PCAimplementation

2019 年 3 月 2 日

1 PCA 実装

In [51]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from layers.decisionregionplotfunction import plot_decision_regions as pdr
```

In [52]:
```
# data loading
df_wine=pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.dat
x,y=df_wine.iloc[:,1:].values,df_wine.iloc[:,0].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y,random_state=0)
```

In [53]:
```
# standard scaling
sc=StandardScaler()
x_train_std=sc.fit_transform(x_train)
x_test_std=sc.transform(x_test)
```

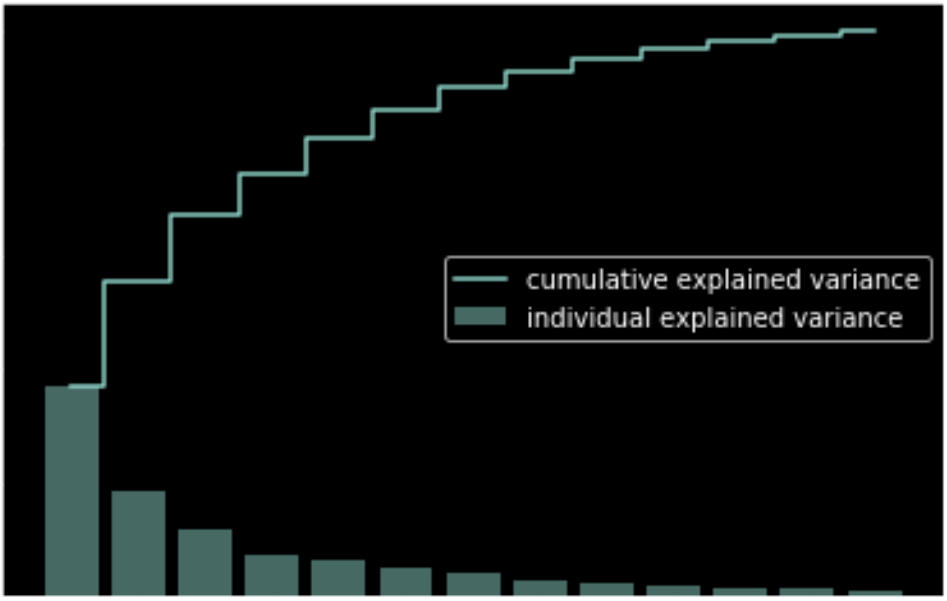
In [54]:
```
# make covariance matrix
cov_mat=np.cov(x_train_std.T)
eigen_vals,eigen_vecs=np.linalg.eig(cov_mat)
print('\nEigenvalues \n%s' % eigen_vals)
```

```
Eigenvalues
[4.84274532 2.41602459 1.54845825 0.96120438 0.84166161 0.6620634
 0.51828472 0.34650377 0.3131368  0.10754642 0.21357215 0.15362835
 0.1808613 ]
```

In [55]:
```
# variance explained ratio(to show importances)
tot=sum(eigen_vals)
var_exp=[(i/tot) for i in sorted(eigen_vals, reverse=True)]
```

```
cum_var_exp=np.cumsum(var_exp)
```

In [56]:
```
# show figure(variance explained ratio & cumlative variance explained ratio)
plt.bar(range(1,14),var_exp,alpha=0.5,align='center',label='individual explained variance')
plt.step(range(1,14),cum_var_exp,where='mid',label='cumulative explained variance')
plt.ylabel('Explaned variance ratio')
plt.xlabel('Principal component index')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



In [57]:
```
# make (eigen value,eigen vectol)tuple list & sort it
eigen_pairs=[(np.abs(eigen_vals[i]),eigen_vecs[:,i])for i in range(len(eigen_vals))]
eigen_pairs.sort(key=lambda k: k[0],reverse=True)
```

In [58]:
```
# extract two eigen vectols which eigen values are biggest
w=np.hstack((eigen_pairs[0][1][:,np.newaxis],eigen_pairs[1][1][:,np.newaxis]))
print('Matrix W:\n',w)
```

