demo_kernel_pca

March 4, 2021

Luis Antonio Ortega Andrés
Antonio Coín Castro

1 Kernel PCA

Adapted by alberto.suarez@uam.es from

Kernel PCA (sklearn)
Authors: Mathieu Blondel, Andreas Mueller

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import HTML

from sklearn import datasets
from sklearn.decomposition import PCA, KernelPCA

import kernel_machine_learning as kpca

%load_ext autoreload
%autoreload 2

seed = 0
np.random.seed(seed)  # for reproducible results
```

1.1 PCA, KPCA and comparison with Sklearn

[2]:
```
# Input data
X, y = datasets.make_moons(n_samples=400,
                           noise=.05,
                           random_state=seed)
```

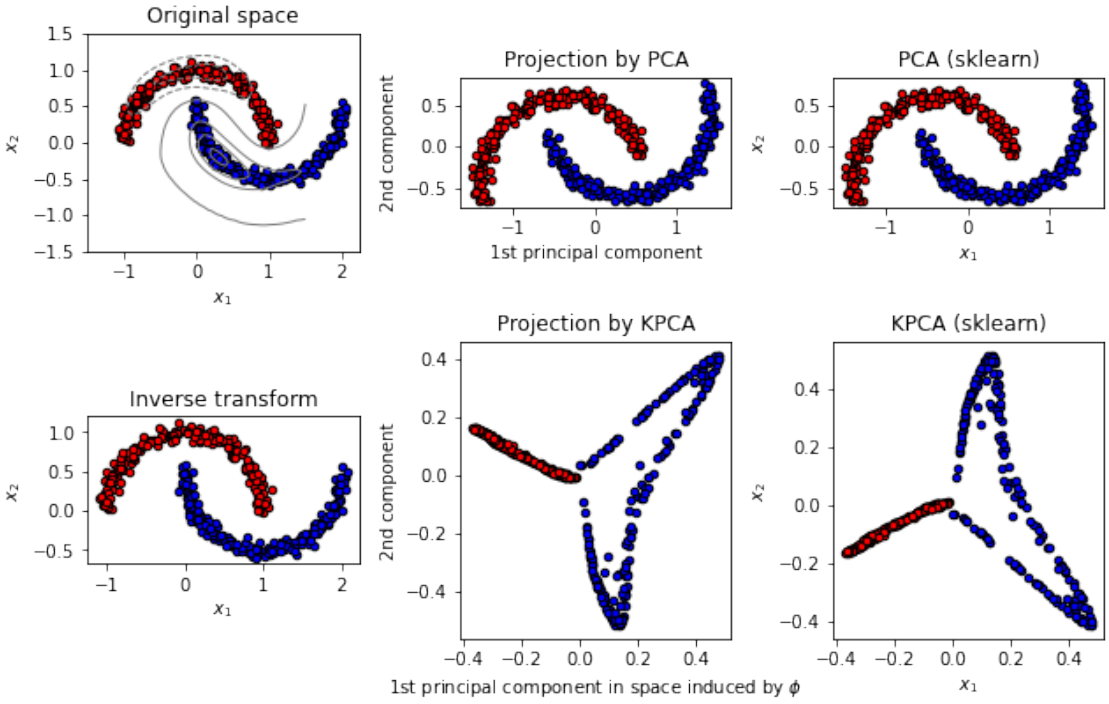
[3]:
```
# Principal componets

# PCA (linear)
X_pca, eigenvals_pca, eigenvecs_pca = \
    kpca.kernel_pca(X, X, kpca.linear_kernel)
```

```
# Kernel PCA
A = 1.0
gamma = 20.0
L = np.sqrt(0.5/gamma)

def rbf_kernel(X, X_prime):
    return kpca.rbf_kernel(X, X_prime, A, L)


X_kpca, eigenvals_kpca, eigenvecs_kpca = \
    kpca.kernel_pca(X, X, rbf_kernel)

# PCA (sklearn)
pca = PCA()
X_pca_sk = pca.fit_transform(X)

# Kernel PCA (sklearn)
kernel_pca = KernelPCA(kernel='rbf',
                       fit_inverse_transform=True,
                       gamma=gamma)
X_kpca_sk = kernel_pca.fit_transform(X)
X_back = kernel_pca.inverse_transform(X_kpca_sk)
```

[4]:
```
# Plot results

plt.figure(figsize=(9, 6))
plt.subplot(2, 3, 1, aspect='equal')
plt.title('Original space')
reds = y == 0
blues = y == 1

# Original dataset
plt.scatter(X[reds, 0], X[reds, 1], c='red',
            s=20, edgecolor='k')
plt.scatter(X[blues, 0], X[blues, 1], c='blue',
            s=20, edgecolor='k')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')

# Projection on the first principal component (in the phi space)
X1, X2 = np.meshgrid(np.linspace(-1.5, 1.5, 50),
                     np.linspace(-1.5, 1.5, 50))
X_grid = np.array([np.ravel(X1), np.ravel(X2)]).T
Z_grid = kernel_pca.transform(X_grid)[:, 0].reshape(X1.shape)
plt.contour(X1, X2, Z_grid, colors='grey', linewidths=1, origin='lower')
```

```
# Inverse transform
plt.subplot(2, 3, 4, aspect='equal')
plt.scatter(X_back[reds, 0], X_back[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X_back[blues, 0], X_back[blues, 1], c="blue",
            s=20, edgecolor='k')
plt.title("Inverse transform")
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")

# PCA and KPCA projections
plt.subplot(2, 3, 2, aspect='equal')
plt.scatter(X_pca[reds, 0], X_pca[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X_pca[blues, 0], X_pca[blues, 1], c="blue",
            s=20, edgecolor='k')
plt.title("Projection by PCA")
plt.xlabel("1st principal component")
plt.ylabel("2nd component")

plt.subplot(2, 3, 5, aspect='equal')
plt.scatter(X_kpca[reds, 0], X_kpca[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X_kpca[blues, 0], X_kpca[blues, 1], c="blue",
            s=20, edgecolor='k')
plt.title("Projection by KPCA")
plt.xlabel(r"1st principal component in space induced by $\phi$")
plt.ylabel("2nd component")

plt.subplot(2, 3, 3, aspect='equal')
plt.scatter(X_pca_sk[reds, 0], X_pca_sk[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X_pca_sk[blues, 0], X_pca_sk[blues, 1], c="blue",
            s=20, edgecolor='k')
plt.title("PCA (sklearn)")
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")

plt.subplot(2, 3, 6, aspect='equal')
plt.scatter(X_kpca_sk[reds, 0], X_kpca_sk[reds, 1], c="red",
            s=20, edgecolor='k')
plt.scatter(X_kpca_sk[blues, 0], X_kpca_sk[blues, 1], c="blue",
            s=20, edgecolor='k')
plt.title("KPCA (sklearn)")
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
```

```
plt.tight_layout()
plt.show()
```



1.2 KPCA animation

We make an animation that shows the evolution of the projections onto the first two KPCA components for $0.002 \leq \gamma \leq 20000.0$.

[5]:
```
animation = kpca.AnimationKPCA(n_frames=100,
                               xlims=(-1, 1),
                               ylims=(-1, 1)).animate(X, X, y)
plt.close()
HTML(animation.to_jshtml())
```

[5]: <IPython.core.display.HTML object>