机器学习

概述

什么是机器学习

机器学习是一门能够让编程计算机从数据中学习的计算机科学。
一个计算机程序在完成任务T之后，获得经验E，其表现效果为P，如果任务T的性能表现，也就是用以衡量的P，随着E增加而增加，那么这样计算机程序就被称为机器学习系统。
自我完善，自我增进，自我适应。

为什么需要机器学习

- 自动化的升级和维护
- 解决那些算法过于复杂甚至跟本就没有已知算法的问题

机器学习的问题

1. 建模问题
   所谓机器学习，在形式上可这样理解：在数据对象中通过统计或推理的方法，寻找一个接受特定输入X，并给出预期输出Y的功能函数f，即Y=f(X)。
2. 评估问题
   针对已知的输入，函数给出的输出(预测值)与实际输出(目标值)之间存在一定的误差，因此需要构建一个评估体系，根据误差的大小判定函数的优劣。
3. 优化问题
   学习的核心在于改善性能，通过数据对算法的反复锤炼，不断提升函数预测的准确性，直至获得能够满足实际需求的最优解，这个过程就是机器学习。

机器学习的种类

监督学习、无监督学习、半监督学习、强化学习

1. 有监督学习：用已知输出评估模型的性能。
2. 无监督学习：在没有已知输出的情况下，仅仅根据输入信息的相关性，进行类别的划分。
3. 半监督学习：先通过无监督学习划分类别，再根据人工标记通过有监督学习预测输出。
4. 强化学习：通过对不同决策结果的奖励和惩罚，使机器学习系统在经过足够长时间的训练以后，越来越倾向于给出接近期望结果的输出。

批量学习和增量学习

1. 批量学习：将学习的过程和应用的过程截然分开，用全部的训练数据训练模型，然后再在应用场景中实现预测，当预测结果不够理想时，重新回到学习过程，如此循环。
2. 增量学习：将学习的过程和应用的过程统一起来，在应用的同时以增量的方式，不断学习新的内容，边训练边预测。

基于实例的学习和基于模型的学习

1. 根据以往的经验，寻找与待预测输入最接近的样本，以其输出作为预测结果。

年龄	学历	经验	性别	月薪
25	硕士	2	女	中
20	本科	3	男	低
…	…	…	…	…
20	本科	3	男	？

2. 基于模型的学习：根据以往的经验，建立用于联系输出和输入的某种数学模型，将待预测输入代入该模型，预测其结果。
输入	输出
1	2
2	4
3	6| Y = 2 * X
…	
9	?| -> 18

机器学习的一般过程

数据处理

1. 数据收集 （数据检索、数据挖掘、爬虫）
2. 数据清洗
3. 特征工程

机器学习

1. 选择模型 （算法）
2. 训练模型 （算法）
3. 评估模型 （工具、框架、算法知识）
4. 测试模型

业务运维

1. 应用模型
2. 维护模型

机器学习的典型应用

股价预测、推荐引擎、自然语言识别、语音识别、图像识别、人脸识别

机器学习的基本问题

1)回归(Regression)问题：根据已知的输入和输出寻找某种性能最佳的模型，将未知输出的输入代入模型，得到连续的输出。

2)分类(Classification)问题：根据已知的输入和输出寻找某种性能最佳的模型，将未知输出的输入代入模型，得到离散的输出。

3)聚类问题：根据已知输入的相似程度，将其划分为不同的群落。

4)降维问题：在性能损失尽可能小的前提下，降低数据的复杂度。

数据预处理

数据预处理的过程： 输入数据 -> 模型 -> 输出数据

数据样本矩阵

年龄	学历	经验	性别	月薪
25	硕士	2	女	10000
20	本科	3	男	8000
...	...	...	...	...

一行一样本，一列一特征。

数据预处理相关库

```
# 解决机器学习问题的科学计算工具包
import sklearn.preprocessing as sp

          年龄       工作经验          月薪
张三       23          2          10000
李四       21          1          9000
王五       40          18         10500
```

均值移除(标准化)

由于一个样本的不同特征值差异较大，不利于使用现有机器学习算法进行样本处理。**均值移除**可以让样本矩阵中的每一列的平均值为0，标准差为1。

如何使样本矩阵中的每一列的平均值为0呢？

```
例如有一列特征值表示年龄： 17, 20, 23
mean = (17 + 20 + 23)/3 = 20
a' = -3
b' =  0
c' =  3
完成!
```

如何使样本矩阵中的每一列的标准差为1呢？

```
a' = -3
b' =  0
c' =  3
s' = std(a', b', c')
[a'/s',  b'/s',  c'/s']
```

