Machine Learning

课件 1　　11.10.2021

一．What is Machine Learning?

机器学习简单流程：

1. 使用大量和任务相关的数据集来训练模型；
2. 通过模型在数据集上的误差不断迭代训练模型，得到对数据集拟合合理的模型；
3. 将训练好调整好的模型应用到真实的场景中；

我们最终的目的是将训练好的模型部署到真实的环境中，希望训练好的模型能够在真实的数据上得到好的预测效果，换句话说就是希望模型在真实数据上预测的结果误差越小越好。我们把模型在真实环境中的误差叫做**泛化误差**，***最终的目的是希望训练好的模型泛化误差越低越好***。

二．Small demo : Iris flower data set.

Iris 数据集在模式识别研究领域应该是最知名的数据集
前四列为花萼长度，花萼宽度，花瓣长度，花瓣宽度等 4 个用于识别鸢尾花的属性
第 5 列为鸢尾花的类别（包括 Setosa，Versicolour，Virginica 三类）
即通过判定花萼长度，花萼宽度，花瓣长度，花瓣宽度的尺寸大小来识别鸢尾花的

```
# load the iris data set
iris     = load_iris()
x        = iris.data
x_names = iris.feature_names
y        = iris.target
y_names = iris.target_names
```

类别。

1.导入语句 from---import

如果从数据包里导入，直接利用 sklearn 包 datasets 模块导入 import load_iris。如下：from sklearn.datasets import load_iris

iris=load_iris()

attributes=iris.data #获取属性数据

target=iris.target #获取类别数据，这里注意的是已经经过了处理，target 里 0、1、2 分别代表三种类别

labels=irs.feature_names #获取列属性值

2.将数据可视化 (visualize the data)

```
# visualize the data
def show_iris(x, y, labels=None, symbol_label=None, **kwargs):
    i, j, k = 2, 1, 0          # choose some coordinates
    if labels is None:
        labels = {'x':x_names[i], 'y':x_names[j], 'z':x_names[k],
                  'color':'species', 'symbol': symbol_label}
    fig = scatter_3d(x    = x[:,i],       # x axis of plot
                     y    = x[:,j],       # y axis of plot
                     z    = x[:,k],       # z axis of plot
                     color  = [y_names[n] for n in y],
                     labels = labels,
                     **kwargs)
    fig['layout']['scene']['aspectmode'] = "data"
    return fig
show_iris(x, y)
```

4. **将数据分为训练集和测试集** (traning and testing sets)

```
# split it into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=42)
x_all = vstack([x_train, x_test])    # first comes the train, then the test examples
y_all = hstack([y_train, y_test])
train_test = x_train.shape[0]*['train'] + x_test.shape[0]*['test']
```

1.训练集与测试集

前面说到我们既不能通过直接将泛化误差作为了解模型泛化能力的信号，因为在部署环境和训练模型之间往复，代价很高，也不能使用模型对训练数据集的拟合程度来作为了解模型泛化能力的信号，因为我们获得的数据往往不干净。

更好的方式就是将数据分割成两部分：训练集和测试集。我们可以使用训练集的数据来训练模型，然后用测试集上的误差作为最终模型在应对现实场景中的泛化误差。有了测试集，我们想要验证模型的最终效果，只需将训练好的模型在测试集上计算误差，即可认为此误差即为泛化误差的近似，我们只需让我们训练好的模型在测试集上的误差最小即可。

这里有几点需要注意：

1. 通常将数据集的 80%作为训练集，20%作为测试集；
2. 通常需要在开始构建模型之前把数据集进行划分，防止数据窥探偏误，也就是说我们避免了解太多关于测试集中的样本特点，防止我们认为的挑选有助于测试集数据的模型，这样的结果会过于乐观，但是实际上并没有预期的那样优秀；
3. 通常我们在构建模型的时候需要将数据进行处理，包括一些数据的清洗，数据的特征缩放（标准化或者归一化），此时我们只需要在训练集上进行这些操作，然后将其在训练集上得到的参数应用到测试集中，也就是说，在工作流程中，你不能使用在测试数

据集上计算的得到的任何结果。比如：我们得到的属性中可能有缺失值，因为在这些操作之前，我们已经把数据集分成了训练集和测试集，通常的做法是通过计算属性值的中位数来填充缺失值，注意此时计算属性值的中位数是通过训练集上的数据进行计算的，当我们得到一个模型的时候，如果想要测试模型的测试误差来近似泛化误差的时候，可能此时的测试集也会有一些缺失值，此时对应属性的缺失值是通过训练集计算的中位数来进行填充的；

4. 由于测试集作为对泛化误差的近似，所以训练好模型，最后在测试集上近似估计模型的泛化能力。此时假设有两个不同的机器学习模型，犹豫不决的时候，可以通过训练两个模型，然后对比他们在测试数据上的泛化误差，选择泛化能力强的模型。

如何将数据集分为训练集和测试集呢

通过 sklearn 实现：

```
from sklearn.model_selection import train_test_split
#data:需要进行分割的数据集
#random_state:设置随机种子，保证每次运行生成相同的随机数
#test_size:将数据分割成训练集的比例
train_set, test_set = train_test_split(data, test_size=0.2,
random_state=42)
```

1. **stack()函数** 原型为：stack(arrays, axis=0)，arrays 可以传数组和列表
   列 a= [[1,2,3],[4,5,6]] print(a)
   增加一维，新维度下标为 0 – c = np.stack(a,axis =0)
   输出为 [[1 2 3]
   [4 5 6]]

增加一维，新维度下标为 1 --- c =np.stack(a,axis = 1)
输出为 [[1 4]
[2 5]
[3 6]]

2. **hstack 函数**
函数原型：hstack(tup) ，参数 tup 可以是元组，列表，或者 numpy 数组，返回结果为 numpy 的数组。看下面的代码体会它的含义
a = [[1],[2],[3]]
b = [[1],[2],[3]]
c = [[1],[2],[3]]
d = [[1],[2],[3]]
print(np.hstack((a,b,c,d)))
输出 [[1 1 1 1]
[2 2 2 2]
[3 3 3 3]]
其实就是水平(按列顺序)把数组给堆叠起来，vstack()函数正好和它相反
输出[[1]
[2]…它是垂直（按照行顺序）的把数组给堆叠起来
3.shape 函数
它的功能是查看矩阵或者数组的维数
建立一个 3×3 的单位矩阵 e, e.shape 为（3，3），表示 3 行 3 列,第一维的长度为 3，第二维的长度也为 3

分类算法 Classification (supervised learning)

监督式学习，是机器学习的一种方法，可以由训练资料中学到或创建一个模式（函数 / learning model），并依此模式推测新的实例。
训练资料是由输入对象（通常是向量）和预期输出所组成。函数的输出可以是一个连续的值（称为回归分析），或是预测一个分类标签（称作分类）

什么是「分类」

虽然我们人类都不喜欢被分类，被贴标签，但数据研究的基础正是给数据“贴标签”进行分类。类别分得越精准，我们得到的结果就越有价值。

分类是一个有监督的学习过程，目标数据库中有哪些类别是已知的，分类过程需要做的就是把每一条记录归到对应的类别之中。由于必须事先知道各个类别的信息，并且所有待分类的数

据条目都默认有对应的类别，因此分类算法也有其局限性，当上述条件无法满足时，我们就需要尝试聚类分析。

```
# predict the species from the measure petal/sepal lengths
# input = x_train, output = y_train
nn = KNeighborsClassifier(n_neighbors=1)     #
nn.fit(x_train, y_train)                     # learns the parameters
```

通过测量花瓣/萼片的长度来预测物种

输入 = x_train，输出 = y_train

nn = KNeighborsClassifier(n_neighbors=1) #

nn.fit(x_train, y_train) # 学习参数

. KNN 算法

K 近邻(k-Nearest Neighbor，KNN)分类算法的核心思想是如果一个样本在特征空间中的 k 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别。KNN 算法可用于多分类，KNN 算法不仅可以用于分类，还可以用于回归。通过找出一个样本的 k 个最近邻居，将这些邻居的属性的平均值赋给该样本，作为预测值。

KNeighborsClassifier 在 scikit-learn 在 sklearn.neighbors 包之中。KNeighborsClassifier 使用很简单，三步：

