METHODS IN BIOINFORMATICS

Exercise 1: Principal Component Analysis

Kyriakis Dimitris

27/3/2017

PCA

Principal Component Analysis (PCA) is a multivariate statistical technique using sophisticated underlying mathematical principles to transform a number of possibly correlated variables into a smaller number of variables called principal components. Seeks a space of lower dimensionality, known as the principal subspace. There are two commonly used definitions of PCA that lead to the same algorithm. The first one is that the orthogonal projection of the data onto a lower dimension, in order to maximize the variance of the projected data[1]. Equivalently, it can be defined as the linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections[2].

PPCA

PCA has some limitations because it is not based on a statistical model, so can't extend the aim of PCA[5]. PCA often fails to reveal underlying groups in the data, therefore providing not readable data structure[3]. Other limitations include the inability of PCA to deal with missing data appropriately[4]. In other words, if we perform PCA on some data and then ask how well new data are fit by the model, the only criterion used is the squared distance of the new data from their projections into the principal subspace[7].

Advantages of Probabilistic PCA:

- Constrained form of the Gaussian distribution
- Use of EM algorithm -> computationally efficient
- Deal with missing data
- Comparison with other probabilistic models
- Applied to classification problems
- Use mixtures of PPCA models

PPCA using EM algorithm does not require computing the sample covariance and has a complexity limited by O (knp) operations where k is the number of leading eigenvectors to be learned[7], in contrast with conventional PCA which has complexity $O(n^3)$. The EM is an iterative procedure for finding the subspace spanned by the k leading eigenvectors without explicit computation of the sample covariance[7].

Kernel –PCA

PCA will always detect all structure in a given data set, because of linearity. By using suitable nonlinear features, we can extract more information. Vapnik Chervonenkis theory tells us that often mappings our data into a higher dimensional space than the dimension of the input space provide us with greater classification power and the data could become more easily separated.

Kernel PCA is very well suited to extract interesting nonlinear structures in the data[7]. This aim to consider nonlinear de-noising based on Kernel PCA and to clarify the connection between feature space expansions and meaningful patterns in input space. Kernel PCA first maps the data into some feature space F via a (usually nonlinear) function Φ I and then performs linear PCA on the mapped data.

The most common use of Kernel is the support vector machine (SVM). The general task of pattern analysis is to find and study general types of relations (such as clusters, rankings, principal components, correlations, classifications) [6].

The kernels that we used for the analysis:

1. Gaussian

The Gaussian kernel it could be implemented using:

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \quad \text{or} \quad k(x,y) = \exp\left(-\gamma\|x-y\|^2\right)$$

The adjustable parameter sigma plays a major role in the performance of the kernel. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. For our exercise we used the second implementation.

2. Polynomial

Polynomial kernels are well suited for problems where all the training data is normalized.

k (xi, xj) = (1+ xi xj) ^{p}.

Adjustable parameters are the constant term c and the polynomial degree d.

3. Hyperbolic tangent

The Hyperbolic Tangent Kernel (Sigmoid Kernel, Multilayer Perceptron (MLP)) kernel. Also called Sigmoid Kernel because in Neural Networks field, the bipolar sigmoid function is often used as an activation function for artificial neurons.

k (xi, xj) = tanh (xixj +d)

Also, despite being only conditionally positive definite, it has been found to perform well in practice.

Theoretical

Problem 1.

Our goal was to construct a standard PCA (eigen decomposition and SVD algorithms), a PPCA (with sigma zero and non-zero) and a Kernel PCA function. For the Kernel PCA we have to construct the basis kernel Gaussian, Polynomial and Hyperbolic tangent.

- Gaussian

$$K(x_i, x_j) = exp\left(-\beta\|x_i - x_j\|^2\right)$$

- Polynomial

$$K(x_i, x_j) = (1 + x_i x_j)^p$$

- Hyperbolic tangent

$$K(x_i, x_j) = tanh(x_i x_j + \delta)$$

Perform PCA in the given dataset with all approaches. For the Kernel PCA compare and contrast the performance with the 3 kernels. Finally, change the noise in the given dataset to 3. Run again kernel PCA with the Gaussian kernel. Describe the results. Generate a dataset X which is randomly gaussian distributed with mean a 10x1 vector with 1 in every position and 10 samples. Plot the eigenvalues in descending order. Run again PCA keeping only 5 samples. Inspect the eigenvalues. Describe your results.

1. NOISE = 0.05

- **PCA**

  Running PCA and PPCA in 2 dimensions we can see that our data can't split linearly. Also in one dimension, we can't distinguish the two groups of data.

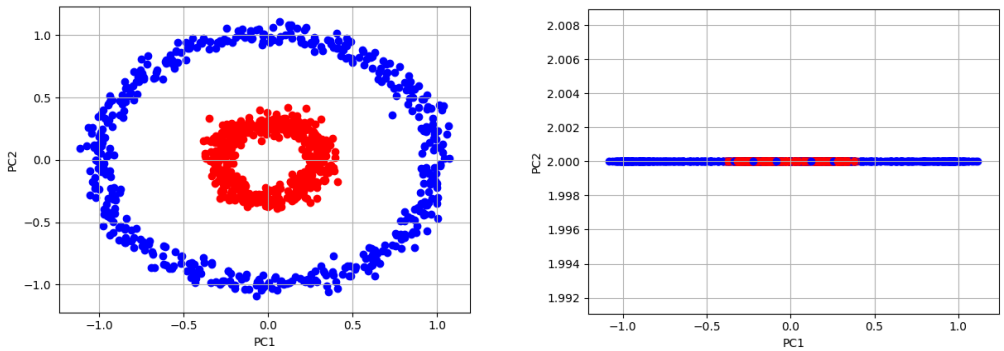


Figure 1: PCA with noise 00.5. Left: PCA in 2 Dimension. Right: PCA in one dimension

- **PPCA**

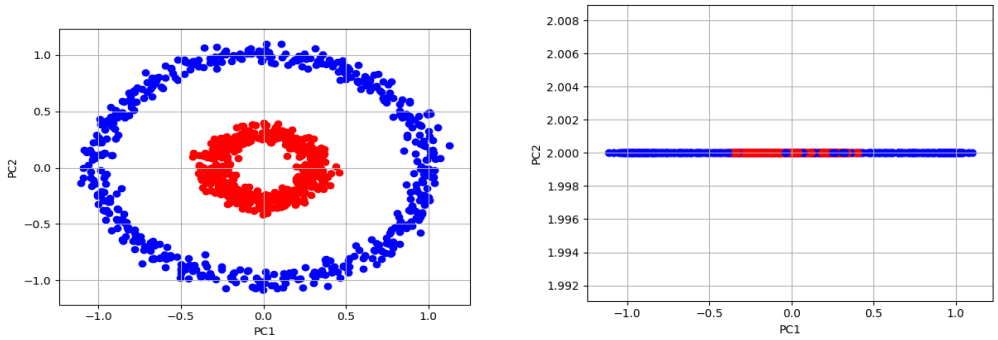


Figure 2: PPCA with noise 00.5. Left: PPCA in 2 Dimension. Right: PPCA in one dimension

- **Kernel**

On the other hand if we use Kernel PCA with Gaussian kernel, our data separated in the two expected groups. The data can't separate using the other two kernels.

