KPCAimplementation

2019 年 3 月 5 日

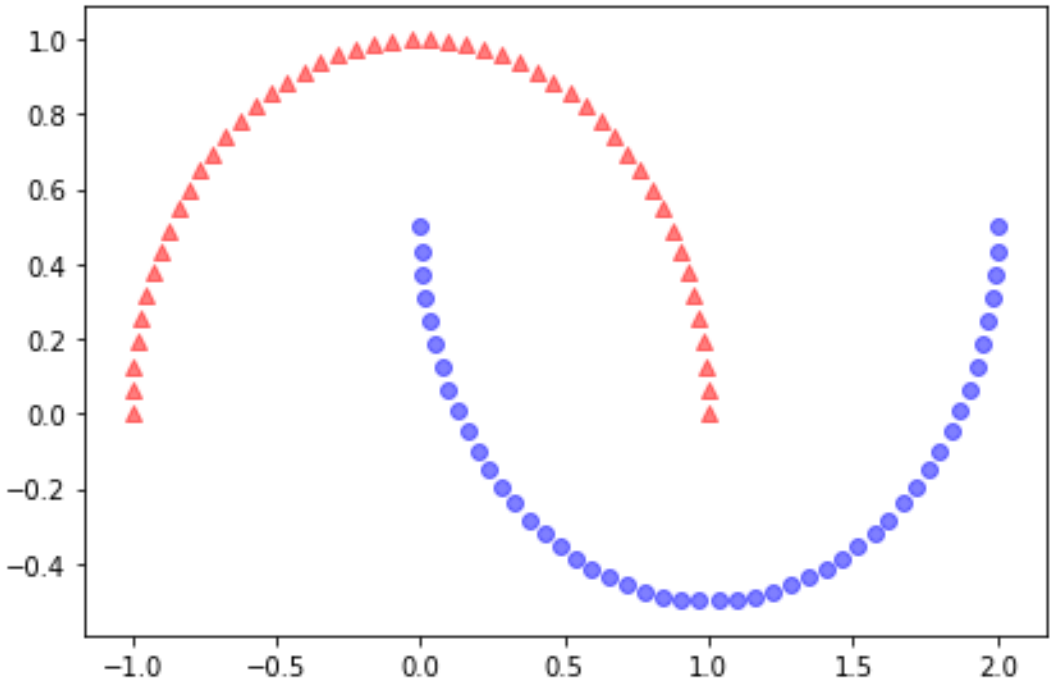
1 KPCA 実装

In [37]:
```
from scipy.spatial.distance import pdist,squareform
from scipy import exp
from scipy.linalg import eigh
import numpy as np
from sklearn.datasets import make_moons
from sklearn.datasets import make_circles
from sklearn.decomposition import PCA
from matplotlib.ticker import FormatStrFormatter
import matplotlib.pyplot as plt
from sklearn.decomposition import KernelPCA
```

In [38]:
```
# difine Kernel PCA
def rbf_kernel_pca(x,gamma,n_components):
    # difine Kernel matrix
    sq_dists=pdist(x,'sqeuclidean')
    mat_sq_dists=squareform(sq_dists)
    K=exp(-gamma*mat_sq_dists)
    # centralize Kernal matrix
    N=K.shape[0]
    one_n=np.ones((N,N))/N
    K=K-one_n.dot(K)-K.dot(one_n)+one_n.dot(K).dot(one_n)
    # calculate eigen fancots & sort into descending order
    eigvals,eigvecs=eigh(K)
    eigvals,eigvecs=eigvals[::-1],eigvecs[:,::-1]
    # select k eigen factors
    alphas=np.column_stack((eigvecs[:,i] for i in range(n_components)))
    # select eigen values
    lambdas=[eigvals[i] for i in range(n_components)]

    return alphas, lambdas
```