均值移除API：

```
import sklearn.preprocessing as sp
# scale函数用于对函数进行预处理，实现均值移除。
# array为原数组，返回A为均值移除后的结果。
A = sp.scale(array)
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    [17., 100., 4000],
    [20., 80., 5000],
    [23., 75., 5500]])

std_samples = sp.scale(raw_samples)
print(std_samples)
print(std_samples.mean(axis=0))
print(std_samples.std(axis=0))
```

范围缩放

将样本矩阵中的每一列的最小值和最大值设定为相同的区间，统一各列特征值的范围。一般情况下会把特征值缩放至[0, 1]区间。

如何使一组特征值的最小值为0呢？

```
例如有一列特征值表示年龄： [17, 20, 23]
每个元素减去特征值数组所有元素的最小值即可：[0, 3, 6]
```

如何使一组特征值的最大值为1呢？

```
[0, 3, 6]
把特征值数组的每个元素除以最大值即可：[0, 1/2, 1]
```

范围缩放API：

```
# 创建MinMax缩放器
mms = sp.MinMaxScaler(feature_range=(0, 1))
# 调用mms对象的方法执行缩放操作，返回缩放过后的结果
result = mms.fit_transform(原始样本矩阵)
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    [17., 100., 4000],
    [20., 80., 5000],
    [23., 75., 5500]])
print(raw_samples)
mms_samples = raw_samples.copy()
for col in mms_samples.T:
    col_min = col.min()
    col_max = col.max()
    a = np.array([
        [col_min, 1],
        [col_max, 1]])
    b = np.array([0, 1])
    x = np.linalg.solve(a, b)
    col *= x[0]
    col += x[1]
```

```
print(mms_samples)
# 根据给定范围创建一个范围缩放器
mms = sp.MinMaxScaler(feature_range=(0, 1))
# 用范围缩放器实现特征值的范围缩放
mms_samples = mms.fit_transform(raw_samples)
print(mms_samples)
```

归一化

有些情况每个样本的每个特征值具体的值并不重要，但是每个样本特征值的占比更加重要。

	动作	爱情	科幻
老王	20	10	5
小泽	4	2	1
老祁	15	11	13

所以归一化即是用每个样本的每个特征值除以该样本各个特征值绝对值的总和。变换后的样本矩阵，每个样本的特征值绝对值之和为1。

归一化相关API：

```
# array 原始样本矩阵
# norm  范数
#    l1 - l1范数，向量中个元素绝对值之和
#    l2 - l2范数，向量中个元素平方之和
# 返回归一化预处理后的样本矩阵
sp.normalize(array, norm='l1')
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    [17., 100., 4000],
    [20., 80., 5000],
    [23., 75., 5500]])
print(raw_samples)
nor_samples = raw_samples.copy()
for row in nor_samples:
    row /= abs(row).sum()
print(nor_samples)
print(abs(nor_samples).sum(axis=1))
# 归一化预处理
nor_samples = sp.normalize(raw_samples, norm='l1')
print(nor_samples)
print(abs(nor_samples).sum(axis=1))
```

二值化

有些业务并不需要分析矩阵的详细完整数据（比如图像边缘识别只需要分析出图像边缘即可），可以根据一个事先给定的阈值，用0和1表示特征值不高于或高于阈值。二值化后的数组中每个元素非0即1，达到简化数学模型的目的。

二值化相关API：

```
# 给出阈值，获取二值化器
bin = sp.Binarizer(threshold=阈值)
# 调用transform方法对原始样本矩阵进行二值化预处理操作
result = bin.transform(原始样本矩阵)
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    [17., 100., 4000],
    [20., 80., 5000],
    [23., 75., 5500]])
print(raw_samples)
bin_samples = raw_samples.copy()
bin_samples[bin_samples <= 80] = 0
bin_samples[bin_samples > 80] = 1
print(bin_samples)
# 根据给定的阈值创建一个二值化器
bin = sp.Binarizer(threshold=80)
# 通过二值化器进行二值化预处理
bin_samples = bin.transform(raw_samples)
print(bin_samples)
```

独热编码(onehot)

为样本特征的每个值建立一个由一个1和若干个0组成的序列，用该序列对所有的特征值进行编码。

```
两个数    三个数    四个数
1        3        2
7        5        4
1        8        6
7        3        9
为每一个数字进行独热编码：
1-10     3-100    2-1000
7-01     5-010    4-0100
         8-001    6-0010
                  9-0001
编码完毕后得到最终经过独热编码后的样本矩阵：
101001000
010100100
100010010
011000001


          导演       演员       上映地点     类型
战狼1      吴京       余男       内地         战争
钢铁侠1    钢哥       铁蛋       美国         科幻
战狼2      吴京       吴京       内地         战争
          10       100          10       10
          01       010          01       01
          10       001          10       10

          101001010
```

```
          010100101
          100011010


          导演      演员      上映地点     类型
战狼1      吴京   吴京,余男      内地        战争 主旋律
钢铁侠1    钢哥      铁蛋        美国        科幻 特效
战狼2      吴京   吴京,姗姗      内地        战争 主旋律
```

独热编码相关API：

```
# 创建一个独热编码器
# sparse：  是否使用紧缩格式（稀疏矩阵）
# dtype：   数据类型
ohe = sp.OneHotEncoder(sparse=是否采用紧缩格式, dtype=数据类型)
# 对原始样本矩阵进行处理，返回独热编码后的样本矩阵。
result = ohe.fit_transform(原始样本矩阵)
```

```
ohe = sp.OneHotEncoder(sparse=是否采用紧缩格式, dtype=数据类型)
# 对原始样本矩阵进行训练，得到编码字典
encode_dict = ohe.fit(原始样本矩阵)
# 调用encode_dict字典的transform方法 对数据样本矩阵进行独热编码
result = encode_dict.transform(原始样本矩阵)
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    [17., 100., 4000],
    [20., 80., 5000],
    [23., 75., 5500]])
# 创建独热编码器
ohe = sp.OneHotEncoder(sparse=False, dtype=int)
# 用独特编码器对原始样本矩阵做独热编码
ohe_dict = ohe.fit(raw_samples)
ohe_samples = ohe_dict.transform(raw_samples)

ohe_samples = ohe.fit_transform(raw_samples)
print(ohe_samples)
```

标签编码

根据字符串形式的特征值在特征序列中的位置，为其指定一个数字标签，用于提供给基于数值算法的学习模型。

标签编码相关API：

```
# 获取标签编码器
lbe = sp.LabelEncoder()
# 调用标签编码器的fit_transform方法训练并且为原始样本矩阵进行标签编码
result = lbe.fit_transform(原始样本数组)
# 根据标签编码的结果矩阵反查字典 得到原始数据矩阵
samples = lbe.inverse_transform(result)
```

案例：

```
import numpy as np
import sklearn.preprocessing as sp
raw_samples = np.array([
    'audi', 'ford', 'audi', 'toyota',
    'ford', 'bmw', 'toyota', 'ford',
    'audi'])
print(raw_samples)
lbe = sp.LabelEncoder()
lbe_samples = lbe.fit_transform(raw_samples)
print(lbe_samples)
inv_samples = lbe.inverse_transform(lbe_samples)
print(inv_samples)
```

回归模型

线性回归

```
输入      输出
0.5       5.0
0.6       5.5
0.8       6.0
1.1       6.8
1.4       7.0
...
y = f(x)
```