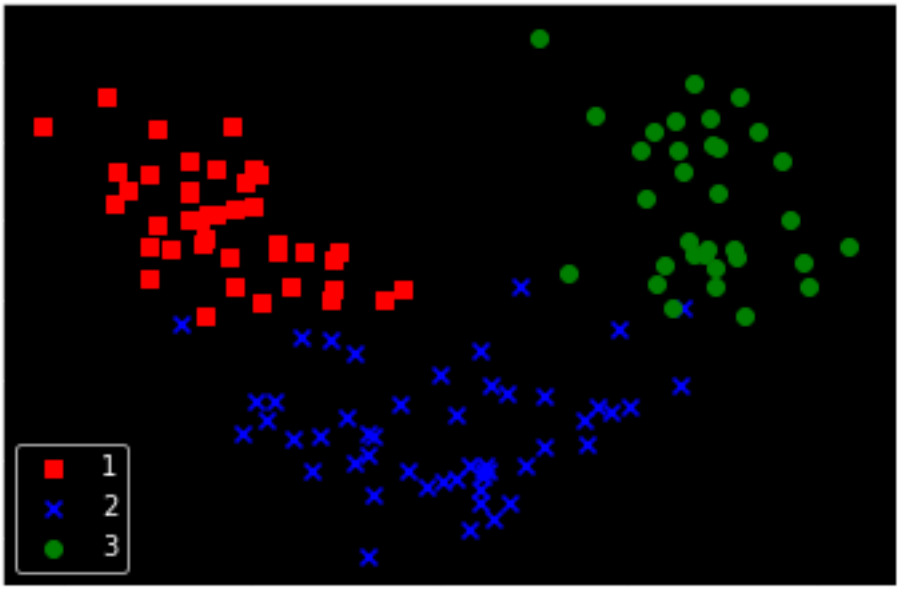
```
Matrix W:
 [[-0.13724218  0.50303478]
 [ 0.24724326  0.16487119]
 [-0.02545159  0.24456476]
 [ 0.20694508 -0.11352904]
 [-0.15436582  0.28974518]]
```

```
 [-0.39376952  0.05080104]
 [-0.41735106 -0.02287338]
 [ 0.30572896  0.09048885]
 [-0.30668347  0.00835233]
 [ 0.07554066  0.54977581]
 [-0.32613263 -0.20716433]
 [-0.36861022 -0.24902536]
 [-0.29669651  0.38022942]]
```

In [59]:
```
# 13 dimensions-->2 dimensions
x_train_pca=x_train_std.dot(w)
```

In [60]:
```
# show figure
colors=['r','b','g']
markers=['s','x','o']
for l,c,m in zip(np.unique(y_train),colors,markers):
    plt.scatter(x_train_pca[y_train==l,0],x_train_pca[y_train==l,1],c=c,label=l,marker=m)

plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend(loc='lower left')
plt.tight_layout
plt.show()
```



