基于支持向量机的手写数字识别

github地址

1、读取 MNIST 数据

它包含了四个部分:

1. Training set images: train-images-idx3-ubyte.gz (包含 60,000 个样本)
2. Training set labels: train-labels-idx1-ubyte.gz (包含 60,000 个标签)
3. Test set images: t10k-images-idx3-ubyte.gz (包含 10,000 个样本)
4. Test set labels: t10k-labels-idx1-ubyte.gz (包含 10,000 个标签)

图片是以字节的形式进行存储, 需要把它们读取到 NumPy array 中, 以便训练和测试算法。

```
def load_mnist(path, kind='train'):
    labels_path = os.path.join(path, '%s-labels.idx1-ubyte' % kind)
    # os.path.join()函数用于路径拼接文件路径
    images_path = os.path.join(path, '%s-images.idx3-ubyte' % kind)
    with open(labels_path, 'rb') as lbpath:
        magic, n = struct.unpack('>II', lbpath.read(8))
        labels = np.fromfile(lbpath, dtype=np.uint8)
    with open(images_path, 'rb') as imgpath:
        magic, num, rows, cols = struct.unpack(">IIII", imgpath.read(16))
        images = np.fromfile(imgpath, dtype=np.uint8).reshape(len(labels), 784)
    return images, labels
```

load_mnist 函数返回两个数组，第一个是一个 n x m 维的 NumPy array(images)， 这里的 n 是样本数(行数)， m 是特征数(列数)。 训练数据集包含 60,000 个样本，测试数据集包含 10,000 样本。在 MNIST 数据集中的每张图片由 28 x 28 个像素点构成， 每个像素点用一个灰度值表示.。在这里，将 28 x 28 的像素展开为一个一维的行向量，这些行向量就是图片数组里的行(每行 784 个值, 或者说每行就是代表了一张图片).。load_mnist 函数返回的第二个数组(labels) 包含了相应的目标变量，也就是手写数字的类标签(整数 0-9)。

2、划分训练集和测试集

数据集已经划分为训练集和测试集，其中训练集60000个，测试集10000个。

3、特征缩放

由于每个样本都是一个一维像素矩阵，取值为0-255，于是可让每个分量都除以255，这样既可以缩放到[0,1]范围，又能保持数据分布。

4、构建支持向量机模型

这里使用sklearn中的svm包，模型训练语句如下：

```
classifier = svm.SVC(C=1.0, kernel='poly',decision_function_shape='ovo')  # ovo:
多对多策略
classifier.fit(X_train, y_train.ravel())
```

1. C：SVC的惩罚参数C，默认值是1.0。C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样对训练集测试时准确率很高，但泛化能力弱。C值小，对误分类的惩罚减小，允许容错，将他们当成噪声点，泛化能力较强。
2. kernel ： 核函数，默认是'rbf'，可以是'linear'（线性核），'poly'（多项式核），'rbf'（RBF函数），'sigmoid'。
3. decision_function_shape ： ‘ovo’, ‘ovr’ or None，default=None3（选用ovr，一对多）。

本次实验分别选择rbf、linear、poly三个核函数。

```
#模型保存与加载：
import joblib
joblib.dump(classifier, "model.m")
model1 = joblib.load("model.m")
```

5、模型评估

查准率：

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

召回率：

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

F1：

$$F1 = 2 * \frac{P * R}{P + R}$$

```
#多项式核函数模型
查准率： [0.983, 0.974, 0.978, 0.981, 0.974, 0.97, 0.983, 0.975, 0.979, 0.975]
召回率 [0.989, 0.993, 0.975, 0.974, 0.984, 0.974, 0.978, 0.968, 0.974, 0.96]
F1值： [0.986, 0.983, 0.976, 0.977, 0.979, 0.972, 0.98, 0.971, 0.976, 0.967]
```

```
#线性核函数模型
查准率： [0.952, 0.97, 0.925, 0.906, 0.936, 0.915, 0.957, 0.952, 0.937, 0.949]
召回率 [0.977, 0.989, 0.937, 0.938, 0.959, 0.9, 0.95, 0.931, 0.9, 0.914]
F1值： [0.964, 0.979, 0.931, 0.922, 0.947, 0.907, 0.953, 0.941, 0.918, 0.931]
```

```
#径向基核模型
查准率： [0.98, 0.989, 0.976, 0.975, 0.983, 0.986, 0.985, 0.976, 0.971, 0.972]
召回率 [0.993, 0.992, 0.975, 0.985, 0.979, 0.976, 0.985, 0.969, 0.975, 0.961]
F1值： [0.986, 0.99, 0.975, 0.98, 0.981, 0.981, 0.985, 0.972, 0.973, 0.966]
```

由于有10种数字，所以要单独计算每一类的查准率、召回率和F1值。列表第i项表示数字i的对应数值。可以看出大部分数值都大于0.95，说明模型预测结果非常好，模型性能优异。