2016年1月21日木曜日

武藤研究会　最終発表

71547812 環境情報学部１年　藤波 秀麿

　今期は、自分にとって、研究会一期目となり、初めて専門的な勉強に勤しんだ。武藤研究会では、主に、機械学習を学び、それに関する知識を身につけた。

　さて、今回の最終発表で何をしようかと考えたところ、やはり、今期身につけた知識を活かすようなことをしなければならないと感じた。そこで、画像認識とポーカーハンズについて、機械学習をすることにした。また、機械学習するだけではつまらないので、いろいろな方法で機械学習を行い、精度を比較することにした。

　どのような手法の機械学習を用いたかというと、KMeans(K平均法)、SVM(サポートベクターマシン)、Randomforest、PCA(主成分分析)である。KMeansとPCAは、Unsupervised Learning(教師なし学習)であり、SVMとRandomforestはSupervised Learning(教師あり学習)なので、この４つを比較するのは、バランスが取れていて、比較するには適当であると感じた。

　ここから、具体的なコードについて述べる。コードは、すべてipython notebookに書いた。また、数字の画像は、scikitlearnのライブラリの中の一つである、load_digitsを用いた。

　まず、必要なライブラリをインポート。

In [123]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.stats import mode
from sklearn.cluster import KMeans
from sklearn.datasets import load_digits
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.cross_validation import train_test_split
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
```

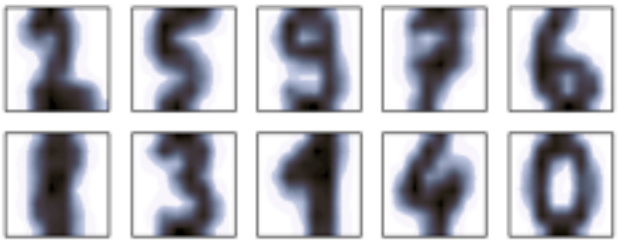
まず、KMeans で画像認識を行う。

Number Recognition with KMeans

In [100]:
```
digits = load_digits()

est = KMeans(n_clusters=10)
clusters = est.fit_predict(digits.data)

fig = plt.figure(figsize=(8, 3))
for i in range(10):
    ax = fig.add_subplot(2, 5, 1 + i, xticks=[], yticks=[])
    ax.imshow(est.cluster_centers_[i].reshape((8, 8)), cmap=plt.cm.binary)
```



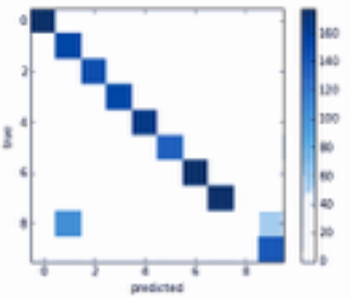
上に表示されている１０個の数字は、機械学習によって認識された、数字である。

In [110]:
```
labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(digits.target[mask])[0]

print(confusion_matrix(digits.target, labels))

plt.imshow(confusion_matrix(digits.target, labels),
           cmap='Blues', interpolation='nearest')
plt.colorbar()
plt.grid(False)
plt.ylabel('true')
plt.xlabel('predicted');
```

```
[[177   0   0   0   1   0   0   0   0   0]
 [  0 154  24   1   0   1   2   0   0   0]
 [  1  10 148  13   0   0   0   3   0   2]
 [  0   7   0 154   0   2   0   7   0  13]
 [  0   5   0   0 166   0   0  10   0   0]
 [  0   0   0   2   2 136   1   0   0  41]
 [  1   3   0   0   0   0 177   0   0   0]
 [  0   2   0   0   0   0   0 177   0   0]
 [  0 105   3   2   0   4   2   5   0  53]
 [  0  20   0   7   0   4   0   9   0 140]]
```



そして、これが結果である。上に表示されている、表と図は、縦軸が本当の数字、横が予測した数字を表している表になっている。この場合、もし斜めの一直線に数字が現れ、その直線以外がすべて０になっていたら、予測は１００％という意味である。