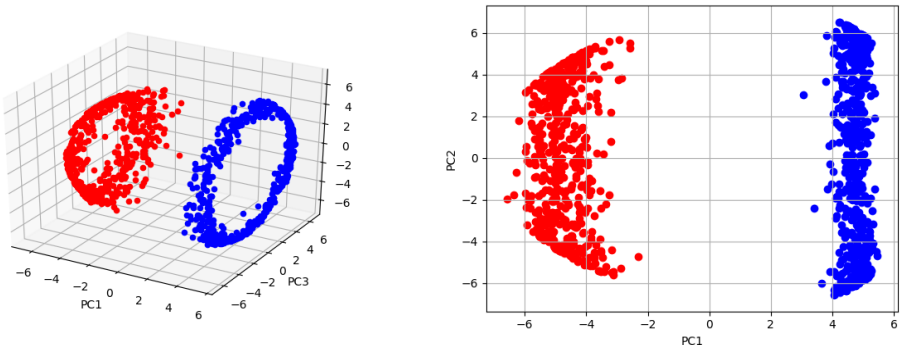


Figure 3: Gaussian 2 with noise 0.05

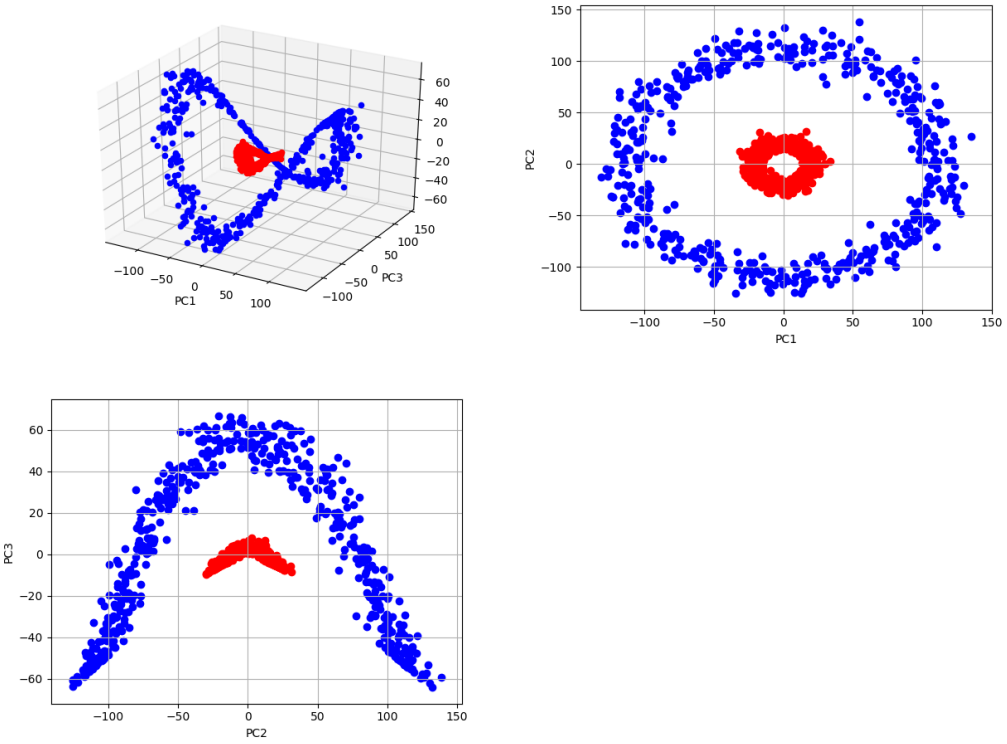
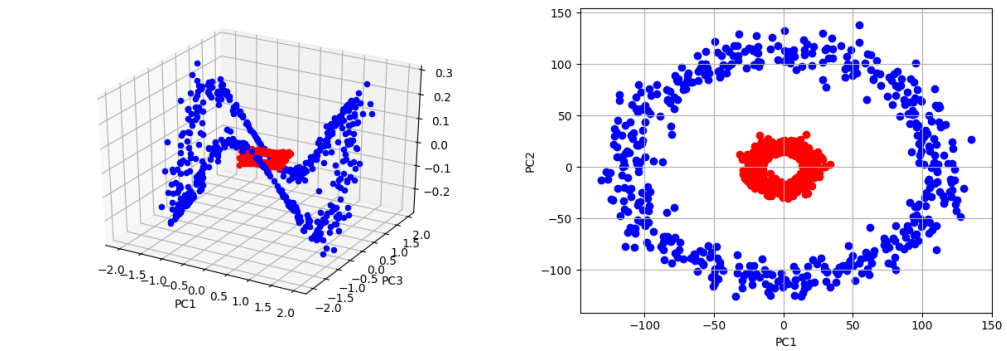


