华东师范大学数据科学与工程学院实验报告

课程名称：数据科学与工程算法基础	**年级：**2020	**上机实践成绩：**
指导教师：董启文	**姓名：**李昕原	**学号：**10205501425
上机实践名称：人脸识别		**上机实践日期：**12.20
上机实践编号：	**组号：**	**上机实践时间：**

1. 实验任务

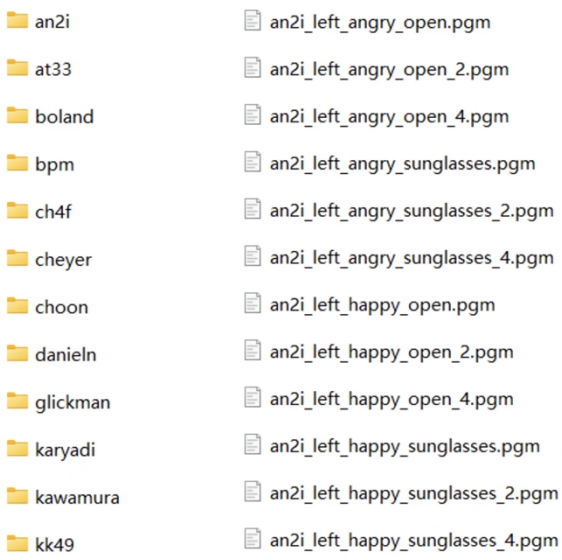
获取 CMU Machine Learning Faces 数据集，包含 20 人中每人 32 张含表情脸图。

- 任务 1：使用机器学习进行人脸分类识别，给出识别准确率
- 任务 2：使用聚类或分类算法发现表情相似的脸图

2. 实验过程

2.1 获取并查看数据集

从 http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html 下载 faces 数据集，查看数据集的基本信息：



该数据集文件有 20 个文件夹，代表 20 个人的不同图片，每个文件夹里存有 32 张图片，而图片的文件名代表了此人的面部表情，包括名字(an2i)、面部方向(left,right,straight,up)、表情(angry,happy,neutral,sad)、是否戴墨镜(open,sunglasses)，通过划分文件名将各个元素写入对应列表，结果如图所示：

```
name: ['an2i', 'at33', 'boland', 'bpm', 'ch4f', 'cheyer', 'choon', 'danieln', 'glickman', 'karyadi', 'kawamura', 'kk49', 'megak', 'mitchell', 'night',
'phoebe', 'saavik', 'steffi', 'sz24', 'tammo']
direction: ['left', 'right', 'straight', 'up']
expression: ['angry', 'happy', 'neutral', 'sad']
sunglasses: ['open', 'sunglasses']
```

2.2 数据预处理及训练集、测试集划分

得到该人脸识别数据集的基本组成后需要将其构造为可以用于机器学习模型训练的数据。主要思路是将 name、direction、expression、sunglasses 中的各个属性定量化（如 sunglasses 中的两个属性

open 和 sunglasses 的取值分别设为 0 和 1），得到 name 取值为[0,19]，direction 取值为[0,3]，expression 取值为[0,3]，sunglasses 取值为[0,1]（均取整数），由于对各个文件遍历时是顺序读取文件，会出现某类特征全部属于训练集或测试集的情况，且训练集与测试集的样本特征重合度不高，从而导致模型训练的效果不佳，因此需要将文件的排序打乱，考虑使用 random.shuffle 实现这一要求：

```
random.shuffle(paths)
```

训练集与测试集的比例定为 3:1，将打乱顺序后的文件名拆分为词并量化为整数，并以字典的方式存储（包括四个基本属性及图像特征），构造训练集及测试集数据并查看：

```
def data(path):
    img, name_, direction_, expression_, sunglasses_ = [], [], [], [], []
    for f in path:
        img.append(np.array(list(Image.open(f).convert('L').getdata())))
        str_ = f.split('_')
        #特征量化
        num = 0
        for i in range(len(name)):
            if name[i] == str_[0].split('/')[2]:
                num = i
        name_.append(num)
        for i in range(len(direction)):
            if direction[i] == str_[1]:
                num = i
        direction_.append(num)
        for i in range(len(expression)):
            if expression[i] == str_[2]:
                num = i
        expression_.append(num)
        for i in range(len(sunglasses)):
            if sunglasses[i] == str_[3]:
                num = i
        sunglasses_.append(num)

    img = np.array(img)
    name_ = np.array(name_)
    direction_ = np.array(direction_)
    expression_ = np.array(expression_)
    sunglasses_ = np.array(sunglasses_)
    return {'image': img, 'name': name_, 'direction': direction_, 'expression':
            expression_, 'sunglasses': sunglasses_}

#训练集与测试集大小比例为3：1
train_size= int(len(paths) * 0.75)
path_1 = paths[:train_size]
path_2 = paths[train_size:]
Data = data(paths)
train_data = data(path_1)
test_data = data(path_2)
print(train_data)
print(train_data['image'].shape)
print(test_data['image'].shape)
```