```
In [111]: accuracy_score(digits.target, labels)
Out[111]: 0.79521424596549806
```

そして、精度は、79.5%ほどになった。教師なし学習にしては、まずますの精度を得ることができたのではないかと感じた。

次に、SVMを用いて数字認識を行った。

Number Recognition with SVM

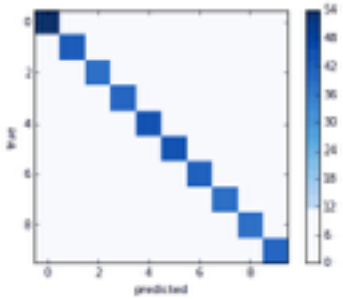
```
In [112]: digits = load_digits(10)
data_train, data_test, label_train, label_test = train_test_split(digits.data, digits.target)
estimator = LinearSVC(C=1.0)
estimator.fit(data_train, label_train)
label_predict = estimator.predict(data_test)
```

ここでは、scikitlearnのライブラリの一つ、train_test_splitを用いて、データを学習データとテストデータに分類している。

```
In [113]: print confusion_matrix(label_test, label_predict)

plt.imshow(confusion_matrix(label_test, label_predict),
           cmap='Blues', interpolation='nearest')
plt.colorbar()
plt.grid(False)
plt.ylabel('true')
plt.xlabel('predicted');
```

```
[[54  0  0  0  0  0  0  1  0  0]
 [ 0 42  0  0  0  0  1  0  3  1]
 [ 0  4 38  0  0  0  0  0  0  0]
 [ 0  0  1 40  0  0  0  0  1  0]
 [ 0  1  0  0 44  0  0  0  1  2]
 [ 0  2  1  0  0 44  0  0  0  0]
 [ 0  0  0  0  1  0 41  0  0  0]
 [ 0  0  0  0  0  0  0 38  1  1]
 [ 0  1  0  2  0  0  1  0 38  0]
 [ 0  0  0  1  0  0  0  0  4 40]]
```



```
In [114]: accuracy_score(label_test, label_predict)
Out[114]: 0.93111111111111111
```

精度は、９３．１%であった。

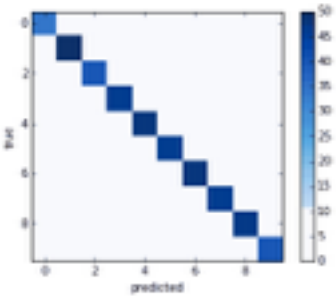
次に、授業で扱ったこともあるRandomForestを用いて、行った。

```
In [115]: clf=RandomForestClassifier(n_estimators=200, min_samples_split=1)
digits = load_digits(10)
data_train, data_test, label_train, label_test = train_test_split(digits.data, digits.target)
clf.fit(data_train, label_train)
label_predict = estimator.predict(data_test)
```

```
In [116]: print confusion_matrix(label_test, label_predict)

plt.imshow(confusion_matrix(label_test, label_predict),
           cmap='Blues', interpolation='nearest')
plt.colorbar()
plt.grid(False)
plt.ylabel('true')
plt.xlabel('predicted');
```

```
[[34  0  0  0  0  0  0  0  0  0]
 [ 0 50  0  0  0  0  0  0  1  0]
 [ 0  1 40  0  0  0  0  0  0  0]
 [ 0  0  0 46  0  0  0  0  0  0]
 [ 0  1  0  0 48  0  0  0  0  0]
 [ 0  0  0  0  0 46  0  0  0  0]
 [ 0  0  0  0  0  0 48  0  0  0]
 [ 0  0  0  0  0  0  0 46  0  1]
 [ 0  0  0  0  0  0  1  0 47  0]
 [ 0  0  0  0  0  0  0  0  0 40]]
```




```
In [117]: accuracy_score(label_test, label_predict)
Out[117]: 0.9888888888888889
```

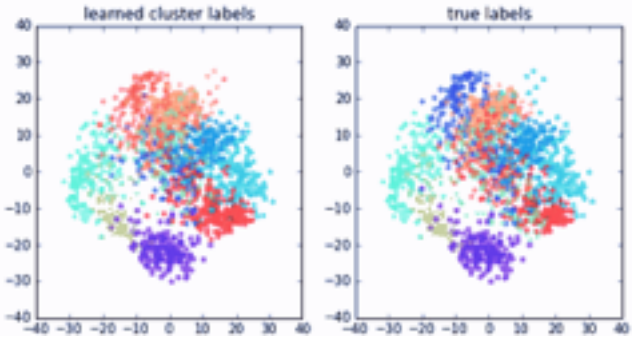
精度は、98.9%になった。RandomForestは、機械学習をする上で非常に有用なライブラリであるということがわかるような結果であると感じた。

最後に、PCAを用いて、行った。

Number Recognition with PCA and KMeans

In [8]:
```
digits = load_digits()
X = digits.data
y = digits.target
target_names = digits.target_names
pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)
est = KMeans(n_clusters=10)
est.fit(X)
fig = plt.figure(figsize=(8, 3))
for i in range(10):
    ax = fig.add_subplot(2, 5, 1 + i, xticks=[], yticks=[])
    ax.imshow(est.cluster_centers_[i].reshape((8, 8)), cmap=plt.cm.binary)
labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(digits.target[mask])[0]
kwargs = dict(cmap = plt.cm.get_cmap('rainbow', 10),
              edgecolor='none', alpha=0.6)
fig, ax = plt.subplots(1, 2, figsize=(8, 4))
ax[0].scatter(X_r[:, 0], X_r[:, 1],c=labels, **kwargs)
ax[0].set_title('learned cluster labels')