2 Sklearn での LogisticRegression を用いた実装

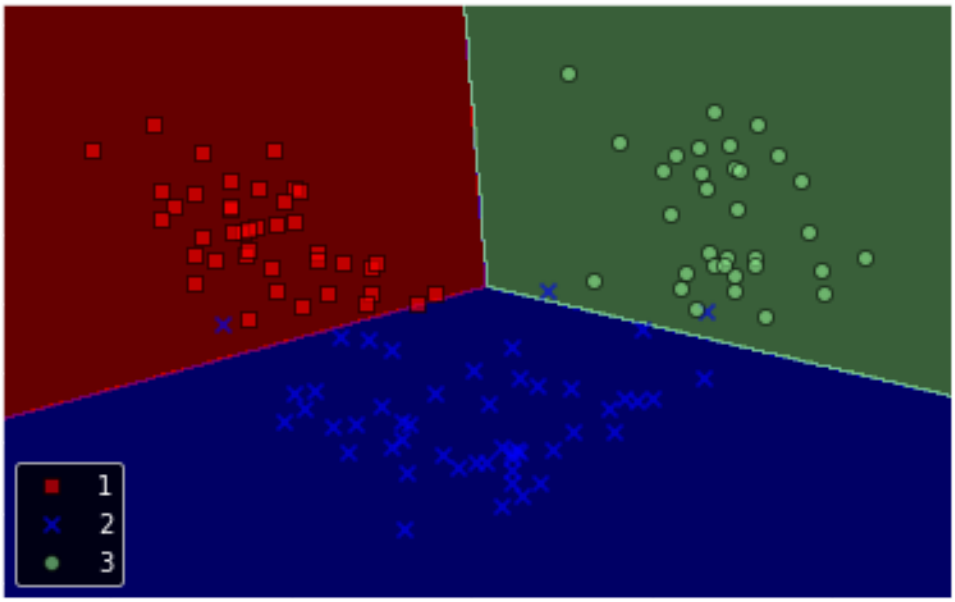
In [61]:
```
# sklearn implementation
pca=PCA(n_components=2)
lr=LogisticRegression()
x_train_pca_sk=pca.fit_transform(x_train_std)
x_test_pca_sk=pca.transform(x_test_std)
lr.fit(x_train_pca_sk,y_train)
```

```
C:\Users\taiki\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\linear_model\logisti
  FutureWarning)
C:\Users\taiki\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\linear_model\logisti
  "this warning.", FutureWarning)
```

Out[61]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

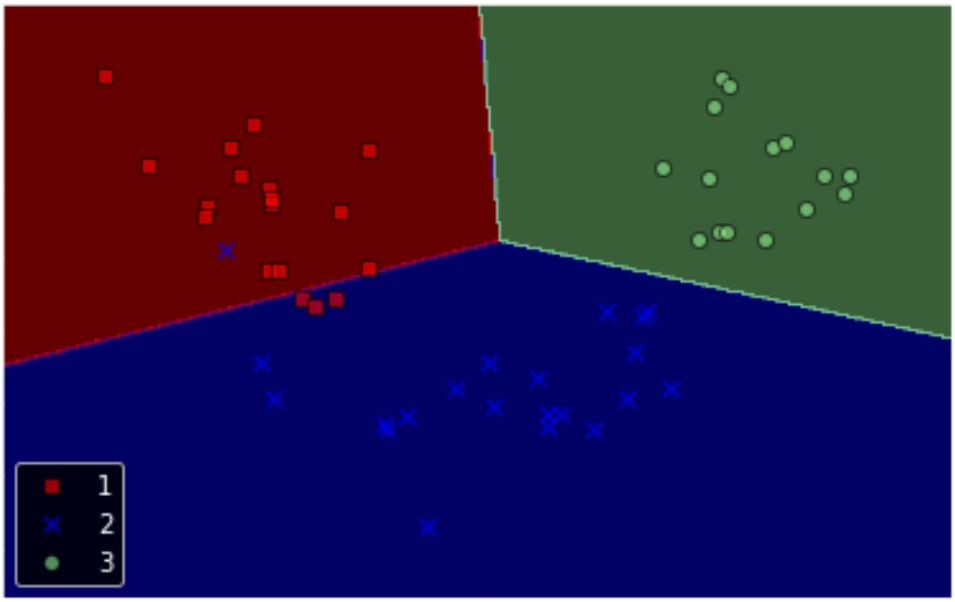
In [62]:
```
# show figure
pdr(x_train_pca_sk,y_train,classifier=lr)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(loc='lower left')
plt.tight_layout()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
```



In [63]:
```
# fit test-->plot
pdr(x_test_pca_sk,y_test,classifier=lr)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(loc='lower left')
plt.tight_layout()
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
```



3　Sklearn を用いた分散説明率の把握

In [64]:
```
# explained variance ratio
pca=PCA(n_components=None)
x_train_pca_sk=pca.fit_transform(x_train_std)
pca.explained_variance_ratio_
```

Out[64]:
```
array([0.36951469, 0.18434927, 0.11815159, 0.07334252, 0.06422108,
       0.05051724, 0.03954654, 0.02643918, 0.02389319, 0.01629614,
       0.01380021, 0.01172226, 0.00820609])
```

In [65]: