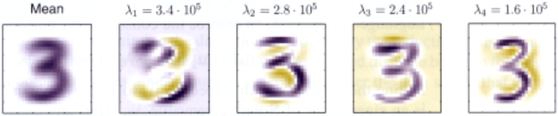
Principal Component Analysis

Principal Component Analysis (PCA)

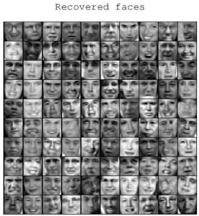
- High dimensional datasets contain correlations between the dimensions, so a portion of the data is redundant.
- Linear transformation on data that reduces data to a few dimensions.

Mean and Principal Components



Reconstruction from Principal Components

PCA Applications: Face Image Analysis

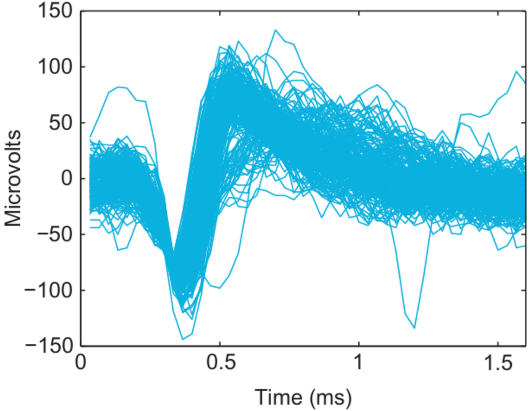


http://mikedusenberry.com/on-eigenfaces/

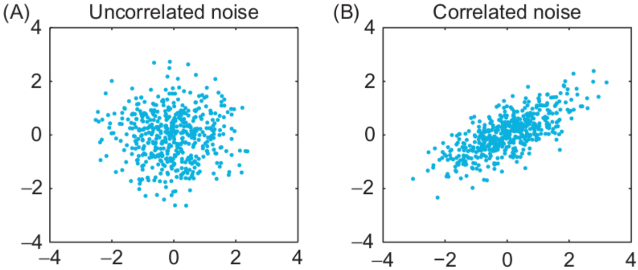
Projection of face images onto a feature space that spans the significant variations among known face images

Turk & Pentland, 1991

PCA Applications: Neuroscience



Correlations between dimensions:



`scripts/corr_data.m`

```
n=500
a(:,1) = normrnd(0,1,n,1);
a(:,2) = normrnd(0,1,n,1);
b(:,1) = normrnd(0,1,n,1);
b(:,2) = b(:,1)∗.5 + .5 ∗ normrnd(0,1,n,1);
```

Univariate Sample Mean:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

: **Univariate Sample Variance:**

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Multivariate Sample Mean:

$X = [\mathbf{x}_1, \cdots, \mathbf{x}_p]$

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$

Multivariate Sample Covariance:

$$\Sigma_{jk}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

Covariance expresses the extent to which two rvs vary together. If two variables are independent, $cov(x, y)$ tends to zero. Matlab cov function is the sample covariance

Multivariate Gaussian

$$p(X) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(X-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(X-\boldsymbol{\mu})\right)$$

Change of basis:

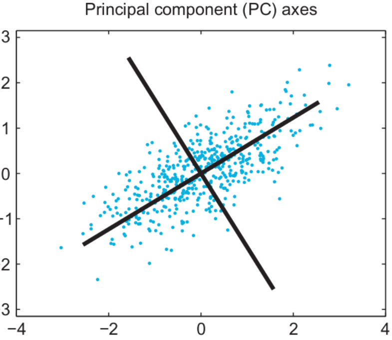
$$Y = Q(X-\mu)$$

Q are the eigenvectors of Σ

$$p(Y) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}Y^TD^{-1}Y\right)$$

In general, covariance can only be estimated.
Idea behind PCA: orthogonal linear transformation of the data using eigenvectors of the sample covariance. Note that PCA can be applied to non-Gaussian data.

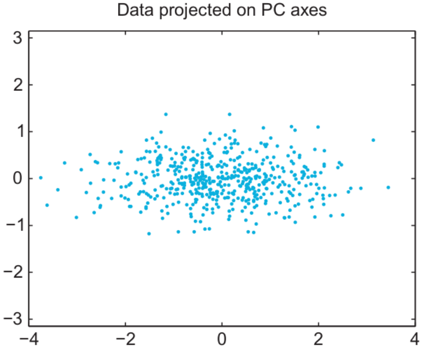
- PCA can be viewed as a coordinate transformation
- Original data is plotted on horizontal and vertical axes
- PCA rotates the axes so that the new horizontal axis lies along the direction of **maximum variation.**



- The new axes are the obtained by a change of basis consisting of the **eigenvectors** of the covariance matrix.