查看训练集及训练集与测试集规模：

```
{'image': array([[ 26,  25,  23, ..., 112,  33,  27],
       [ 24,  34,  33, ..., 135,  46,   3],
       [  0,   1,  53, ...,  16,  14,  20],
       ...,
       [  7,  23,  29, ..., 122,  61,  41],
       [ 14,  34,  32, ..., 124, 105,   7],
       [ 47,  35,  45, ...,  40,  11,   0]]), 'name': array([ 3,  5, 16,  0,  0, 10,  3,  5,  6,  1,  4,  6,  0,  9,  7,  6,  5,
       18, 13,  1,  8, 16,  9,  7, 10, 16, 18,  7,  2,  8, 13, 19, 19,  0,
       14,  1,  2, 17,  1, 18, 10, 18,  3, 11, 15,  4,  7, 11, 11, 15,  5,
        8, 16,  8, 10,  6, 17,  4, 15, 13, 12,  4,  5, 13, 17, 15,  4, 19,
        9,  7, 12,  3,  6,  4, 19, 19, 10, 19,  4, 15,  5,  3,  2, 12,  1,
        6, 17, 11,  9,  3, 19, 17, 18, 15, 17,  1, 19,  2, 18,  8,  0,  1,
       15, 13, 11, 18, 10,  3,  8,  4, 12,  6,  1,  2,  7,  2, 17,  2, 16,
        9,  9, 16,  0, 16, 15, 19, 16, 13, 18, 17, 16,  0,  7,  2, 13,  4,
       13,  4, 10,  2,  7, 12,  5,  6,  0,  5, 19,  7, 10, 19, 15, 14, 17,
        0,  3, 10, 12,  9, 16, 11,  8, 13, 17, 12, 10,  6,  7,  3, 10, 13,
       18, 12,  8, 14,  7,  0, 15, 12,  9, 19, 18,  4, 16, 13,  9, 19, 11,
        0,  0, 12,  1,  1, 10, 19, 10,  9, 18, 19, 14, 14, 18,  9,  3,  1,
       16, 18,  3,  2, 19,  5, 13, 17, 15,  0, 17,  3,  1, 12, 14, 16,  9,
       14, 18,  4,  1, 19,  9,  3, 12,  6,  9,  4, 11, 10, 15,  3,  6,  7,
       16,  5,  3, 10, 17, 10,  3,  9, 16,  6,  9, 13,  9, 13,  3,  7,  8,
        6,  5, 16,  8, 16,  9,  4,  0, 18,  5,  2, 14, 18,  0, 12,  6, 19,
       10, 12,  6, 13,  5,  7,  2,  1, 12, 10, 19, 18,  6,  2, 16,  5,  7,
       16,  6, 14,  4,  7, 15,  5,  1, 11,  8,  6, 19,  0, 14, 11,  4, 14,
       13, 17,  0,  5,  4,  8, 19,  2,  7, 15,  5,  7, 13, 13, 11,  5, 10,
...
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0])}
(468, 960)
(156, 960)
```

