Principal Component Analysis (PCA)

and

PCA Neural Networks: An Application of Unsupervised Hebbian Learning and intro to Independent Components Analysis (ICA)
What is PCA?

- A standard statistical technique for reducing dimensionality of data (without using neural approach).

- Purpose: Better understanding or communication of data; used a lot in the sciences to select the most important features.

- In so reducing, we want to lose as little information as possible, given the before- and after- dimensions.

- Also known as Karhuenen-Loeve (K-L) transformation (Watanabe, 1969).

- Could be used to preprocess input to, say, MLP.
PCA

n-dimensional vectors → m-dimensional vectors

m < n
Reconstruction from Components

weights

vector components

= composite

reconstruction
Scientific Uses

Transform coordinates to get a better understanding of the underlying phenomena.
Scientific Uses

Transform coordinates to get a better understanding of the underlying phenomena.
Comparison with Linear Regression

- Linear regression requires one to \textit{pre-identify} dependent vs. independent variables.

- PCA does not.
What is a “Principal Component”? 

- If we were to plot the data, the **first principal component** would be the **predominant direction** in the data.
- The second principal component would be the second-most predominant direction, etc. up to the number of **dimensions** of the data points.
The main idea

- Transform the input data into fewer dimensions
- Preserve as much of the variance as possible

transform from 2 to 1 dimension
Transformation

- Preserve as much of the variance as possible

rotate

more information (variance)

less information

project
What is a Predominant Direction?

- This may seem subjective, but a mathematical definition can be provided.

- Let $A$ be the matrix of data points:
  - Each point or observation is a column.
  - The rows correspond to the observed values of variables.
First P. C. Demo, you pick the points
http://www.igi.tugraz.at/lehre/CI/algorithms/pca_applet.html

This applet uses an iterative algorithm (APEX) to find the first PC.
Covariance vs. Correlation

• Most PCA presentations are based on covariance, which assumes input variables have commensurate units.

• Can also base PCA on correlation, which doesn’t.

• To use covariance in general, first normalize by dividing by each mean-adjusted variable by its variance.
Linear Transformation

- The types of transformations shown are linear:
  - $B = W A$
    where
    - $A$ is the matrix of data (each point as a column vector), which is already mean-adjusted.
    - $W$ is an $m \times n$ matrix, $m < n$
    - $B$ is the transformed data matrix (called the “scores”)
- We want the columns of $W$ to be **orthonormal** (i.e. $W W^T = I$).
- The reconstruction $A'$ of $A$ from $B$ will be obtained by:
  - $A' = W^T B$
    $= W^T(WA) = (W^T W)A.$
    (Note: $W^T W$ won’t generally be $I$, although $W W^T$ is.)
Computing Principal Components

- PCA tries to minimize the *expectation* of reconstruction error:

  \[ E[ \| A - A' \|^2 ] = \]
  \[ E[ \text{tr}( (A - A')(A-A')^T ) ] = \]
  \[ \text{tr}(E[AA']) - \text{tr}(WE[AA']W^T) = \]
  \[ \text{tr}(R) - \text{tr}(WRW^T) \]

  where

  - \( R \) is the *covariance matrix*, defined as \( E[AA^T] \)
  - \( E[ ] \) is the expectation, i.e. average over the data points
  - \( \text{tr} \) is the trace (sum of diagonal elements).
What are the Principal Components, really?

- Want $W$ such that error $\text{tr}(R) - \text{tr}(WRW^T)$ is minimized.

- Equivalently, $WRW^T$ (i.e. the variance of the transformed data) is maximized.

- The eigenvectors of the covariance matrix $R = AA^T$, ordered corresponding to largest to smallest eigenvalue, are the principal components.