scripts/pca_eigen.m

```
sigma = cov(b)
[V, D] = eig(sigma)

%returns

V = 0.5387 −0.8425
  − 0.8425 −0.5387

D = 0.2048 0
          0 1.3341

plot(b(:,1),b(:,2),'b.'); hold on
plot(3*[−V(1,1) V(1,1)],3*[−V(1,2) V(1,2)],'k')
plot(3*[−V(2,1) V(2,1)],3*[−V(2,2) V(2,2)],'k')

plot(b*V,'b.');
```


Data projected on PC axes
3
2
1
0
−1
−2
−3
−4
−2
0
2
4

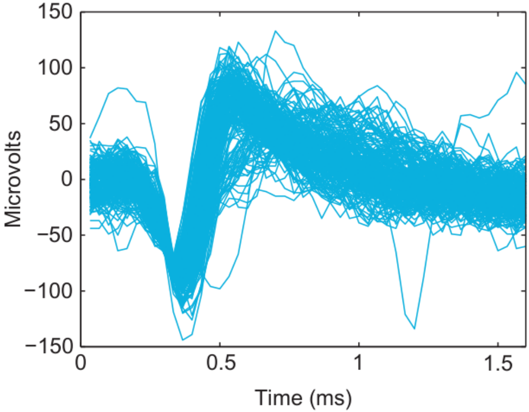
How much Variation is Captured?

- The fraction of variation captured by each PC is the ratio of its eigenvalue to the sum of all the eigenvalues:

$$\frac{\lambda_i}{\sum_j \lambda_j}$$

[coeff, score, latent] = pca(b)

- coeff are the eigenvectors
- score is the transformed data ordered from largest to smallest PC
- latent are the eigenvalues



- Extracellular recordings of spike waveforms.
- Raw data per recording has 48 points
- There seems to be two different traces. Can we tease them apart?

PCA Applications: Neuroscience

- Load data

scripts/load_spike_Waveform.m

```
load Chap19_SpikeSorting.mat
wf=session(2).wf;
plot(wf(1:1000,:)','b');
```

- Run PCA

scripts/pca_spike_Waveform.m

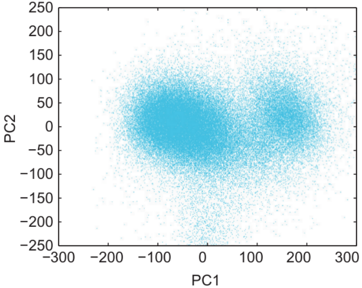
```
[ coeff ,score, latent ] = pca(double(wf(1:1000,:)))
```

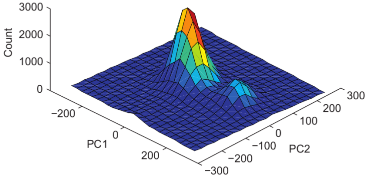
- Plot First 2 components

scripts/pca_spike_Waveform_plot.m

```
scatter(score(:,1),score(:,2))
```

PCA Applications: Neuroscience
Viewing the first two PCs

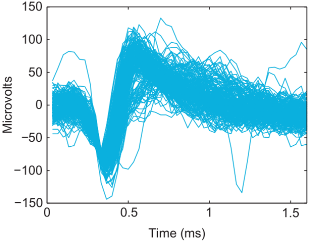


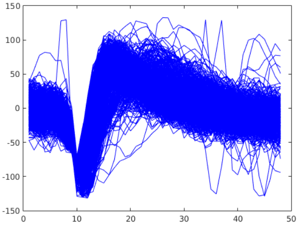


Reconstructing the data from PCs

$$\mathbf{x} = \mathbf{s}Q^\top + \boldsymbol{\mu}$$

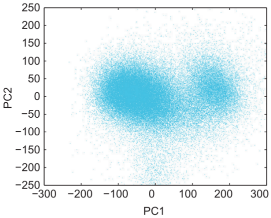
where $\boldsymbol{\mu}$ is average (here the average waveform) and $\mathbf{s}$ is the vector of PCs

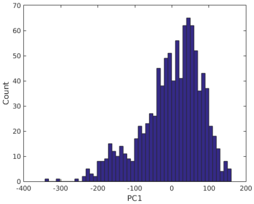




- Data compression: Each waveform is stored using 2 variables instead of 48

- Can we systematically discriminate the two groups/classes?