1.1 例 1　半月形データの射影　KPCA vs PCA

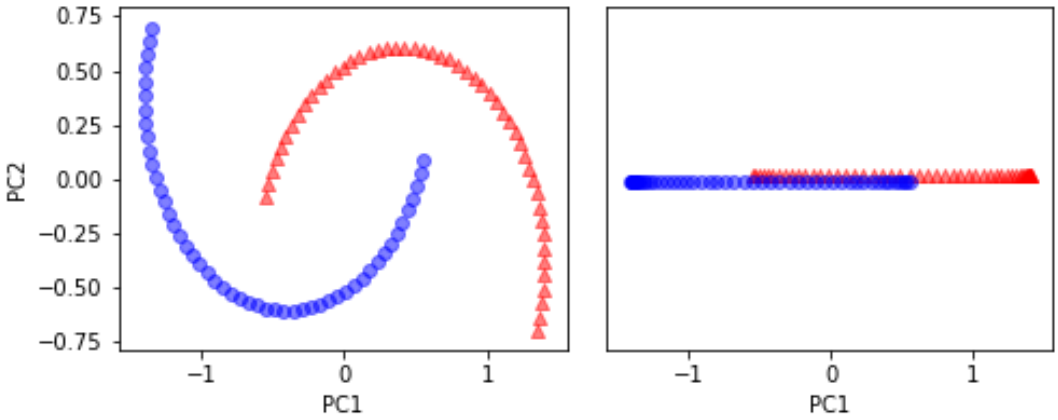
In [39]:
```
# ex1) moon shape data
x,y=make_moons(n_samples=100,random_state=123)
fig, ax=plt.subplots()
fig.set_facecolor('white')
ax.scatter(x[y==0,0],x[y==0,1],color='red',marker='^',alpha=0.5)
ax.scatter(x[y==1,0],x[y==1,1],color='blue',marker='o',alpha=0.5)
plt.tight_layout()
plt.show()
```



1.1.1 PCA ではどうなるか

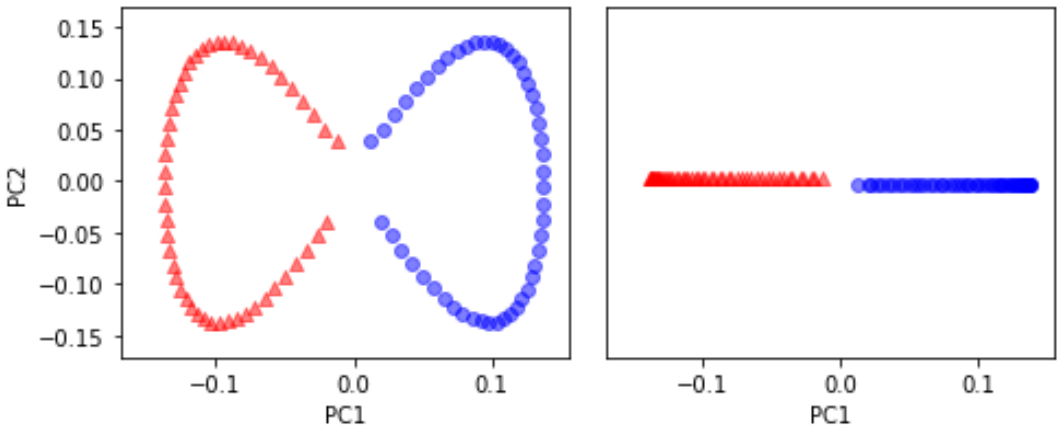
In [40]:
```
# how about PCA?
scikit_pca=PCA(n_components=2)
x_spca=scikit_pca.fit_transform(x)
fig, ax=plt.subplots(nrows=1,ncols=2,figsize=(7,3))
fig.set_facecolor('white')
ax[0].scatter(x_spca[y==0,0],x_spca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(x_spca[y==1,0],x_spca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(x_spca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0.5)
ax[1].scatter(x_spca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=0.5)
ax[0].set_xlabel('PC1')
ax[0].set_ylabel('PC2')
```

```
ax[1].set_ylim([-1,1])
ax[1].set_yticks([])
ax[1].set_xlabel('PC1')
plt.tight_layout()
plt.show()
```