可见训练集的规模为(468,960)，测试集规模为(156,960)。

2.3 使用机器学习进行人脸分类识别，给出识别准确率

在人脸分类识别任务中，name 为需要分类的标签，图像的信息为特征，将通过支持向量机 SVM 与一个神经网络来实现人脸识别分类。

SVM 比较重要的参数有核函数类型 kernal、惩罚系数 C 与核函数参数 gamma，将使用网格搜索寻找最优超参数：

```
#SVM
params = {'kernel':['linear', 'rbf', 'poly', 'sigmoid'],
          'C': [1e-2, 0.1, 1, 10, 1e2],
          'gamma': ['auto', 0.08, 0.2, 0.4, 0.6, 0.8]}
svm = GridSearchCV(SVC(decision_function_shape='ovr'), params)
svm = svm.fit(train_data['image'], train_data['name'])
print(svm.best_params_)
y_pred = svm.predict(test_data['image'])
print(classification_report(test_data['name'], y_pred, target_names=name))
#五折交叉验证
scores = cross_val_score(svm, Data['image'], Data['name'], cv=5, scoring='accur
acy')
print('五折交叉验证结果', scores)
```

```
{'C': 0.01, 'gamma': 'auto', 'kernel': 'poly'}
              precision    recall  f1-score   support

        an2i       1.00      1.00      1.00         7
        at33       1.00      1.00      1.00         9
      boland       1.00      1.00      1.00         8
         bpm       1.00      1.00      1.00        11
        ch4f       1.00      1.00      1.00        10
      cheyer       1.00      1.00      1.00         7
       choon       1.00      1.00      1.00         6
     danieln       0.86      1.00      0.92         6
    glickman       1.00      1.00      1.00         5
     karyadi       1.00      0.89      0.94         9
    kawamura       1.00      1.00      1.00         9
        kk49       1.00      1.00      1.00         7
       megak       1.00      1.00      1.00        10
    mitchell       1.00      1.00      1.00         8
       night       1.00      1.00      1.00         5
      phoebe       1.00      1.00      1.00        10
      saavik       1.00      1.00      1.00         5
      steffi       1.00      1.00      1.00         9
        sz24       1.00      0.88      0.93         8
       tammo       0.88      1.00      0.93         7

    accuracy                           0.99       156
   macro avg       0.99      0.99      0.99       156
weighted avg       0.99      0.99      0.99       156

五折交叉验证结果 [0.984      0.992      1.         0.992      0.97580645]
```

可见最优的超参数组合为多项式核函数 poly、惩罚系数 C=0.01，核函数参数 gamma 为 auto(1/n_features)，最终的 SVM 模型人脸分类识别准确率达到了 0.99，且五折交叉验证的结果也很好，可见分类效果非常理想。

接下来实现一个神经网络来进行人脸分类识别。神经网络的实现依靠深度学习库 Keras 实现，将通过 keras.model.Sequential 构建网络模型，model.compile 与 model.fit 来分别配置训练方法并进行训练。神经网络配置的选择方面，采用一层全连接层，为了保证充分拟合的同时避免过拟合，神经元数目设为 30；激活函数方面，选择 softmax 激活函数，可以增加区分对比度并提高学习效率，同时 softmax 连续可导无拐点，对于梯度下降法非常必要；正则化方面选择 L2 正则化来提高泛化能力；学习率设置为 0.01，采用随机梯度下降法 SGD，损失函数则使用交叉熵损失函数，在模型效果差的时候学习速度比较快，在模型效果好的时候学习速度变慢。

$$L=\frac{1}{N}\sum_{i}L_i=-\frac{1}{N}\sum_{i}\sum_{c=1}^{M}y_{ic}\log(p_{ic})$$

其中:
- M ——类别的数量
- y_{ic} ——符号函数（0 或 1），如果样本 i 的真实类别等于 c 取 1，否则取 0
- p_{ic} ——观测样本 i 属于类别 c 的预测概率

最后要将神经网络输出的 one-hot 编码转为对应类别以通过 sklearn 中的 classification_report 计算准确率，并将训练过程可视化作图：

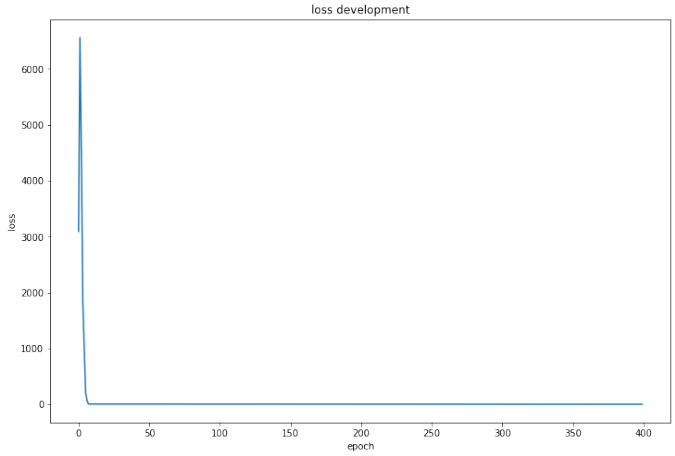
```
#neural networks
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(
```