- Construct matrix $W$ as using the top so-many correspondingly-ordered eigenvectors as rows.
Example (worked in Matlab)
Turtle Shell Classification (D. Morrison)

data =
(columns are data points)

1.3000  1.4000  1.5000  1.2000  1.1000  (Height)
3.2000  2.8000  3.1000  2.9000  3.0000  (Width)
3.7000  4.1000  4.6000  4.8000  4.8000  (Length)

means =

1.3000
3.0000
4.4000

mean_adjusted =

0   0.1000  0.2000  -0.1000  -0.2000
0.2000 -0.2000  0.1000  -0.1000   0
-0.7000 -0.3000  0.2000  0.4000  0.4000

[M,N] = size(data);
means = mean(data, 2);
mean_adjusted = data - repmat(means,1,N);

Can fewer than 3 variables explain all?
Example

covariance =

\[
\begin{pmatrix}
0.0250 & 0.0025 & -0.0275 \\
0.0025 & 0.0250 & -0.0250 \\
-0.0275 & -0.0250 & 0.2350 \\
\end{pmatrix}
\]

variances =

\[
\begin{pmatrix}
0.2415 \\
0.0225 \\
0.0210 \\
\end{pmatrix}
\]

explained_variation =

\[
\begin{pmatrix}
0.8472 \\
0.0791 \\
0.0737 \\
\end{pmatrix}
\]

covariance = \frac{\text{meanAdjusted} \times \text{meanAdjusted}'}{N-1};
[eigenvectors, eigenvalues] = eig(covariance);
[junk, rindices] = sort(-1*eigenvalues);
variances = eigenvalues(rindices);
explained_variation = variances/sum(variances)
Example: Principal Components

PC = eigenvectors(:, rindices);

% transform the original data set by column
scores = PC * mean_adjusted;

scores = (transformed data, columns)

-0.1265  0.5534  -0.8232
-0.1153  -0.8325  -0.5419
0.9852   -0.0263  -0.1691

-0.4656  -0.3703  0.1947  0.2866  0.3546
-0.5459  -0.0076  0.0021  0.3116  0.2398
-0.1236  0.0531  0.2282 -0.0283 -0.1294
Example: Orthonormality Check

orthogonality_check =

\[
\begin{pmatrix}
1.0000 & 0.0000 & 0 \\
0.0000 & 1.0000 & 0.0000 \\
0 & 0.0000 & 1.0000 \\
\end{pmatrix}
\]

\[\text{orthogonality_check} = \text{PC} \times \text{PC}'\]
Example: Reconstruction

\[
\text{subtracted\_means} = \\
\begin{bmatrix}
1.3000 & 1.3000 & 1.3000 & 1.3000 & 1.3000 \\
3.0000 & 3.0000 & 3.0000 & 3.0000 & 3.0000 \\
4.4000 & 4.4000 & 4.4000 & 4.4000 & 4.4000 \\
\end{bmatrix}
\]

\[
\text{reconstruction\_using\_all\_components} = \\
\begin{bmatrix}
1.3000 & 1.4000 & 1.5000 & 1.2000 & 1.1000 \\
3.2000 & 2.8000 & 3.1000 & 2.9000 & 3.0000 \\
3.7000 & 4.1000 & 4.6000 & 4.8000 & 4.8000 \\
\end{bmatrix}
\]

\[
\text{reconstruction\_error} = \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\text{reconstruction\_using\_all\_components} = \ldots \\
(\text{PC}' \ast \text{scores}) + \text{subtracted\_means}
\]

\[
\text{reconstruction\_error} = \text{reconstruction\_using\_all\_components} - \text{data}
\]
Example: Reconstruction

reconstruction_using_first_two_components =

1.4218  1.3477  1.2751  1.2278  1.2275
3.1967  2.8014  3.1060  2.8993  2.9966
3.7209  4.0910  4.5614  4.8048  4.8219

reconstruction_error =

0.1218  -0.0523  -0.2249  0.0278  0.1275
-0.0033  0.0014   0.0060 -0.0007 -0.0034
 0.0209 -0.0090  -0.0386  0.0048  0.0219

mse_reconstruction =

0.0058

reconstruction_using_first_two_components = ...
  (components(1:2, :)' * scores(1:2, :)) + subtracted_means

reconstruction_error = reconstruction_using_first_two_components - data

mse_reconstruction = mse(reconstruction_error)
Example: Reconstruction

reconstruction_using_first_component_only =

1.3589  1.3468  1.2754  1.2637  1.2551
2.7423  2.7951  3.1077  3.1586  3.1962
4.0167  4.0952  4.5603  4.6359  4.6919

reconstruction_error =

0.0589  -0.0532  -0.2246  0.0637  0.1551
-0.4577  -0.0049  0.0077  0.2586  0.1962
0.3167  -0.0048  -0.0397  -0.1641  -0.1081

mse_reconstruction =

0.0360

reconstruction_using_first_component_only = ...
   (components(1:1, :)' *scores(1:1, :)) + subtracted_means

reconstruction_error = reconstruction_using_first_component_only - data

mse_reconstruction = mse(reconstruction_error)
Matlab cov Function

A = (mean adjusted, by row)

\[
\begin{bmatrix}
0 & 0.2000 & -0.7000 \\
0.1000 & -0.2000 & -0.3000 \\
0.2000 & 0.1000 & 0.2000 \\
-0.1000 & -0.1000 & 0.4000 \\
-0.2000 & 0 & 0.4000 \\
\end{bmatrix}
\]