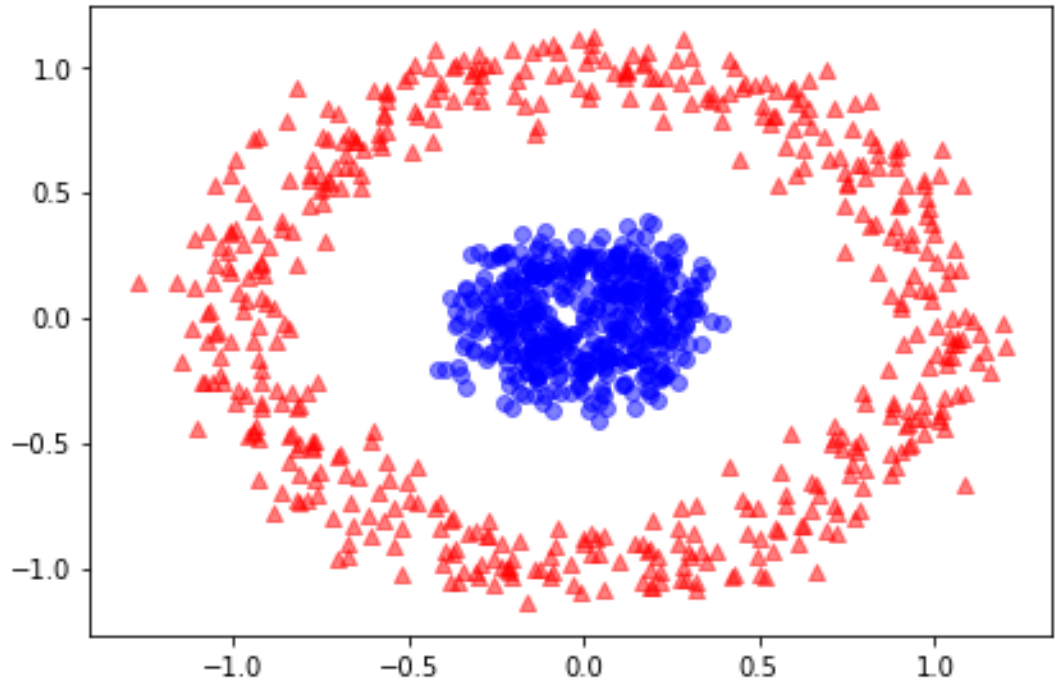
1.1.2 KPCA ではどうなるか

In [41]:
```
# How about Kernel PCA?
x_kpca,lambdas=rbf_kernel_pca(x,gamma=15,n_components=2)
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(7,3))
fig.set_facecolor('white')
ax[0].scatter(x_kpca[y==0,0],x_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(x_kpca[y==1,0],x_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(x_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0.5)
ax[1].scatter(x_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=0.5)
ax[0].set_xlabel('PC1')
ax[0].set_ylabel('PC2')
ax[1].set_ylim([-1,1])
ax[1].set_yticks([])
ax[1].set_xlabel('PC1')
plt.tight_layout()
plt.show()
```