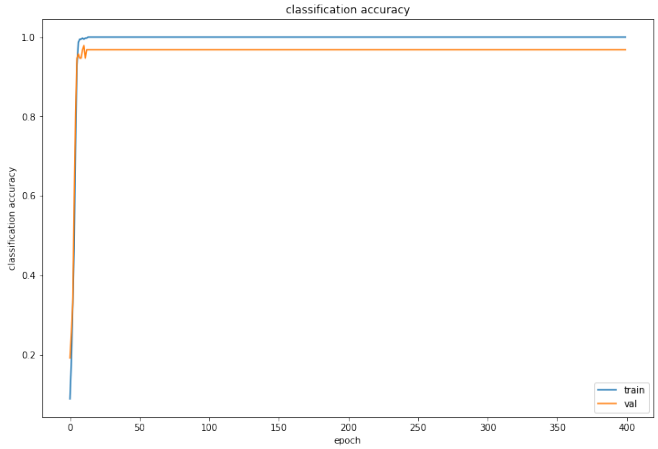
```
        30,
        activation = "softmax",
        kernel_regularizer = tf.keras.regularizers.l2()
    )
])
model.compile(
    optimizer = tf.keras.optimizers.SGD(learning_rate = 0.01),
    loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = False),
    metrics = ['accuracy']
)
history = model.fit(
    train_data['image'], train_data['name'],
    batch_size = 64,
    epochs = 400,
    validation_split = 0.2
)
y_pred_hot = model.predict(test_data['image'])
y_pred=[]
#one-hot 编码转为对应类别
for i in y_pred_hot:
    y_pred.append(np.argmax(i))
print(classification_report(test_data['name'], y_pred, target_names=name))

plt.rcParams['figure.figsize'] = (12,8)
plt.plot(history.history['loss'])
plt.title('loss development')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
plt.plot(history.history['val_accuracy'])
plt.title('classification accuracy')
plt.xlabel('epoch')
plt.ylabel('classification accuracy')
plt.legend()
plt.show()
```

```
Epoch 395/400
6/6 [==============================] - 0s 7ms/step - loss: 2.4864 - accuracy: 1.0000 - val_loss: 3.5399 - val_accuracy: 0.9894
Epoch 396/400
6/6 [==============================] - 0s 8ms/step - loss: 2.4805 - accuracy: 1.0000 - val_loss: 3.5327 - val_accuracy: 0.9894
Epoch 397/400
6/6 [==============================] - 0s 6ms/step - loss: 2.4745 - accuracy: 1.0000 - val_loss: 3.5254 - val_accuracy: 0.9894
Epoch 398/400
6/6 [==============================] - 0s 9ms/step - loss: 2.4686 - accuracy: 1.0000 - val_loss: 3.5183 - val_accuracy: 0.9894
Epoch 399/400
6/6 [==============================] - 0s 8ms/step - loss: 2.4627 - accuracy: 1.0000 - val_loss: 3.5111 - val_accuracy: 0.9894
Epoch 400/400
6/6 [==============================] - 0s 6ms/step - loss: 2.4568 - accuracy: 1.0000 - val_loss: 3.5039 - val_accuracy: 0.9894
```

损失下降与训练集、测试集准确率图：





最终报告：

	precision	recall	f1-score	support
an2i	1.00	1.00	1.00	7
at33	1.00	0.89	0.94	9
boland	1.00	1.00	1.00	8
bpm	0.92	1.00	0.96	11
ch4f	1.00	1.00	1.00	10
cheyer	1.00	1.00	1.00	7
choon	1.00	1.00	1.00	6
danieln	0.86	1.00	0.92	6
glickman	1.00	1.00	1.00	5
karyadi	1.00	1.00	1.00	9
kawamura	1.00	1.00	1.00	9
kk49	0.88	1.00	0.93	7
megak	1.00	1.00	1.00	10
mitchell	1.00	1.00	1.00	8
night	1.00	1.00	1.00	5
phoebe	1.00	1.00	1.00	10
saavik	1.00	1.00	1.00	5
steffi	1.00	1.00	1.00	9
sz24	1.00	0.75	0.86	8
tammo	1.00	1.00	1.00	7
accuracy			0.98	156
macro avg	0.98	0.98	0.98	156
weighted avg	0.98	0.98	0.98	156