Figure 4: Polynomial p=4 noise=0.05



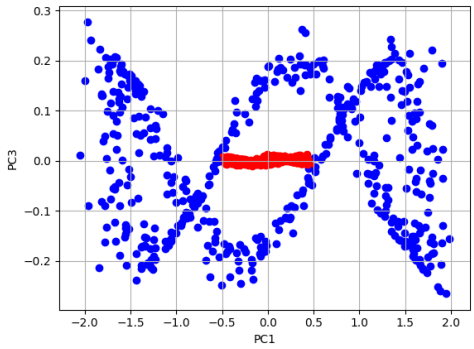


Figure 5: Hyperbolic δ=2 noise = 0.05

2. NOISE = 3

From the plots below we can see that increasing the noise to the dataset, make it more difficult to distinguish the different groups than earlier. In kernel PCA, different values were used for the parameters (g, p, d) of the different kernels (Gaussian, Polynomial, Hyperbolic) respectively. None of them seems to make the data separated.

- **PCA**

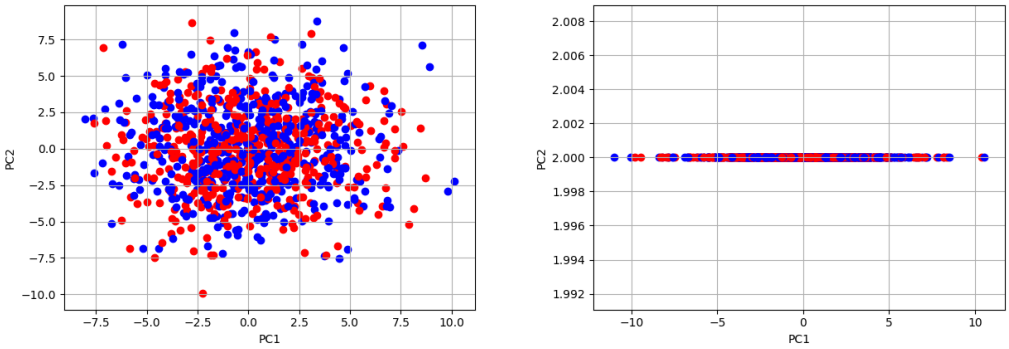


Figure 6: PCA with noise 3