1）创建 KNeighborsClassifier 对象，

2）调用 fit 函数，

3）调用 predict 函数进行预测。

```
# calculate the train accuracy
correct_train = nn.predict(x_train) == y_train
print(correct_train.mean())
show_iris(x_train, y_train,
          symbol          = correct_train,
          symbol_sequence = ['circle', 'x'],
          symbol_label    = 'correct?',
          title = 'training')
```

```
# calculate the test accuracy
correct_test = nn.predict(x_test) == y_test
print(correct_test.mean())
show_iris(x_test, y_test,
          symbol          = correct_test,
          symbol_sequence = ['circle', 'x'],
          symbol_label    = 'correct?',
          title='test set')
```

Regression(supervised learning) 回归(监督学习)

回归：预测连续的、具体的数值。

Regression (supervised learning)

If the output set is continuous and available for training, we have a regression problem.

```
# predict one of the measured lengths from the others
# input = x_train[:3], output = xtrain[3]
lr = LinearRegression()
lr.fit(x_train[:,:3], x_train[:,3])   # predict last from previous coordinates
print(lr.coef_)
print(lr.intercept_)
```

```
# calculate the train error
mean((lr.predict(x_train[:,:3]) - x_train[:,3])**2)
```

```
# calculate the test error
mean((lr.predict(x_test[:,:3]) - x_test[:,3])**2)
```

聚类　无监督学习：数据没有目标属性。发现数据中存在的内在结构及规律

聚类的形式定义：

• 聚类的形式定义

形式化地说，假定样本集 $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ 包含 m 个无标记样本，每个样本 $\boldsymbol{x}_i = (x_{i1}; x_{i2}; \ldots; x_{in})$ 是一个 n 维特征向量，则聚类算法将样本集 D 划分为 k 个不相交的簇 $\{C_l \mid l = 1, 2, \ldots, k\}$，其中 $C_{l'} \bigcap_{l' \neq l} C_l = \varnothing$ 且 $D = \bigcup_{l=1}^{k} C_l$. 相应地，我们用 $\lambda_j \in \{1, 2, \ldots, k\}$ 表示样本 $\boldsymbol{x}_j$ 的“簇标记”(cluster label)，即 $\boldsymbol{x}_j \in C_{\lambda_j}$. 于是，聚类的结果可用包含 m 个元素的簇标记向量 $\boldsymbol{\lambda} = (\lambda_1; \lambda_2; \ldots; \lambda_m)$ 表示.

Clustering (unsupervised learning)

If the output set is discrete and not available for training, we have a clustering problem.

```
# invent clusters that act like labels
km = KMeans(n_clusters=3)
km.fit(x_train)
z_train = km.predict(x_train)
z_test  = km.predict(x_test)
```

```
show_iris(x_train, y_train, symbol = z_train, symbol_label = 'cluster', title='training set')
```

PCA （Principal Component Analysis） 是一种常见的数据分析方式，常用于高维数据的降维，可用于提取数据的主要特征分量。

PCA 的数学推导可以从最大可分型和最近重构性两方面进行，前者的优化条件为划分后方差最大，后者的优化条件为点到划分平面距离最小，

PCA (unsupervised learning)

If the output set is continuous and not available for training, we have a dimensionality reduction problem.

```
pca = PCA(n_components=3)
pca.fit(x_train)
print(pca.explained_variance_ratio_)
z_train = pca.transform(x_train)
z_test  = pca.transform(x_test)
```

```
show_iris(z_train, y_train, labels={})
```

Types of machine learning

Supervised learning:

- discrete output: classification
- continuous output: regression
- . . . , e.g. graphs or other structures

Unsupervised learning:

- discrete output: clustering
- continuous output: dimensionality reduction

Semi-supervised learning:

- . . .

Reinforcement learning:

Discrete 离散的 | count 连续的 | Graph

Super- Classification 分类|regression 回归

Unsuper- clustering 聚类 |dimension-reduction 降维 PCA

Semi-supre

Reinforcement learning 强化学习

Tools of machine learning

- probability theory (e.g. Bayes' rule)

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

- linear algebra (e.g. vectors and matrices)

$$y = Ax + n$$

- optimization (e.g. gradient descent)

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g(x) \leq 0 \\ &\qquad\quad h(x) = 0 \end{aligned}$$

- computer science

  all the things you learned so far and more!