>> R = cov(A)

R =

\[
\begin{bmatrix}
0.0250 & 0.0025 & -0.0275 \\
0.0025 & 0.0250 & -0.0250 \\
-0.0275 & -0.0250 & 0.2350 \\
\end{bmatrix}
\]
Matlab pcacov function

```matlab
>> [pc, variances, explained] = pcacov(R)

pc =

-0.1265  0.5534  0.8232
-0.1153 -0.8325  0.5419
 0.9852 -0.0263  0.1691

variances =

0.2415
0.0225
0.0210

explained =

84.7212
 7.9114
 7.3674

Eigenvalues

% Variance explained
Singular Value Decomposition: Another approach to computing PCs

- The singular-value decomposition (SVD) of $A$ is:
  $$ A = U S V^T $$

- where:
  - The columns of $U$ are the eigenvectors of $A A^T$.
  - The columns of $V$ are the eigenvectors of $A^T A$, i.e. the PC’s.
  - $S$ is pseudo-diagonal (diagonal insofar as this is possible with a rectangular matrix, the rest of the entries being 0).

- PC from SVD in Matlab:
  $$ Y = \text{mean\_adjusted} / \sqrt{N-1} $$

  $$ [U, S, \text{PC}] = \text{svd}(Y') $$

  [Note: $Y' = U S \text{PC}'$]
Planets Example

• Consider a 3-dimensional data set where the variables are the logarithms of
  • distance to the sun
  • equatorial diameter
  • density
## Data set
*(prior to taking logs)*

<table>
<thead>
<tr>
<th>planet</th>
<th>distance</th>
<th>diameter</th>
<th>density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>0.387</td>
<td>4878</td>
<td>5.42</td>
</tr>
<tr>
<td>Venus</td>
<td>0.723</td>
<td>12104</td>
<td>5.25</td>
</tr>
<tr>
<td>Earth</td>
<td>1.000</td>
<td>12756</td>
<td>5.52</td>
</tr>
<tr>
<td>Mars</td>
<td>1.524</td>
<td>6787</td>
<td>3.94</td>
</tr>
<tr>
<td>Jupiter</td>
<td>5.203</td>
<td>142800</td>
<td>1.314</td>
</tr>
<tr>
<td>Saturn</td>
<td>9.539</td>
<td>120660</td>
<td>0.69</td>
</tr>
<tr>
<td>Uranus</td>
<td>19.18</td>
<td>51118</td>
<td>1.29</td>
</tr>
<tr>
<td>Neptune</td>
<td>30.06</td>
<td>49528</td>
<td>1.64</td>
</tr>
<tr>
<td>Pluto</td>
<td>39.53</td>
<td>2300</td>
<td>2.03</td>
</tr>
</tbody>
</table>
Log Data Projections
(projections being a special case of linear transformation)

In two dimensions, we can plot any one variable against another, e.g.

Of various possible plots, which show the widest variation among planets?
Maximizing Variance

• Transforming using first two principal components preserves more of the variance (summing variances in each dimension) in the projection than does projecting on any 2 variables:
Application: Handwritten Character recognition (K.S. Fu, 1968)

- Attempt recognition based upon **18 radial measurements**, spaced at 20 degree increments, of characters (240 samples).
- Recognition rate was computed vs. the number of measurements used (2, 4, 6, … out of 18) in no particular order.
- The same computation was done with the measurements **ordered by principal components**.
- The next slide compares the two approaches.
A Related Example
(also with the help of http://www.cs.mcgill.ca/~sqrt/dimr/dimreduction.html)

- A 25-dimensional data set:
Projected Data

- Projecting on 2 variables does not help much in discriminating the points, e.g.
Projected Data

- Projecting on the first two principal components achieves the maximum variation in two dimensions:
Eigenfaces Example

- Principal components of a raster image
PCA Source

L.T. Jolliffe
What is a PCA Neural Network?

- The matrix $W$ of PCA can be interpreted as **weights** of a 1-layer neural network.

- Training is by unsupervised Hebbian learning.

- The user pre-selects the output variables.

- The network effectively learns the first PC by outputting the corresponding scores.
3→2 PCA Network
PCA Hebbian-Learning Rules

\[ \Delta W = \eta y_i x_i^T - K_i W, \]  