- **PPCA**

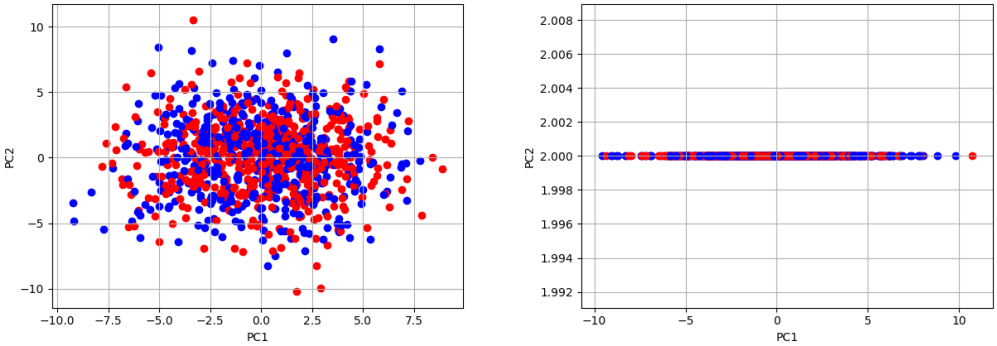


Figure 7: PPCA noise=3

- **Kernel**

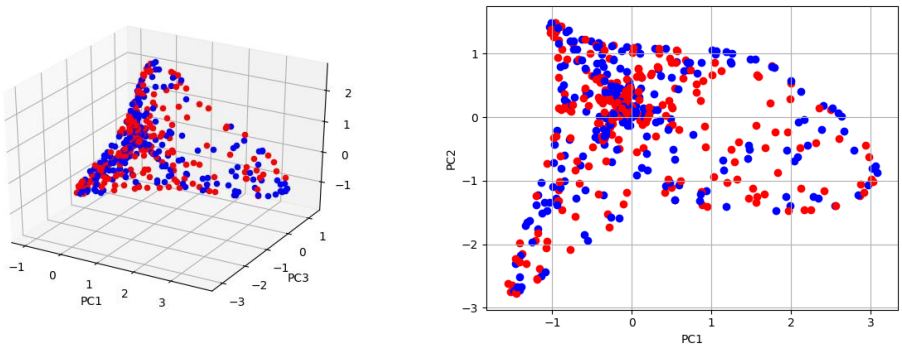


Figure 8: Gaussian g=2, noise=3

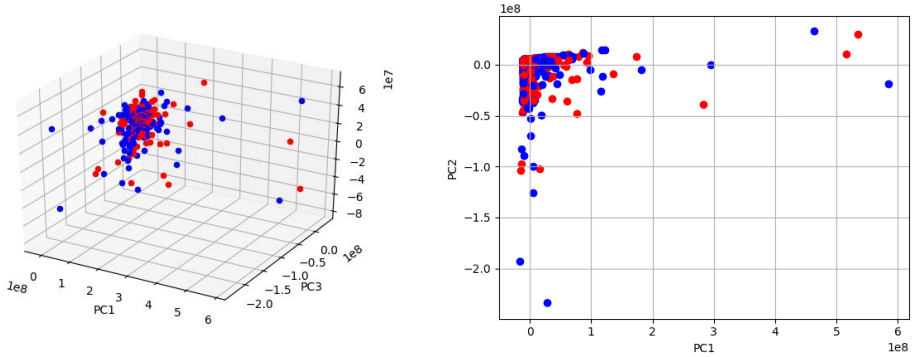


Figure 9: Polynomial p=2 with noise 3

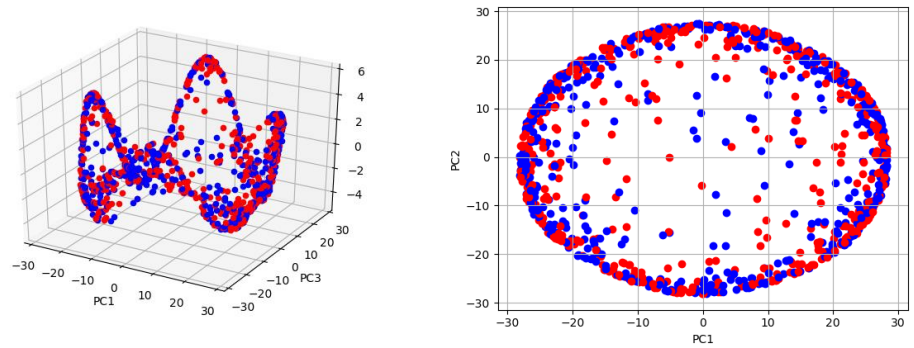


Figure 10: Hyperbolic δ=2 noise=3

Problem 2.

Generate a dataset X which is randomly gaussian distributed with mean a 10x1 vector with 1 in every position and 10 samples. Plot the eigenvalues in descending order. Run again PCA keeping only 5 samples.