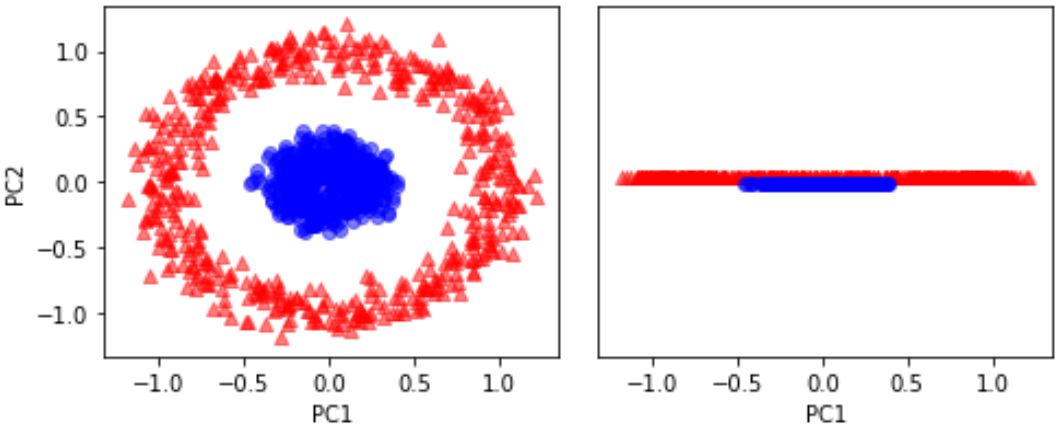
1.2 例２ 同心円状データの射影 KPCA vs PCA

```
In [42]: # ex2) circles
         x,y=make_circles(n_samples=1000,random_state=123,noise=0.1,factor=0.2)
         fig, ax=plt.subplots()
         fig.set_facecolor('white')
         ax.scatter(x[y==0,0],x[y==0,1],color='red',marker='^',alpha=0.5)
         ax.scatter(x[y==1,0],x[y==1,1],color='blue',marker='o',alpha=0.5)
         plt.tight_layout()
         plt.show()
```



1.2.1 PCA ではどうなるか

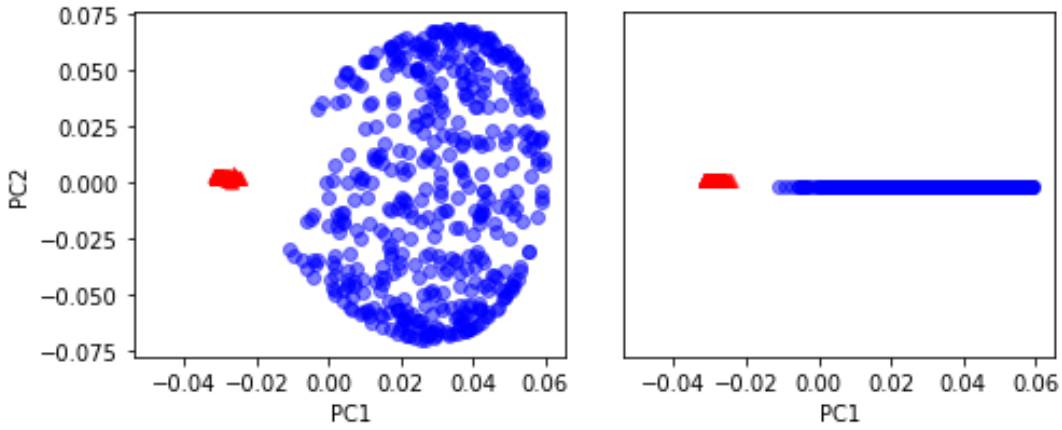
In [43]:
```
# how about PCA?
scikit_pca=PCA(n_components=2)
x_spca=scikit_pca.fit_transform(x)
fig, ax=plt.subplots(nrows=1,ncols=2,figsize=(7,3))
fig.set_facecolor('white')
ax[0].scatter(x_spca[y==0,0],x_spca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(x_spca[y==1,0],x_spca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(x_spca[y==0,0],np.zeros((500,1))+0.02,color='red',marker='^',alpha=0.5)
ax[1].scatter(x_spca[y==1,0],np.zeros((500,1))-0.02,color='blue',marker='o',alpha=0.5)
ax[0].set_xlabel('PC1')
ax[0].set_ylabel('PC2')
ax[1].set_ylim([-1,1])
ax[1].set_yticks([])
ax[1].set_xlabel('PC1')
plt.tight_layout()
plt.show()
```