where

- \( x_i \) is an input vector
- \( y_i \) is the corresponding output vector (= \( W x_i \))
- \( y_i x_i^T \) is the Hebbian component
- \( K_i \) is one of the following (purpose: weight decay):
  - 0 pure Hebbian (unbounded)
  - \( y_i y_i^T \) (Williams’ rule, 1985)
  - \( 3D(y_i y_i^T) + 2L(y_i y_i^T) \) (Oja and Karhunen rule, 1985)
  - \( L(y_i y_i^T) \) (Sanger’s rule, 1989)

where \( D(M) \) maps the diagonal entries of \( M \) to themselves, and other entries to 0, and

- \( L(M) \) maps entries below the main diagonal to themselves, and other entries to 0.
Other Variations Exist: Source for PCA NN

Diamantaras, K.I. / Kung, S.Y.

Principal Component Neural Networks
Theory and Applications

1996. 256 pages.

The coverage in this book is extensive. It gives several additional learning algorithms, including ones with lateral connections as well as feed-forward ones.
Other means of learning for dimensionality reduction

- Self-Organizing Map
  (# of dimensions in superstructure = reduced dimension)

- LVQ
  (# of neurons = reduced dimension)

- Projection pursuit, another statistical technique, effectively learns PC’s sequentially
PCA Shortcomings

- The PCA user must select **how many** target dimensions to use.
- PCA only finds **linear** sub-spaces.
- PCA works best if the individual components are Gaussian-distributed.
- There are many variations, some of which overcome some of these issues.
- Related area: “Factor Analysis”
Independent Component Analysis (ICA)

- A technique for BSS (Blind Source Separation).

- Proposed for neuro-mimetic hardware in 1983 by Herault and Jutten.

- ICA seeks to transform to components that are statistically independent.

- Two variables $x$, $y$ are statistically independent iff
  $$P(x \ y) = P(x)P(y).$$
  Equivalently,
  $$E\{g(x)h(y)\} - E\{g(x)\} E\{h(y)\} = 0$$
  where $g$ and $h$ are any functions.
PCA vs. ICA

- PCA is finding uncorrelated components.
- ICA finds independent components.
- Independent implies uncorrelated, but not necessarily conversely.
- The input signals to ICA, being mixtures, will usually be correlated. The outputs, being independent, will not be.
Signals vs. Mixtures

Joint distribution of two independent signals

Joint distribution of a mixture of the same signals
Suppose we have $n$ distinct mixtures of $n$ statistically-independent signals.

We wish to transform the mixtures to (signals as close as possible to) the original signals.
Fast-ICA Demo
Signals to be Mixed
Whitened Signals after Mixing
Extracted Independent Components
ICA Demos

- http://www.cnl.salk.edu/~tewon/Blind/blind_audio.html
How can/does ICA work?

- One version: Fast ICA
- Throughout signals are assumed to be mean-adjusted.
Correlation within Signal Mixture

(from http://www.cis.hut.fi/projects/ica/icademo/)

Input signals and density
Effect of whitening (aka “sphering”): removing correlations in mixed signals

Whitened signals and density
Whitening Transformation
(essentially a form of PCA)

• Seek a linear transformation $V$ such that when $y = Vx$ we get $E\{yy'\} = I$.

• Set $V = C^{-1/2}$, where $C = E\{xx'\}$ is the correlation matrix of the data.

• Then $E\{yy'\} = E\{Vxx'V'\} = C^{-1/2}CC^{-1/2} = I$.

• For computational aspects, see:
Rotating the Whitened Signals

- The independent components are achieved by an appropriate rotation of the whitened signals.

- This involves maximizing the non-normal aspects of the marginal densities, since

- A linear mixture of independent random variables is necessarily more Gaussian than the original variables