ax[1].scatter(X_r[:, 0], X_r[:, 1], c=y, **kwargs)
ax[1].set_title('true labels');
```



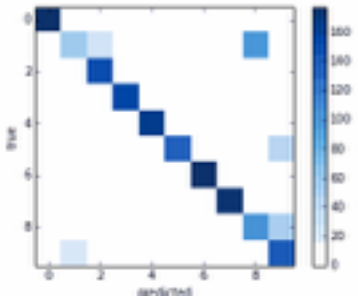
まず、数字の画像一つ一つは、６４次元の情報からなっていたので、それを２次元の情報にまで減らす作業をおこなった。そして、２次元の情報になったものを、KMeansを用いて、クラスタリングを行い、数字を分類した。そこで分類されたのが、上に出ている１０個の数字である。また、２つある表は、６４次元ある情報を２次元にまで減らした情報をプロットしたもので、左側は機械学習で予測したもの、右側は正しいデータで、色付けを行っている。

```
In [10]: print confusion_matrix(y, labels)

         plt.imshow(confusion_matrix(y, labels),
                    cmap='Blues', interpolation='nearest')
         plt.colorbar()
         plt.grid(False)
         plt.ylabel('true')
         plt.xlabel('predicted');

[[177   0   0   0   1   0   0   0   0   0]
 [  0  55  24   1   0   1   2   0  99   0]
 [  1   2 148  13   0   0   0   3   8   2]
 [  0   0   0 154   0   2   0   7   7  13]
 [  0   5   0   0 164   0   0   9   3   0]
 [  0   0   0   1   2 136   1   0   0  42]
 [  1   1   0   0   0   0 177   0   2   0]
 [  0   2   0   0   0   1   0 174   2   0]
 [  0   6   3   2   0   4   2   5 101  51]
 [  0  20   0   6   0   6   0   7   2 139]]

/usr/local/lib/python2.7/dist-packages/numpy/core/fromnumeric.py:2641: VisibleDeprecationWarning: `rank` is deprecated; use the `ndim` attri
bute or function instead. To find the rank of a matrix see `numpy.linalg.matrix_rank`.
  VisibleDeprecationWarning)
```




```
In [11]: accuracy_score(y, labels)
Out[11]: 0.7929883138564273
```

結果としては、７９．３%という風になった。

これは、最初に行った、 KMeansで数字認識を行ったときと同じ結果になった。このことから考えられるのは、今回数字認識において、KMeansのみを使う（正確には、scikit learnのライブラリのKMeans関数を使う)場合と、PCAとKMeansを使う場合を両方やったのだが、２つともほぼ同じ結果に至ったということは、ライブラリに入っているKMeans関数にすでにPCAが組み込まれているのではないということだ。そうでなければ、あまりここまで結果の数字が近い理由を説明できそうにない。

以上のように、私は、４つの方法を用いて、数字認識を行い、精度を比較した。RandomForestが群を抜いて高い数字を叩きだしていたので、この中では最も信用できる機械学習のライブラリと言えるだろう。

今期、機械学習にはじめて触れ、さまざまな機械学習を目の当たりにしてきたが、あまり理論は理解できていないという状況にある。実際問題、理論を理解できていなくて

も、ライブラリの使い方や基本的なコードの書き方をわかってしまっていれば、機械学習は簡単にできるのではあるが、やはり研究を進めていくとなると、一つ一つのライブラリの意味や機械学習のあらゆる手法の理論を理解することは必要不可欠であると考えている。今後は、機械学習の様々な手法を学び、理論を正確に理解したいと考えている。