1.2.2 KPCA ではどうなるか

In [44]:
```
# How about Kernel PCA?
x_kpca,lambdas=rbf_kernel_pca(x,gamma=15,n_components=2)
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(7,3))
fig.set_facecolor('white')
ax[0].scatter(x_kpca[y==0,0],x_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(x_kpca[y==1,0],x_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(x_kpca[y==0,0],np.zeros((500,1))+0.02,color='red',marker='^',alpha=0.5)
ax[1].scatter(x_kpca[y==1,0],np.zeros((500,1))-0.02,color='blue',marker='o',alpha=0.5)
ax[0].set_xlabel('PC1')
ax[0].set_ylabel('PC2')
```

```
ax[1].set_ylim([-1,1])
ax[1].set_yticks([])
ax[1].set_xlabel('PC1')
plt.tight_layout()
plt.show()
```

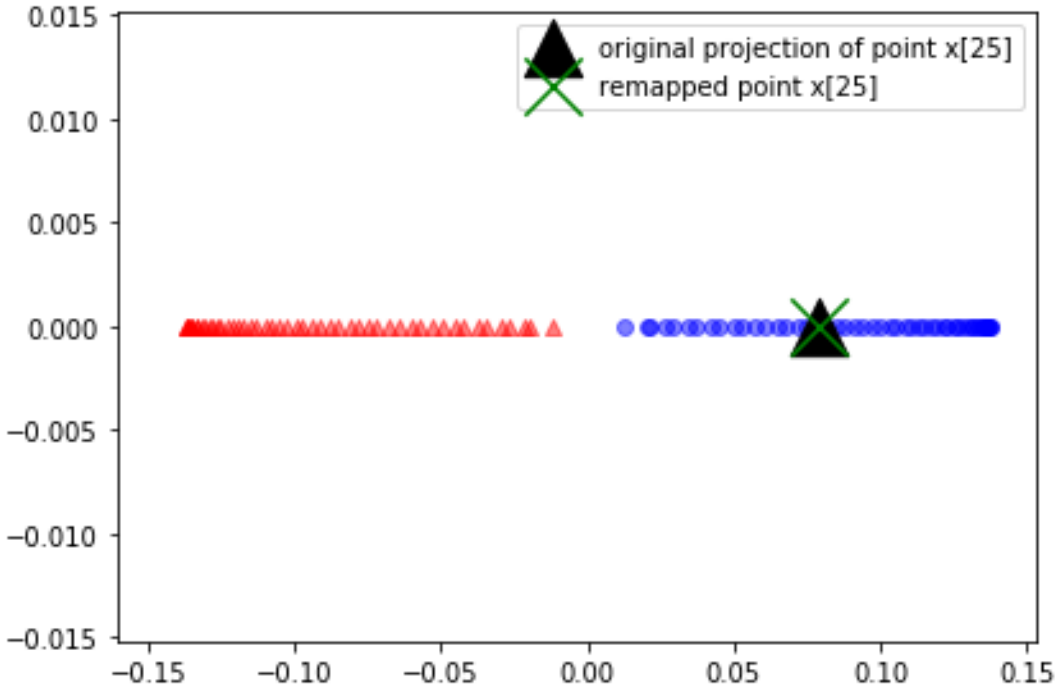


2 KPCA での New data に対する扱い方

トレーニングデータとのカーネル類似関数と固有値による正規化

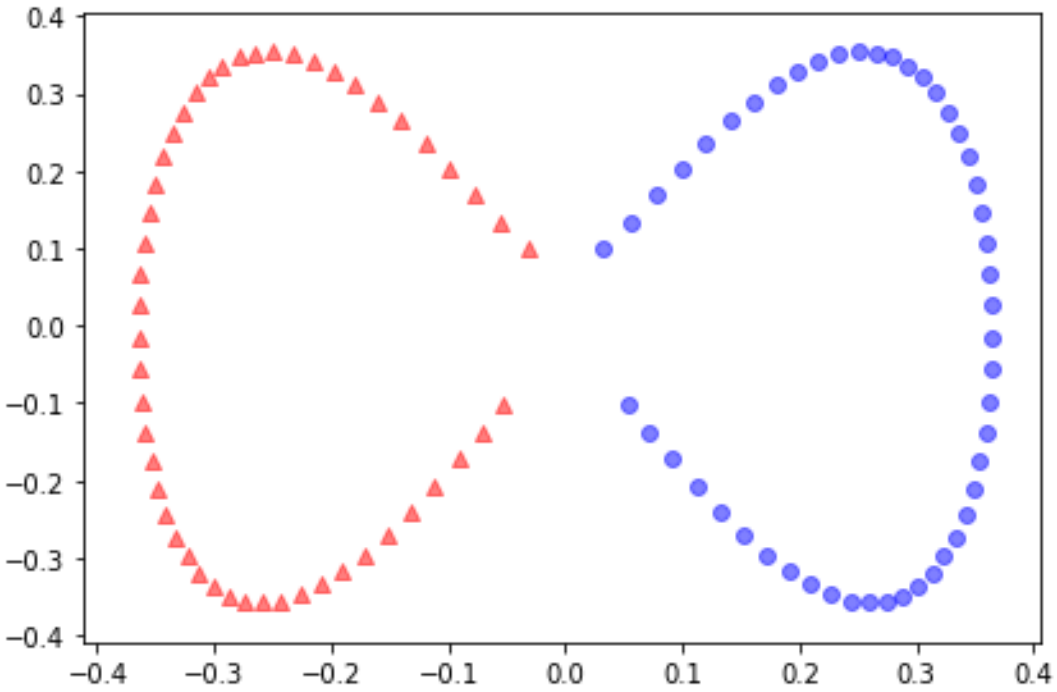
```
In [45]: # How to deal with New data? --> caluculate K between training and new ones
         x,y=make_moons(n_samples=100,random_state=123)
         alphas,lambdas=rbf_kernel_pca(x,gamma=15,n_components=1)
         x_new=x[25]
         x_proj=alphas[25]

         def project_x(x_new,x,gamma,alphas,lambdas):
             pair_dist=np.array([np.sum((x_new-row)**2)for row in x])