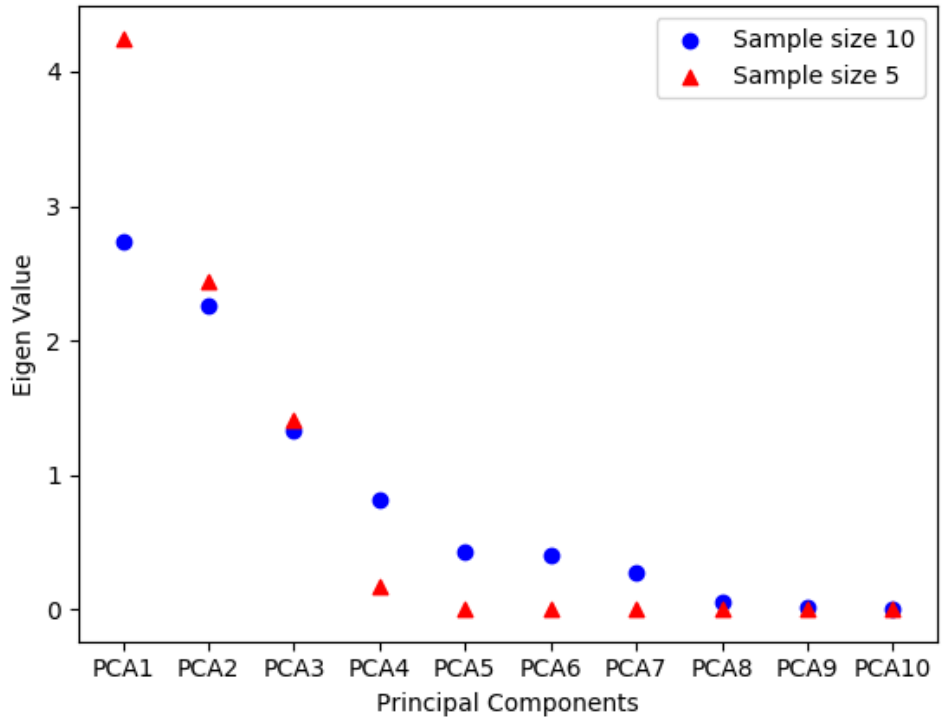


Figure 11: Eigenvalues Plot. Left: A randomly gaussian distributed dataset with mean=1, cov =np.eye and 10 samples. Right: A randomly gaussian distributed dataset with mean=1, cov =np.eye and 5 samples.

As we can see the eigenvalues from the dataset with 10 observations seems to be more informative than eigenvalues from second dataset. As the plot shows in the 5 observations data set only 5 eigenvalues are informative.

Practical

The aim of this exercise is to perform PCA in a real data set. The analysis of livers of C57BL/6J mice fed a high fat diet for up to 24 weeks has shown significant body weight gain was observed after 4 weeks. The results provide insight into the effect of high-fat diets on metabolism in the liver.
I tried to run normal PCA on the data but it crushed because the covariance matrix has large computational cost, in contrast PPCA use statistical model. So I used PPCA to reduce dimensions in order to interpret.

With random W the results seems to be the same with the already compiled PPCA from:

```
from sklearn.decomposition import PCA
```

