Lab13.2

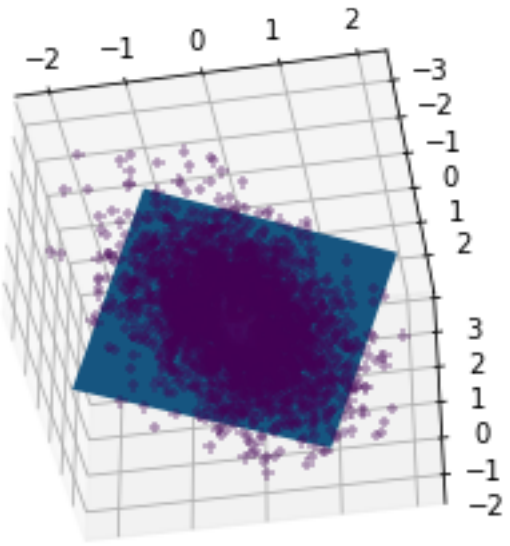
October 4, 2022

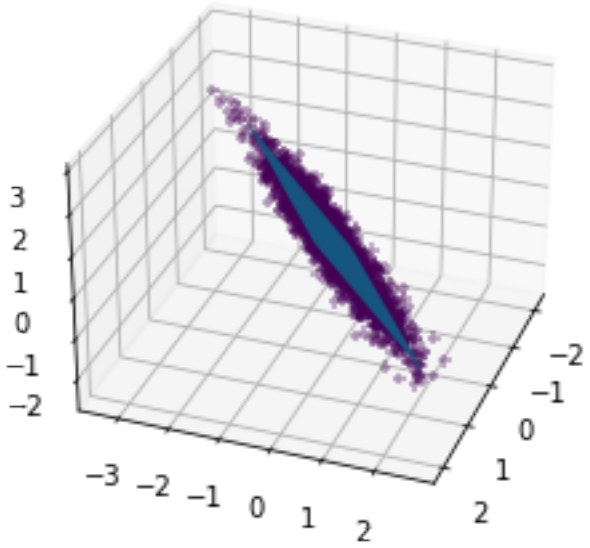
1 Principal Component Analysis (PCA)

Here we see that PCA ignores components with small magnitude and instead fits the part of the data which is best described.

[ ]:
```
# Authors: Gael Varoquaux
#          Jaques Grobler
#          Kevin Hughes
# License: BSD 3 clause

from sklearn.decomposition import PCA

from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats


# ###############################################################################
# Create the data
e = np.exp(1)
np.random.seed(4)
def pdf(x):
    return 0.5 * (stats.norm(scale=0.25 / e).pdf(x)
                  + stats.norm(scale=4 / e).pdf(x))
y = np.random.normal(scale=0.5, size=(30000))
x = np.random.normal(scale=0.5, size=(30000))
z = np.random.normal(scale=0.3, size=len(x))

density = pdf(x) * pdf(y)
pdf_z = pdf(5 * z)
density *= pdf_z

a = x + y
b = 2 * y
c = a - b + z

norm = np.sqrt(a.var() + b.var())
```

```
a /= norm
b /= norm


# ############################################################################
# Plot the figures
def plot_figs(fig_num, elev, azim):
    fig = plt.figure(fig_num, figsize=(4, 3))
    plt.clf()
    ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=elev, azim=azim)

    ax.scatter(a[::10], b[::10], c[::10], c=density[::10], marker='+', alpha=.4)
    Y = np.c_[a, b, c]

    # Using SciPy's SVD, this would be:
    # _, pca_score, Vt = scipy.linalg.svd(Y, full_matrices=False)

    pca = PCA(n_components=3)
    pca.fit(Y)
    V = pca.components_.T

    x_pca_axis, y_pca_axis, z_pca_axis = 3 * V
    x_pca_plane = np.r_[x_pca_axis[:2], - x_pca_axis[1::-1]] #Only plot the␣
 ↪first two components
    y_pca_plane = np.r_[y_pca_axis[:2], - y_pca_axis[1::-1]] #Only plot the␣
 ↪first two components
    z_pca_plane = np.r_[z_pca_axis[:2], - z_pca_axis[1::-1]] #Only plot the␣
 ↪first two components
    x_pca_plane.shape = (2, 2)
    y_pca_plane.shape = (2, 2)
    z_pca_plane.shape = (2, 2)
    ax.plot_surface(x_pca_plane, y_pca_plane, z_pca_plane)
#     ax.w_xaxis.set_ticklabels([])
#     ax.w_yaxis.set_ticklabels([])
#     ax.w_zaxis.set_ticklabels([])


elev = -40
azim = -80
plot_figs(1, elev, azim)

elev = 30
azim = 20
plot_figs(2, elev, azim)

plt.show()
```





[ ]:
```
X = np.c_[a, b, c] # data from earlier
pca = PCA(n_components=3)
pca.fit(X)
V = pca.components_

print(f"principal components: \n {V}")
```

```
print(f"explained variance: {pca.explained_variance_}")

plt.figure()
fig, ax = plt.subplots(1, 3, figsize=(12, 4))

fignum = 0
for i in range(3):
    for j in range(i + 1, 3):
        ax[fignum].axis('equal')
        ax[fignum].scatter((X @ V[i])[::25], (X @ V[j])[::25])
        ax[fignum].set_title("projected data")
        ax[fignum].set_xlabel(f"component {i}")
        ax[fignum].set_ylabel(f"component {j}")

        fignum += 1

fig.tight_layout()
```

```
principal components:
 [[-0.31995789 -0.75414612  0.57348983]
 [ 0.68703979  0.23211827  0.68854734]
 [-0.65238277  0.61431648  0.44386031]]
explained variance: [1.11390257 0.44961862 0.01995951]
```

```
<Figure size 432x288 with 0 Axes>
```