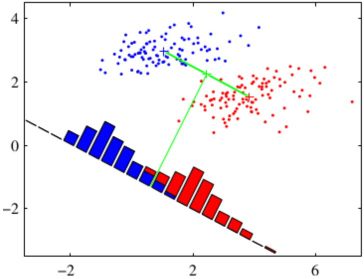


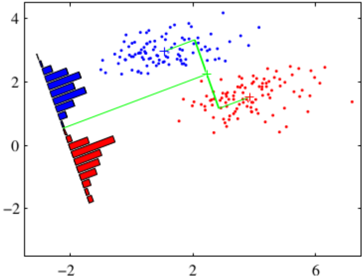
Linear Classification with Dimensionality Reduction

Projection of data to 1 dimension

$$y = \mathbf{w}^\top \mathbf{x}$$

- Place a threshold on y and classify $y \leq -w_0$ as class C_1, otherwise class C_2
- But the projection to 1 dimension throws away a lot of information.





- By adjusting $\mathbf{w}$, we can select a projection that maximizes the class separation

- N_1 points in class C_1
- N_2 points in class C_2
- Mean vectors of the two classes are:

$$\bar{\mathbf{x}}_1 = \frac{1}{N_1} \sum_{i \in C_1} \mathbf{x}_i$$

$$\bar{\mathbf{x}}_2 = \frac{1}{N_2} \sum_{i \in C_2} \mathbf{x}_i$$

- We could choose $\mathbf{w}$ such that

$$m_2 - m_1 = \mathbf{w}^\top (\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1)$$

is maximum. This doesn't work well when the data has correlated features

Solution: Maximize a function that will give a large separation between projected class means while giving a small variance within each class

Fisher Criterion

$$J = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

where s_1 and s_2 are **within class variances**

$$s_1^2 = \frac{1}{N_1} \sum_{n \in C_1} (\mathbf{w}^\top \mathbf{x}_1 - \mathbf{w}^\top \bar{\mathbf{x}}_1)^2$$

$$s_2^2 = \frac{1}{N_2} \sum_{n \in C_2} (\mathbf{w}^\top \mathbf{x}_2 - \mathbf{w}^\top \bar{\mathbf{x}}_2)^2$$

Solution that maximizes the Fisher Criterion

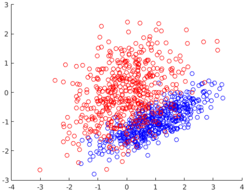
$$w = S_W^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2)$$

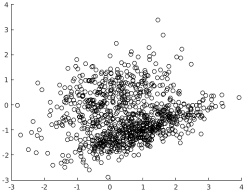
Where S_W is the total within class covariance matrix:

$$S_W = \sum_{n \in C_1} (\mathbf{x}_i - \bar{\mathbf{x}}_1)(\mathbf{x}_i - \bar{\mathbf{x}}_1)^\top + \sum_{i \in C_2} (\mathbf{x}_i - \bar{\mathbf{x}}_2)(\mathbf{x}_i - \bar{\mathbf{x}}_2)^\top$$

Fisher's linear discriminant provides a specific projection $\mathbf{w}$ to one dimension for classification

Example: Mixture of Gaussians

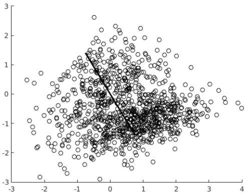


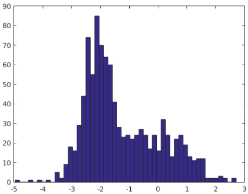


`scripts/gaussian_mixture.m`

```
mu1 = [1 −1]; Sigma1 = [.9 .3; .3 .3];
x1 = mvnrnd(mu1, Sigma1, 500);

mu2 = [0 0]; Sigma2 = [.9 .3; .3 .9];
x2 = mvnrnd(mu2, Sigma2, 500);

figure(); hold on;
scatter(x1(:,1),x1(:,2),'k')
scatter(x2(:,1),x2(:,2),'k')
```

Example: Mixture of Gaussians

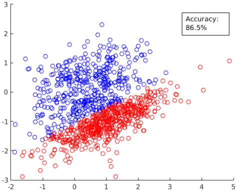


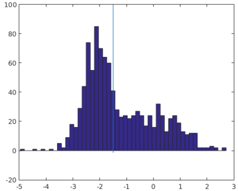


scripts/lda_analysis.m

```
%Fisher's LDA
m1 = mean(x1,1);
m2 = mean(x2,1);
invSw = inv(cov(x1)+cov(x2));
w = invSw*(m2-m1)';
%Project
y_lda = [x1;x2]*w

figure();
hist(y_lda,50);
```

Example: Mixture of Gaussians





`scripts/lda_classify.m`

```
figure();hist(y_lda,50);
line([−1.5,−1.5],[−1.5,100]);
classes_true = [zeros(500,1);ones(500,1)];
class1 = y_lda>−1.5;
class2 = y_lda<=−1.5;
figure();hold on;
scatter(x(class1,1),x(class1,2),'r');
scatter(x(class2,1),x(class2,2),'b');
```