可见神经网络人脸分类识别的最终识别准确率达到了 0.98，效果也非常理想。

综上，人脸分类识别任务通过 SVM 支持向量机、神经网络分别实现，准确率均在 0.99 左右，效果很好。

2.4 使用聚类或分类算法发现表情相似的脸图

在该数据集中人物的表情特征包括人脸朝向 direction、人的表情 expression、是否戴墨镜 sunglasses，而发现表情相似脸图这一问题的标签为表情 expression，首先通过任务一中使用的 SVM 与神经网络进行人脸表情识别，代码与之前任务的人脸识别分类类似（仅将标签由 name 修改为 expression），故不再列出，仅列出结果。

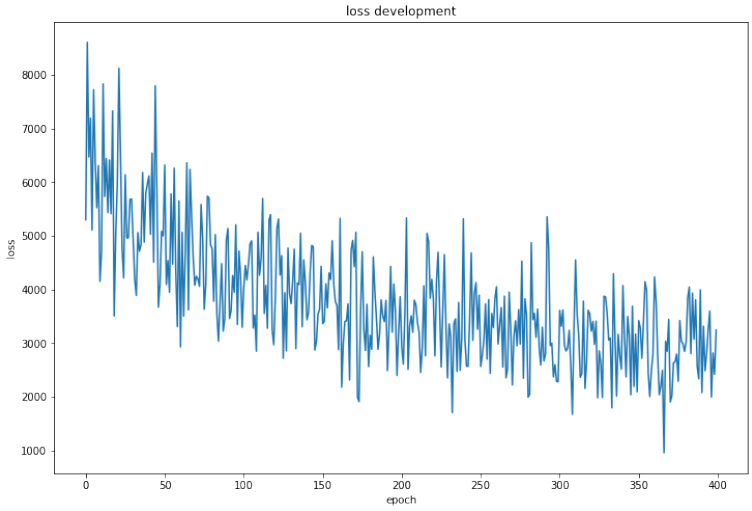
SVM：

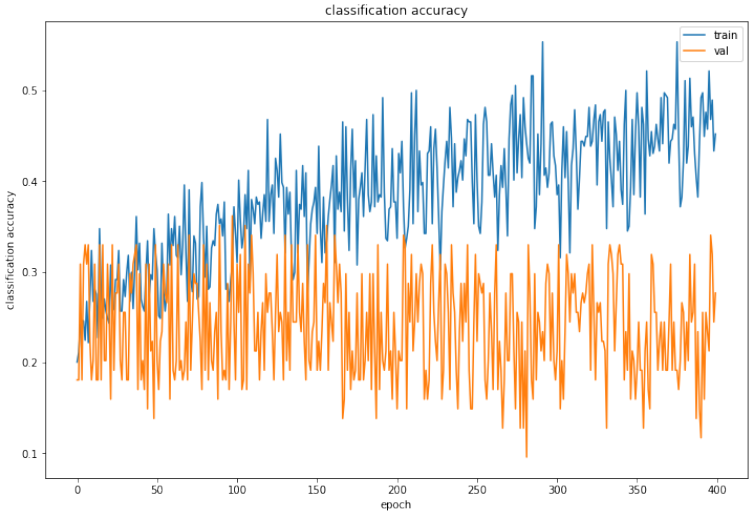
```
{'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf'}
              precision    recall  f1-score   support

       angry       0.00      0.00      0.00        36
       happy       0.00      0.00      0.00        37
     neutral       0.23      1.00      0.38        36
         sad       0.00      0.00      0.00        47

    accuracy                           0.23       156
   macro avg       0.06      0.25      0.09       156
weighted avg       0.05      0.23      0.09       156
```