- Maximization can be done incrementally, using an approach such as Fast-ICA.
Separated Signals

Separated signals after 5 steps of FastICA
Fast ICA Intermediate Steps

Separated signals after 1 step of FastICA
Fast ICA Intermediate Steps

Separated signals after 2 steps of FastICA
Fast ICA Intermediate Steps

Separated signals after 3 steps of FastICA
Fast ICA Intermediate Steps

Separated signals after 4 steps of FastICA
Neural ICA Algorithms

- Bell and Sejnowski: “infomax” method, minimizing mutual information, using an iterative formulation similar to PCA NN:

\[ B_{k+1} = B_k + \beta_k[B_k^{-T} + z_kx_k^T] \]

\[ z(i) = \frac{\partial}{\partial u(i)} \frac{\partial u(i)}{\partial y(i)} \]

\[ u = f(Bx); \ f = \text{tansig, etc.} \]

- Also related to “maximum likelihood estimation”. Subsequent Bayesian derivation by Kevin Knuth
% Begin gradient ascent on h ...
for iter=1:maxiter
    % Get estimated source signals, y.
    y = x*W; % wt vec in col of W.
    % Get estimated maximum entropy signals Y=cdf(y).
    Y = tanh(y);
    % Find value of function h.
    % h = log(abs(det(W))) + sum( log(eps+1-Y(:).^2) )/N;
    detW = abs(det(W));
    h = ( (1/N)*sum(sum(Y)) + 0.5*log(detW) );
    % Find matrix of gradients @h/@W_ji ...
    g = inv(W') - (2/N)*x'*Y;
    % Update W to increase h ...
    W = W + eta*g;
    % Record h and magnitude of gradient ...
    hs(iter)=h; gs(iter)=norm(g(:));
end;
Neural ICA Algorithms

- Bell-Sejnowski was similar to the earlier Herault-Jutten formulation:

\[ B = (I+S)^{-1} \]

\[ S_{k+1} = S_k + \beta_k g(y_k)h(y_k^T) \]

\[ g = t, \ h = t^3; \ g = \text{hardlim}, \ h = \text{tansig} \]
Neural ICA Algorithms

- EASI (Equivariant Adaptive Separation via Independence): Cardoso et al

\[ B_{k+1} = B_k - \beta_k [y_k y_k^T - I + g(y_k)h(y_k)^T - (y_k)g(y_k)^T]B_k \]

\( g=t, \ h=\text{tansig} \)
ICA Image Separation Demo
(David Gleich ‘03)
ICA Image Separation Example (David Gleich Gleich ‘03)

Bell and Sejnowski

EASI Algorithm

Original images
4-source mixture
4-source separation

Simone Fiori

Amari’s Recurrent Neural ICA

- Fully recurrent neural network with self-inhibitory connections.

$$\tau_i \frac{dy_i}{dt} + y_i = x_i(t) - \sum_{i=1}^{n} \hat{w}_{ij}(t)y_j,$$

$$y(t) = (I + \hat{W}(t))^{-1} x(t),$$

$$y(t) = x(t) - \hat{W}(t)y(t - \tau),$$

$$\frac{d\hat{W}}{dt} = -\mu(t)[I + \hat{W}][\Lambda - f(y(t))g^T(y(t))].$$
ICA was used to achieve a separation among 12 EEG channels, used to discriminate between two or three modes of thought.
Eye-blink EEG removal
(Peterson and Anderson, 1999)

Fig. 1. Eye blink subtraction with ICA
Financial Application

Figure 13: (from Kiviluoto and Oja, 1998). Five samples of the original cashflow time series (mean removed, normalized to unit standard deviation). Horizontal axis: time in weeks.

From Aapo Hyvärinen and Erkki Oja
Neural Networks, 13(4-5):411-430, 200
Figure 14: (from Kiviluoto and Oja, 1998). *Four independent components or fundamental factors found from the cashflow data.*
Figure 15: (from Hyvärinen, 1999d). An experiment in denoising. Upper left: original image. Upper right: original image corrupted with noise; the noise level is 50%. Lower left: the recovered image after applying sparse code shrinkage. Lower right: for comparison, a wiener filtered image.
ICA Sources

James V. Stone

Aapo Hyvärinen,

Andrzej Cichocki, Shun-ichi Amari
ICA Shortcomings

- ICA does not take sequence or structure into account; it is purely statistical.
- It might be possible to do better by considering the ignored information.