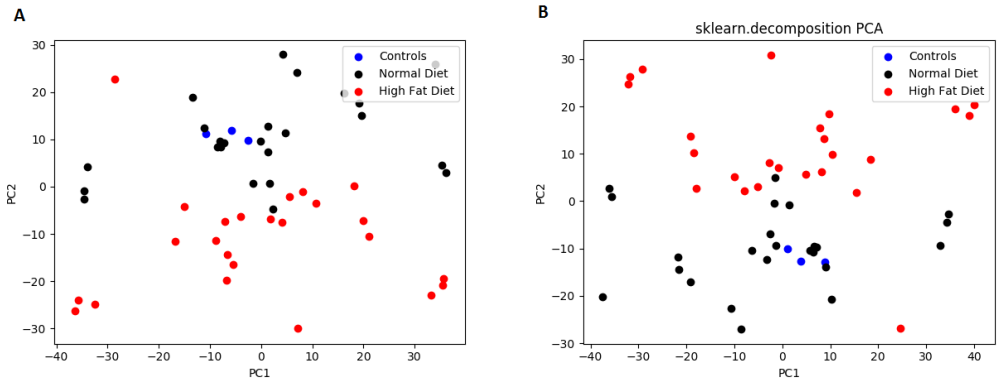


Figure 12: PPCA with 2 components. Blue dots are the controls, black dots the mice with normal diet and the red dots the mice with the high fat diet A) The results from my implemented PPCA. B) Implemented by the sklearn PPCA.

The results seems the same just rotated.

Some other results of the PPCA:

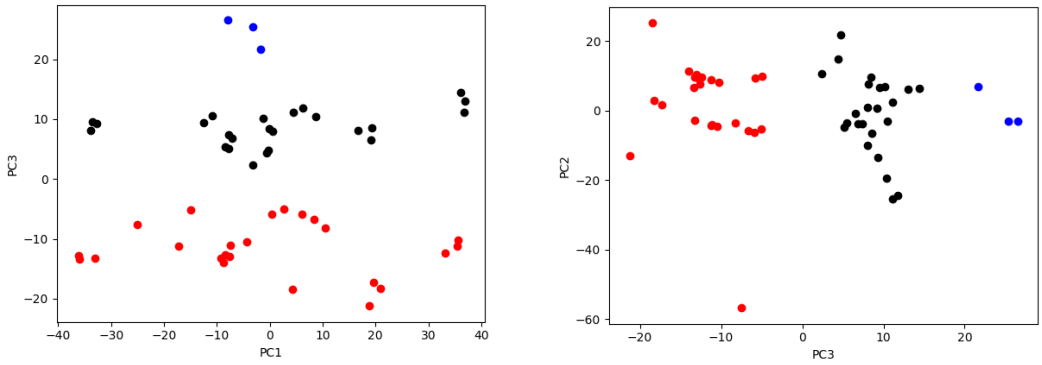
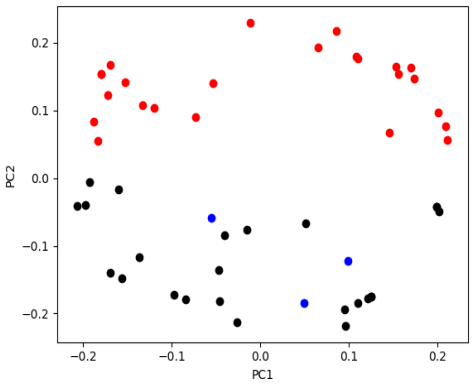


Figure 13: PPCA with 2 components. Blue dots are the controls, black dots the mice with normal diet and the red dots the mice with the high fat diet. For these two PPCA used fixed W to start the PPCA. In the right plot used the W_5.npy to start the analysis.

Finally, I run Kernel PCA with Hypebolic kernel. The data seems to be full separated between the groups. The Polynomial and Gaussian kernel didn't work for our data set.



Bibliography

1. Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24, 417–441.

2. Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science, Sixth Series 2, 559–572.
3. McLachlan GJ, Peel D. Finite Mixture Models. New York: Wiley; 2000.
4. Roweis S. EM Algorithms for PCA and SPCA. Neural Information Processing Systems. 1998;10:626–632.
5. Nyamundanda, G., Brennan, L., & Gormley, I. C. (2010). Probabilistic principal component analysis for metabolomic data. BMC Bioinformatics, 11, 571. http://doi.org/10.1186/1471-2105-11-571
6. Wikipedia contributors, "Kernel methods," Wikipedia, The Free Encyclopedia,http://en.wikipedia.org/w/index.php?title=Kernel_methods&oldid=340911970 (accessed March 3, 2010).
7. S Roweis - Advances in neural information processing systems, 1998