神经网络及可视化结果：





由图可知神经网络训练过程中损失波动很大且下降幅度小，而训练集与测试集的准确率都有较大波动，可知分类效果较差，报告也验证了这一点：

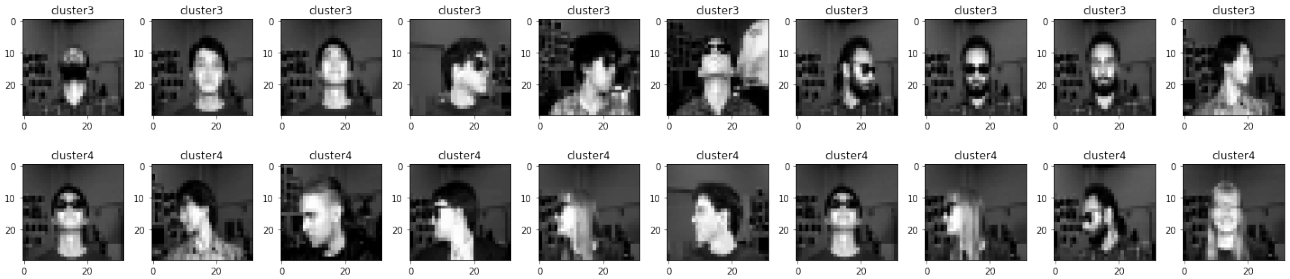
	precision	recall	f1-score	support
angry	0.00	0.00	0.00	36
happy	0.17	0.32	0.22	37
neutral	0.18	0.17	0.17	36
sad	0.21	0.23	0.22	47
accuracy			0.19	156
macro avg	0.14	0.18	0.15	156
weighted avg	0.15	0.19	0.16	156

SVM 的相似表情识别准确率为 0.23，神经网络为 0.19，这与人脸分类识别时两个模型的高分类准确率大相径庭，效果非常不理想。接下来使用聚类算法 KMeans 尝试进行表情相似分类，由于表情 expression 有四类，故无监督学习 KMeans 的 n_clusters 设为 4：

```
clf = cluster.KMeans(
     init='k-means++',
     n_clusters=4,
     random_state=1,
     max_iter = 2000
)
clf.fit(train_data['image'])
y_pred = clf.predict(test_data['image'])
```

聚类后将每一类的前 10 个样本对应的图像输出：

```
cluster_1, cluster_2, cluster_3, cluster_4 = [], [], [], []
for i in range(len(paths) - train_size):
    if(y_pred[i]==0):
        cluster_1.append(i)
    elif(y_pred[i]==1):
        cluster_2.append(i)
    elif(y_pred[i]==2):
        cluster_3.append(i)
    else:
        cluster_4.append(i)
#输出 10 个样本
plt.figure(figsize=(20,20))
for i in range(10):
    plt.subplot(1, 10, i+1)
    grid_data = test_data['image'][cluster_1[i]].reshape(30,32)
    plt.imshow(grid_data, interpolation = "none", cmap = 'gray')
    plt.title('cluster1')
plt.tight_layout()
```



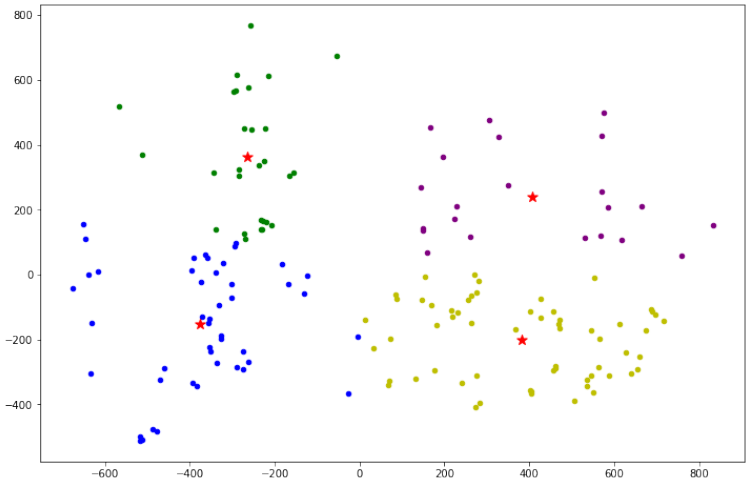
直观上看感觉表情相似聚类效果一般，报告也证明了确实效果不佳：

	precision	recall	f1-score	support
angry	0.21	0.10	0.13	41
happy	0.17	0.22	0.19	27
neutral	0.31	0.50	0.38	48
sad	0.46	0.28	0.34	40
accuracy			0.29	156
macro avg	0.29	0.27	0.26	156
weighted avg	0.30	0.29	0.27	156

KMeans 聚类的准确率仅有 0.29，比 SVM 与神经网络的准确率稍高一点，可见不论是分类算法还是聚类算法在表情相似识别这一问题上均效果不佳，通过 PCA 对数据进行降维再进行 KMeans 聚类：

```
#PCA 降维
pca = PCA(n_components = 2)
pca.fit(train_data['image'])
train_data_pca = pca.transform(train_data['image'])
test_data_pca = pca.transform(test_data['image'])
clf_pca = cluster.KMeans(
    init='k-means++',
     n_clusters=4,
     random_state=1,
     max_iter = 2000
)
clf_pca.fit(train_data_pca)
y_pred = clf.predict(test_data['image'])
print(classification_report(test_data['expression'], y_pred, target_names=expre
ssion))
```

