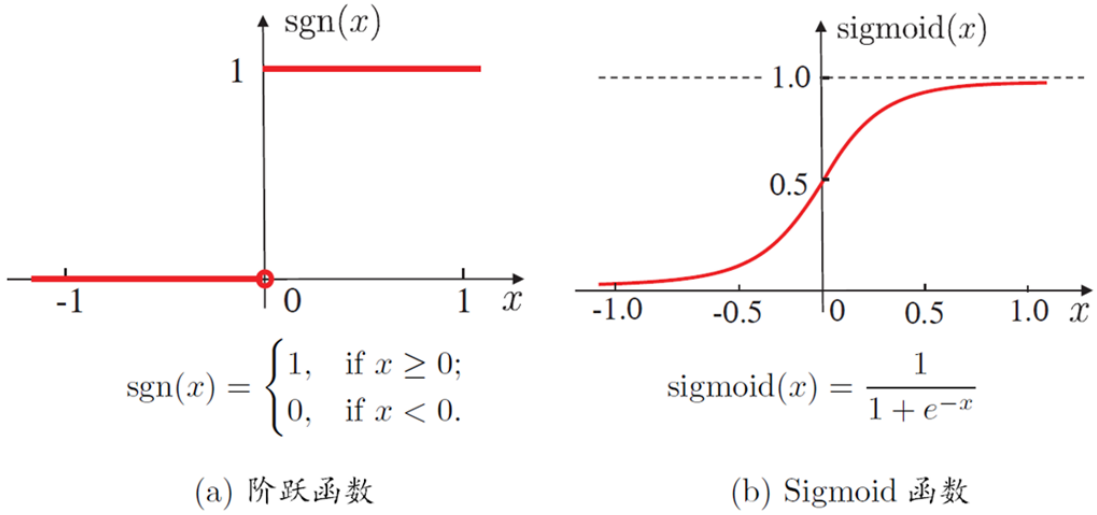


图 5.1　M-P 神经元模型

19.3 感知机

感知机由两层神经元组成, 输入层接受外界输入信号传递给输出层, 输出层是M-P神经元（阈值逻辑单元）

20、激活函数有哪些



$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

(a) 阶跃函数　　(b) Sigmoid 函数

典型的神经元激活函数

理想激活函数是阶跃函数, 0表示抑制神经元而1表示激活神经元
阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数

21、输入层，隐藏层，输出层，损失函数，反向调参策略，全连接网络，偏置向量

输入层：主要用于获取输入的信息，比如黑白照片的像素是黑色还是白色的，大小取决于输入信息的规模

隐藏层：主要进行特征提取，调整权重让隐藏层的神经单元对某种形式形成反应

输出层：用于对接隐藏层并输出模型结果，调整权重以对不同的隐藏层神经元刺激形成正确的反应，输出的兴奋度即为结果

损失函数：是用来度量模型的预测值与真实值的差异程度的运算函数，损失函数越小，模型的鲁棒性就越好。

反向调参策略:

- 将输入示例提供给输入层神经元，逐层将信号前传，直到产生输出结果
- 计算输出层与真实值的误差，将误差使用BP算法传播到整个网络，对连接权重及阈值进行调整
- 该迭代过程循环进行，直到达到某些停止条件为止。（例如训练误差达到了很小的值，或者整个数据集运行了20轮）

全连接神经网络(Multi-Layer Perception, MLP)或者叫多层感知机,是一种连接方式较为简单的人工神经网络结构,属于前馈神经网络的一种,只要有输入层、隐藏层和输出层构成,并且在每个隐藏层中可以有多个神经元。MLP 网络是可以应用于几乎所有任务的多功能学习方法,包括分类、回归,甚至是无监督学习。

偏置向量：偏置单元，它其实就是函数的截距

22、BP神经网络

BP算法是最成功的训练多层前馈神经网络的学习算法，也是现在使用最多的学习算法。

22.1 BP神经网络工作流程

将输入示例提供给输入层神经元，逐层将信号前传，直到产生输出结果

计算输出层与真实值的误差，将误差使用BP算法传播到整个网络，对连接权重及阈值进行调整

该迭代过程循环进行，直到达到某些停止条件为止。（例如训练误差达到了很小的值，或者整个数据集运行了20轮）

22.2 标准、累计BP

标准BP算法
每次针对单个训练样例更新权值与阈值。
参数更新频繁, 不同样例可能抵消, 需要多次迭代。

累计 BP 算法
其优化的目标是最小化整个训练集上的累计误差。

$$E = \frac{1}{m} = \sum_{k=1}^{m} E_k$$

读取整个训练集一遍才对参数进行更新, 参数更新频率较低。

实际应用
但在很多任务中, 累计误差下降到一定程度后, 进一步下降会非常缓慢, 这时标准BP算法往往会获得较好的解, 尤其当训练集非常大时效果更明显

多层前馈网络表示能力
只需要一个包含足够多神经元的隐层, 多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数

多层前馈网络局限
神经网络由于强大的表示能力, 经常遭遇过拟合。表现为：训练误差持续降低, 但测试误差却可能上升
如何设置隐层神经元的个数仍然是个未决问题. 实际应用中通常使用“试错法”调整

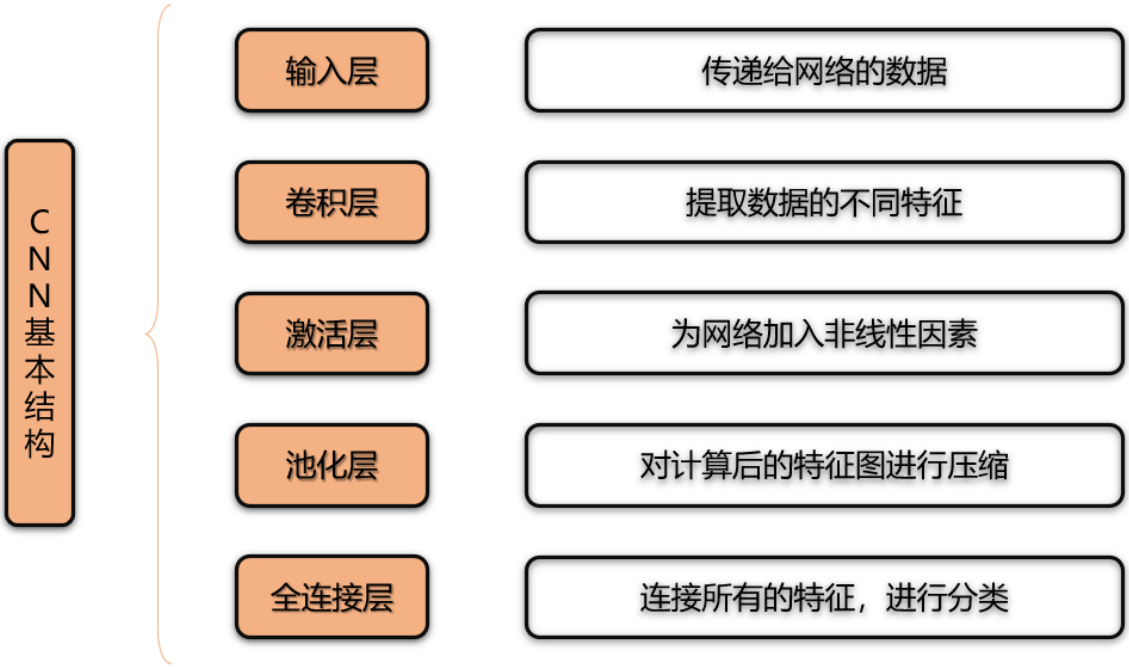
缓解过拟合的策略
早停：在训练过程中, 若训练误差降低, 但验证误差升高, 则停止训练
正则化：在误差目标函数中增加一项描述网络复杂程度的部分, 例如连接权值与阈值的平方和

全局最小与局部最小：参数空间内梯度为零的点, 只要其误差函数值小于邻点的误差函数值, 就是局部极小点

可能存在多个局部极小值, 但却只会有一个全局最小值

模拟退火技术。每一步都以一定的概率接受比当前解更差的结果, 从而有助于跳出局部极小。
随机梯度下降。与标准梯度下降法精确计算梯度不同, 随机梯度下降法在计算梯度时加入

23、卷积神经网络，局部连接，权值共享，卷积层，池化层



23.1 输入层

输入层是对数据进行预处理的阶段，将输入的数据（图像/文字）转换成网络能够计算的数字。

23.2 卷积层

卷积层，卷积神经网络中每层卷积层由若干卷积单元(卷积核)组成，每个卷积单元的参数都是通过反向传播算法优化得到的。

卷积运算的目的是**提取输入的不同特征**，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网络能从低级特征中迭代提取更复杂的特征。（卷积核个数=提取特征的个数，卷积计算=特征抽取）

卷积核（filter/kernel）：用于对输入图像进行共享权值的遍历
步长（stride）：卷积核在图片上移动的大小
填充（padding）：满足输出的图像的维度要求

卷积核元素的总和体现出输出的亮度，若元素总和为1，卷积后的图像与原图像亮度基本一致；若元素总和为0，则卷积后的图像基本上是黑色，其中较亮的部分往往就是提取出图像的某种特征

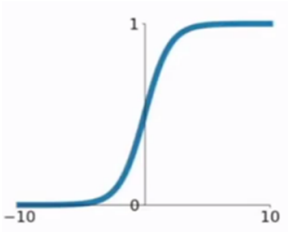
用于图像模糊，平滑处理：均值滤波器、高斯滤波器

卷积核的**步长**代表提取的精度，步长定义了当卷积核在图像上面进行卷积操作的时候，每次卷积跨越的长度。

卷积核深度=上一层数据输入的深度

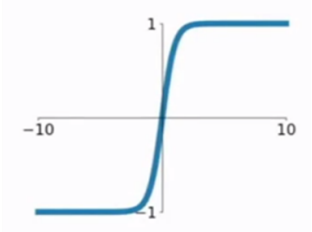
卷积核和图像尺寸不匹配要进行填充

23.3 激活层



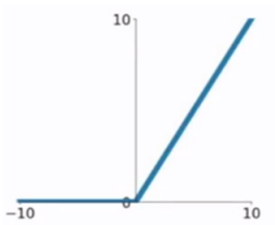
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

sigmod激活函数



$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

tanh激活函数

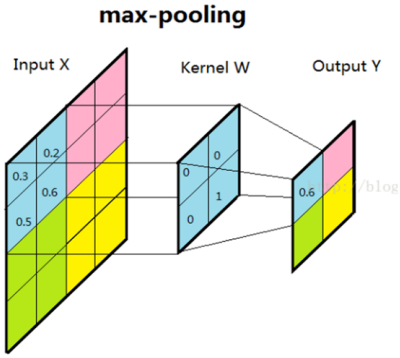


$$f(x) = max(0,x)$$

relu激活函数，最常用

23.4 池化层

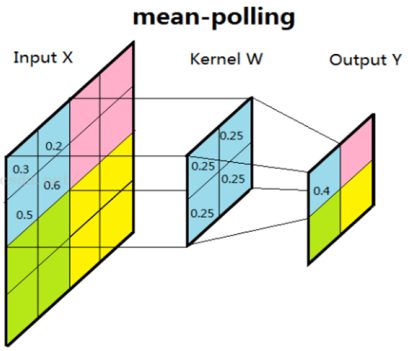
池化层的主要的作用是压缩数据和参数的量（保持最显著的特征），通过去掉上一层的输出中不重要的信息，进一步减少参数数量。Pooling的方法很多，常用方法有最大池化与均值池化。



Max-Pooling:

对邻域内特征点取最大作为最后的特征值

$$\max(0.3,0.2,0.5,0.6) = 0.6$$



Mean-Pooling:

对邻域内特征点取平均作为最后的特征值

$$\frac{0.3+0.2+0.5+0.6}{4} = 0.4$$

23.5 全连接层

将多层的特征映射成一个一维的向量

采用全连接的方式将向量连接向输出层，
打破卷积特征的空间限制
对卷积层获得的不同的特征进行加权
最终目的是得到一个可以对不同类别进行区分的得分

输出层就是获得对应每个类别的得分

23.6 局部连接

作用：降低参数数目，减少网络运算复杂度。

原理：图像的空间联系是局部的像素联系较为紧密，而距离较远的像素相关性则较弱。因而，每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部的信息综合起来就得到了全局的信息，这些局部可以称为局部感知野（亦即卷积核）。

23.7 权值共享

定义：给定一张输入图片，用一个卷积核来卷积这张图，卷积核里的值叫做权重，这张图的每个位置是被同一个卷积核扫的，即卷积的时候所用的权重是一样的。权值共享这个词就是整张图片在使用同一个卷积核内的参数

作用：大减少网络训练参数的同时，还可以实现并行训练

23.8 卷积神经网络训练

本质上是一种输入到输出的映射，只要使用已知的模式对卷积网络加以训练，网络就具有输入输出之间的映射能力，也参照了反向传播算法。

第一阶段向前传播：从样本集中取得样本，将x输入网络，获得输出结果

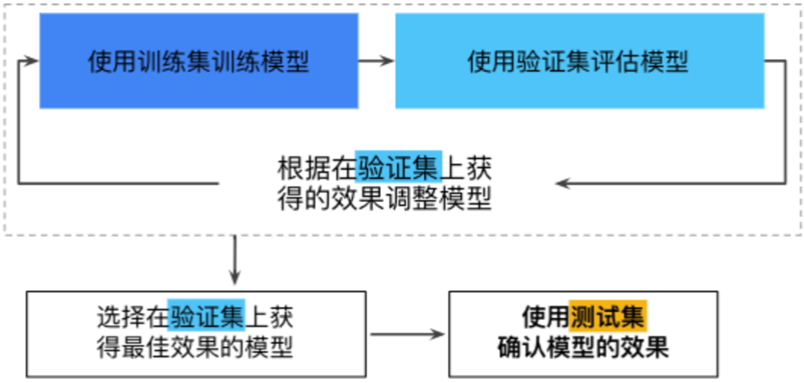
第二阶段向后传播：采用BP算法对误差处理进行更新网络权重

24、LSTM

基于RNN的优化算法-LSTM（长短期记忆），LSTM与RNN结构类似，都是链状结构，但是其中的重复模块有不同的结构。依靠门控机制解决信息遗忘问题。

25、训练集，验证集和测试集

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

1. *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。
2. *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。

26、超参有哪些

机器学习模型中一般有两类参数：

模型参数（Parameter）：需要从数据中学习和估计得到，比如线性回归直线的加权系数（斜率）及其偏差项（截距）。

超参数：机器学习算法中的调优参数，需要人为设定

树的数量或树的深度、矩阵分解中潜在因素的数量、学习率（多种模式）、深层神经网络隐藏层数、k均值聚类中的簇数、正则化器

、正则化系数、初始权重值、优化器优化权重和偏置、聚类中类的个数、话题模型中话题的数量、模型的学习率

27、过拟合和欠拟合

27.1 过拟合

模型学习的参数过多导致拟合函数完美预测训练集，但对新数据的测试集预测结果较差，过拟合通常发生在模型过于复杂的情况下，会导致模型预测性能变弱，增加数据的波动性

改进方法：

从源头获取更多数据（最直观有效）
使用合适的模型（减少网络层数、神经元个数）
使用正则项约束模型的权重，降低模型的非线性
Early stopping使用迭代次数截断防止过拟合，模型对训练数据集迭代收敛之前停止迭代来防止过拟合。
Dropout、交叉验证、决策树剪枝、模型组合（集成学习）

27.2 欠拟合

模型泛化能力差，训练好的模型在训练集表现差、测试集表现也很差。模型拟合程度不高，数据离拟合曲线较远，模型没有很好地捕捉到数据特征。

改进方法：

增加新特征，考虑加入特征组合来增大假设空间
添加多项式特征，线性模型添加二次项、三次项是的模型泛化能力强
减少正则化函数
使用非线性模型，例如核SVM、决策树、深度学习等模型
使用集成学习的方法