预测函数：$y = w_0 + w_1 x$
x: 输入
y: 输出
w_0和w_1: 模型参数

所谓模型训练，就是根据已知的x和y，找到最佳的模型参数w_0 和 w_1，尽可能精确地描述出输入和输出的关系。

5.0 = w_0 + w_1 × 0.5
5.5 = w_0 + w_1 × 0.6

单样本误差：

根据预测函数求出输入为x时的预测值：$y' = w_0 + w_1 x$，单样本误差为$\frac{1}{2}(y' - y)^2$。

总样本误差：

把所有单样本误差相加即是总样本误差：$\frac{1}{2}\Sigma(y' - y)^2$

损失函数：

loss = $\frac{1}{2}\Sigma(w_0 + w_1 x - y)^2$

所以损失函数就是总样本误差关于模型参数的函数，该函数属于三维数学模型，即需要找到一组w_0 w_1使得loss取极小值。

案例：画图模拟梯度下降的过程

1. 整理训练集数据，自定义梯度下降算法规则，求出w_0， w_1，绘制回归线。

```
import numpy as np
import matplotlib.pyplot as mp
train_x = np.array([0.5, 0.6, 0.8, 1.1, 1.4])
train_y = np.array([5.0, 5.5, 6.0, 6.8, 7.0])
test_x = np.array([0.45, 0.55, 1.0, 1.3, 1.5])
test_y = np.array([4.8, 5.3, 6.4, 6.9, 7.3])

times = 1000    # 定义梯度下降次数
lrate = 0.01    # 记录每次梯度下降参数变化率
epoches = []    # 记录每次梯度下降的索引
w0, w1, losses = [1], [1], []
for i in range(1, times + 1):
    epoches.append(i)
    loss = (((w0[-1] + w1[-1] * train_x) - train_y) ** 2).sum() / 2
    losses.append(loss)
    d0 = ((w0[-1] + w1[-1] * train_x) - train_y).sum()
    d1 = (((w0[-1] + w1[-1] * train_x) - train_y) * train_x).sum()
    print('{:4}> w0={:.8f}, w1={:.8f}, loss={:.8f}'.format(epoches[-1],
w0[-1], w1[-1], losses[-1]))
    w0.append(w0[-1] - lrate * d0)
    w1.append(w1[-1] - lrate * d1)

pred_test_y = w0[-1] + w1[-1] * test_x
mp.figure('Linear Regression', facecolor='lightgray')
mp.title('Linear Regression', fontsize=20)
mp.xlabel('x', fontsize=14)
mp.ylabel('y', fontsize=14)
mp.tick_params(labelsize=10)
mp.grid(linestyle=':')
mp.scatter(train_x, train_y, marker='s', c='dodgerblue', alpha=0.5, s=80,
label='Training')
mp.scatter(test_x, test_y, marker='D', c='orangered', alpha=0.5, s=60,
label='Testing')
mp.scatter(test_x, pred_test_y, c='orangered', alpha=0.5, s=80,
label='Predicted')
mp.plot(test_x, pred_test_y, '--', c='limegreen', label='Regression',
linewidth=1)
mp.legend()
mp.show()
```

1. 绘制随着每次梯度下降，w_0，w_1，loss的变化曲线。

```
w0 = w0[:-1]
w1 = w1[:-1]

mp.figure('Training Progress', facecolor='lightgray')
mp.subplot(311)
mp.title('Training Progress', fontsize=20)
mp.ylabel('w0', fontsize=14)
mp.gca().xaxis.set_major_locator(mp.MultipleLocator(100))
mp.tick_params(labelsize=10)
mp.grid(linestyle=':')
mp.plot(epoches, w0, c='dodgerblue', label='w0')
mp.legend()
mp.subplot(312)
mp.ylabel('w1', fontsize=14)
```

```
mp.gca().xaxis.set_major_locator(mp.MultipleLocator(100))
mp.tick_params(labelsize=10)
mp.grid(linestyle=':')
mp.plot(epoches, w1, c='limegreen', label='w1')
mp.legend()

mp.subplot(313)
mp.xlabel('epoch', fontsize=14)
mp.ylabel('loss', fontsize=14)
mp.gca().xaxis.set_major_locator(mp.MultipleLocator(100))
mp.tick_params(labelsize=10)
mp.grid(linestyle=':')
mp.plot(epoches, losses, c='orangered', label='loss')
mp.legend()
```

1. 基于三维曲面绘制梯度下降过程中的每一个点。

```
import mpl_toolkits.mplot3d as axes3d

grid_w0, grid_w1 = np.meshgrid(
    np.linspace(0, 9, 500),
    np.linspace(0, 3.5, 500))

grid_loss = np.zeros_like(grid_w0)
for x, y in zip(train_x, train_y):
    grid_loss += ((grid_w0 + x*grid_w1 - y) ** 2) / 2

mp.figure('Loss Function')
ax = mp.gca(projection='3d')
mp.title('Loss Function', fontsize=20)
ax.set_xlabel('w0', fontsize=14)
ax.set_ylabel('w1', fontsize=14)
ax.set_zlabel('loss', fontsize=14)
ax.plot_surface(grid_w0, grid_w1, grid_loss, rstride=10, cstride=10,
cmap='jet')
ax.plot(w0, w1, losses, 'o-', c='orangered', label='BGD')
mp.legend()
```

1. 以等高线的方式绘制梯度下降的过程。

```
mp.figure('Batch Gradient Descent', facecolor='lightgray')
mp.title('Batch Gradient Descent', fontsize=20)
mp.xlabel('x', fontsize=14)
mp.ylabel('y', fontsize=14)
mp.tick_params(labelsize=10)
mp.grid(linestyle=':')
mp.contourf(grid_w0, grid_w1, grid_loss, 10, cmap='jet')
cntr = mp.contour(grid_w0, grid_w1, grid_loss, 10,
                  colors='black', linewidths=0.5)
mp.clabel(cntr, inline_spacing=0.1, fmt='%.2f',
          fontsize=8)
mp.plot(w0, w1, 'o-', c='orangered', label='BGD')
mp.legend()
mp.show()

```