对表情分类识别的聚类簇进行可视化并标记聚类中心：



	precision	recall	f1-score	support
angry	0.21	0.10	0.13	41
happy	0.17	0.22	0.19	27
neutral	0.31	0.50	0.38	48
sad	0.46	0.28	0.34	40
accuracy			0.29	156
macro avg	0.29	0.27	0.26	156
weighted avg	0.30	0.29	0.27	156

准确率依然较低，推测三种分类与聚类算法准确率均较低的原因有以下两点：

- 数据集除人脸外还包括了与人脸无关的背景，因此存在比较严重的噪声；
- 数据集规模较小且是两通道灰度图，人脸表情的特征损失较多。

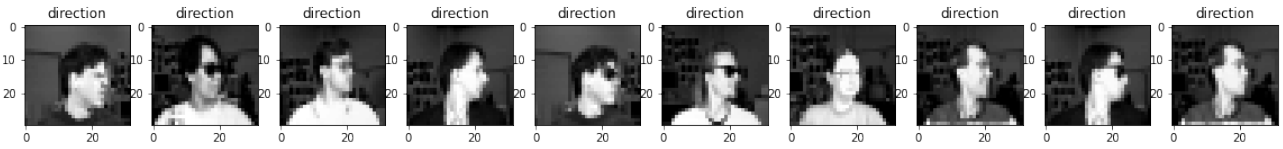
2.5 使用 SVM 识别人脸方向与是否戴墨镜

前面两个任务分别将 name 与 expression 作为标签，而剩下的人脸方向 direction 与是否戴墨镜 sunglasses 并未做过分类识别任务，因此基于 SVM 将这两个特征作为标签进行分类识别，代码与上述相似故不展示，仅展示分类效果及分类结果部分样本的图像：

人脸方向：

```
{'C': 0.01, 'gamma': 'auto', 'kernel': 'poly'}
```

	precision	recall	f1-score	support
left	1.00	1.00	1.00	40
right	1.00	1.00	1.00	34
straight	1.00	0.97	0.99	40
up	0.98	1.00	0.99	42
accuracy			0.99	156
macro avg	0.99	0.99	0.99	156
weighted avg	0.99	0.99	0.99	156

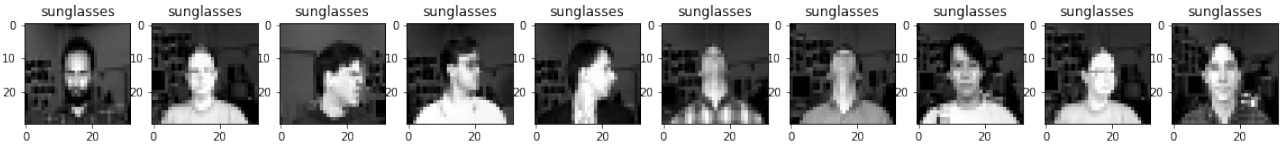


是否戴墨镜：

```
{'C': 0.01, 'gamma': 'auto', 'kernel': 'poly'}
              precision    recall  f1-score   support

        open       0.96      0.94      0.95        78
  sunglasses       0.94      0.96      0.95        78

    accuracy                           0.95       156
   macro avg       0.95      0.95      0.95       156
weighted avg       0.95      0.95      0.95       156
```



样本分别为人脸朝向右侧与不戴墨镜的情况，准确率分别为 0.99 与 0.95 效果均不错，可见除了表情 expression 其余标签的分类识别效果均很优秀，这同样印证了上述人脸表情特征损失较多这一推论。

3. 总结

本次实验获取并查看了 CMU Machine Learning Faces 数据集，通过 SVM 与神经网络模型训练完成了任务 1：人脸分类识别，准确率均在 0.99 左右，取得了不错的效果，后通过 SVM、神经网络与 KMeans 聚类进行任务 2：表情相似识别，但效果不佳，并分析了准确率较低的两点原因（背景噪声严重、人脸表情特征损失较多）。此外还对另外两个标签人脸朝向与是否戴墨镜进行分类识别，准确率也均在 0.95 也上，同样取得了不错的效果。