             k=np.exp(-gamma*pair_dist)
             return k.dot(alphas/lambdas)

         x_reproj=project_x(x_new,x,gamma=15,alphas=alphas,lambdas=lambdas)
         fig, ax=plt.subplots()
         fig.set_facecolor('white')
         ax.scatter(alphas[y==0,0],np.zeros((50)),color='red',marker='^',alpha=0.5)
         ax.scatter(alphas[y==1,0],np.zeros((50)),color='blue',marker='o',alpha=0.5)
         ax.scatter(x_proj,0,color='black',label='original projection of point x[25]',marker='^',s=5
         ax.scatter(x_reproj,0,color='green',label='remapped point x[25]',marker='x',s=500)
         plt.legend(scatterpoints=1)
         plt.tight_layout()
```

```
plt.show()
```



3 Sklearn での実装

```
In [46]: # Sklearn implementation
         x,y=make_moons(n_samples=100,random_state=123)
         scikit_kpca=KernelPCA(n_components=2,kernel='rbf',gamma=15)
         x_skernpca=scikit_kpca.fit_transform(x)
         fig, ax=plt.subplots()
         fig.set_facecolor('white')
         ax.scatter(x_skernpca[y==0,0],x_skernpca[y==0,1],color='red',marker='^',alpha=0.5)
         ax.scatter(x_skernpca[y==1,0],x_skernpca[y==1,1],color='blue',marker='o',alpha=0.5)
         plt.tight_layout()
         plt.show()
```



In [47]: