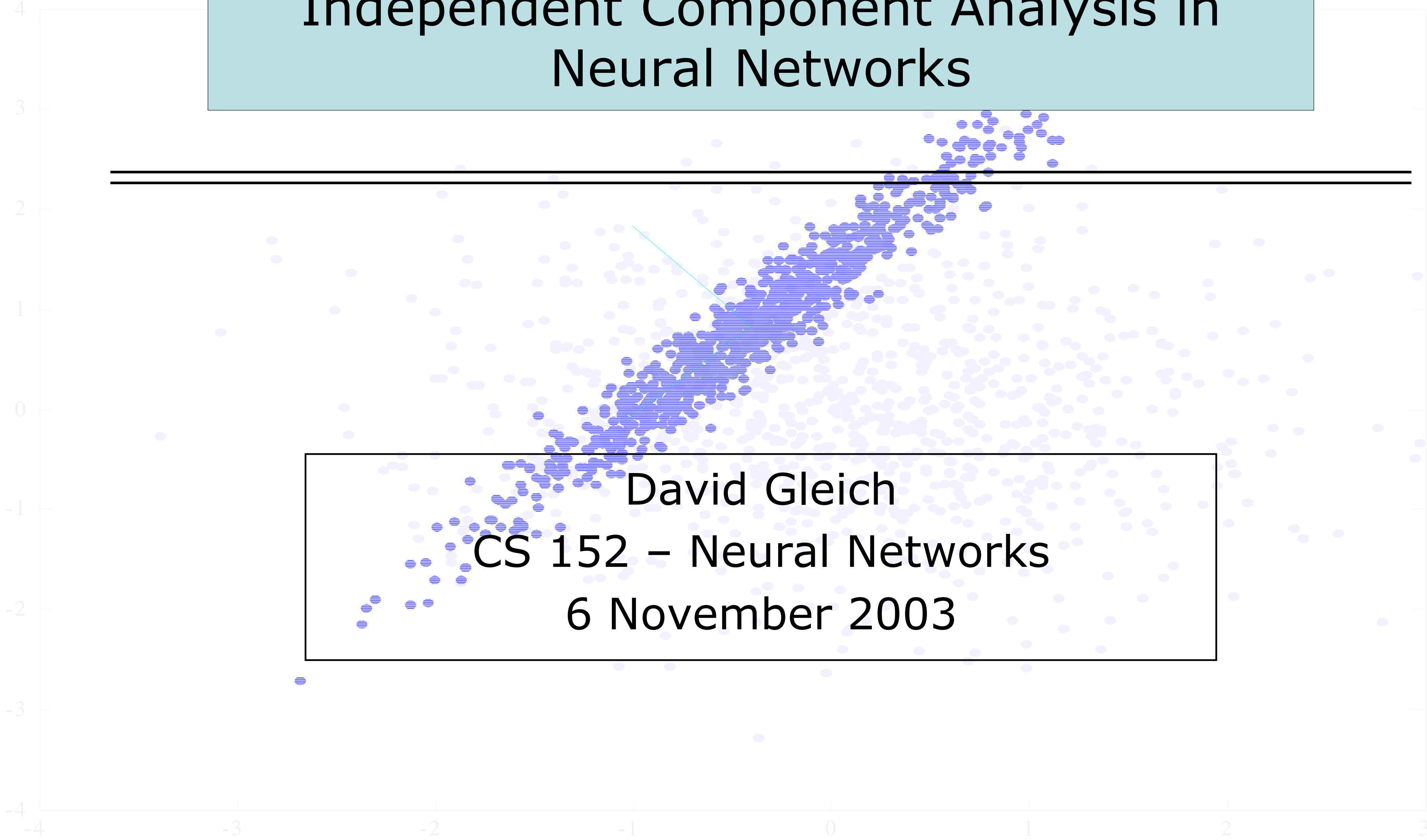
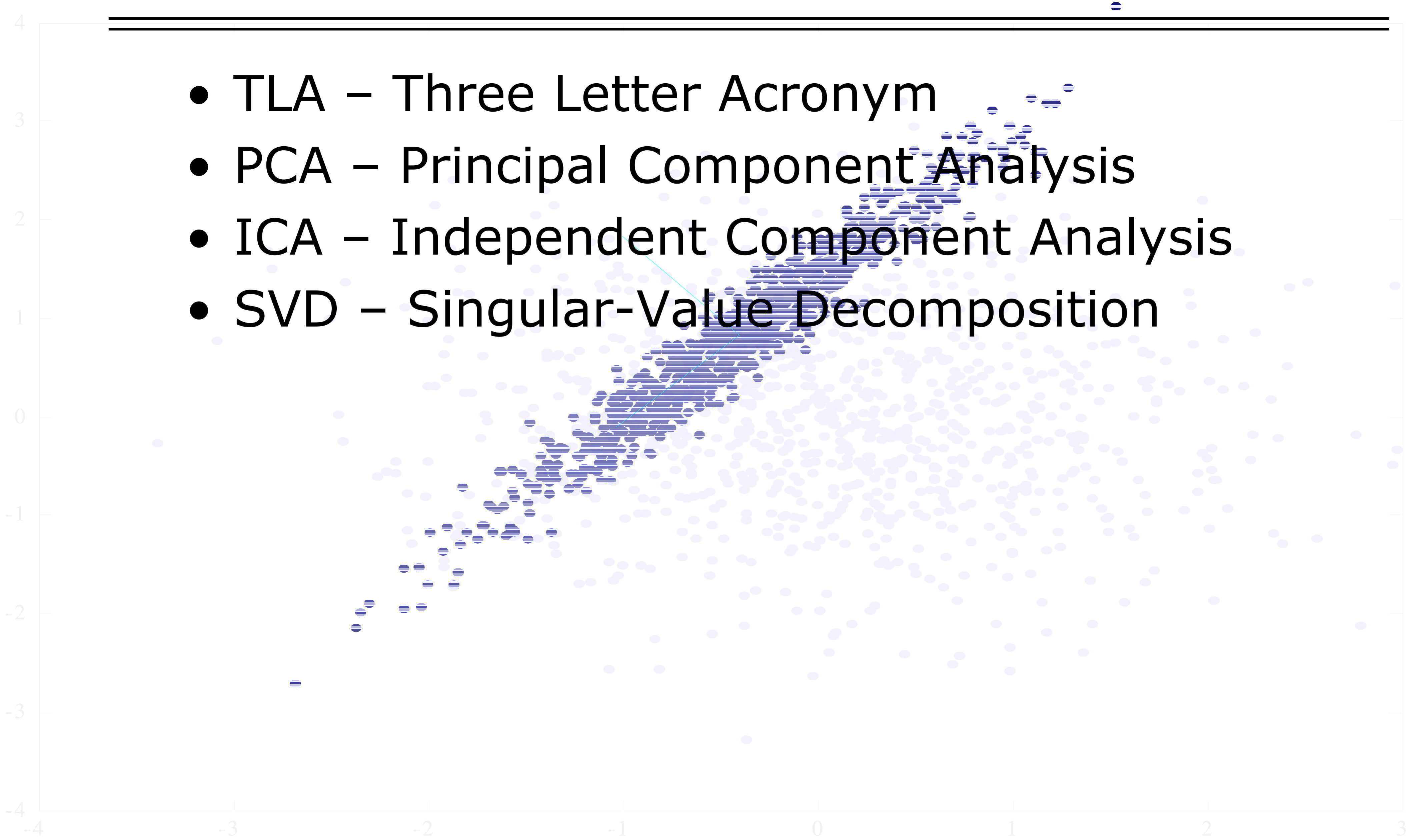
Principal Component Analysis and Independent Component Analysis in Neural Networks



David Gleich
CS 152 – Neural Networks
6 November 2003

TLAs

- TLA – Three Letter Acronym
- PCA – Principal Component Analysis
- ICA – Independent Component Analysis
- SVD – Singular-Value Decomposition



Outline

- Principal Component Analysis
  - Introduction
  - Linear Algebra Approach
  - Neural Network Implementation
- Independent Component Analysis
  - Introduction
  - Demos
  - Neural Network Implementations
- References
- Questions

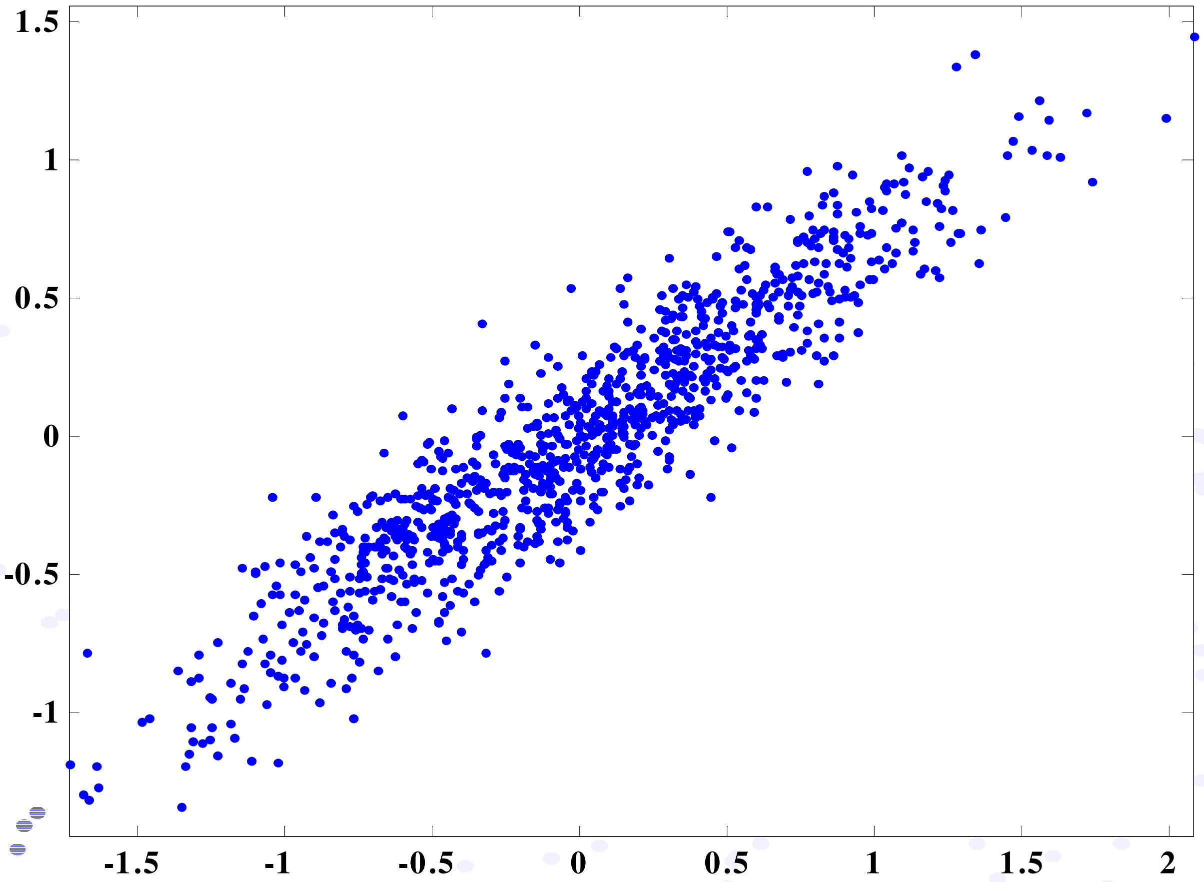
Principal Component Analysis

- PCA identifies an m dimensional explanation of n dimensional data where $m < n$.
- Originated as a statistical analysis technique.
- PCA attempts to minimize the reconstruction error under the following restrictions
  - Linear Reconstruction
  - Orthogonal Factors
- Equivalently, PCA attempts to maximize variance, proof coming.

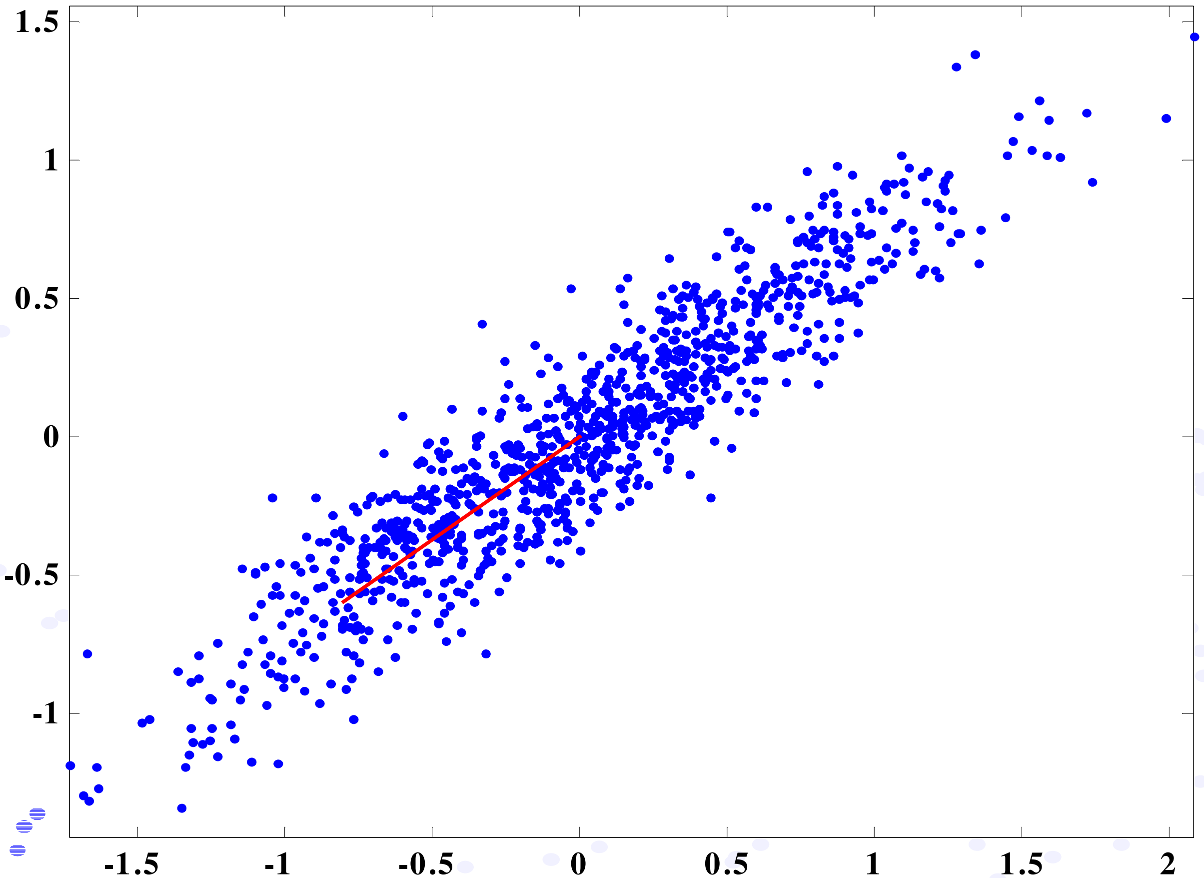
PCA Applications

- Dimensionality Reduction (reduce a problem from n to m dimensions with m << n)
- Handwriting Recognition – PCA determined 6-8 “important” components from a set of 18 features.

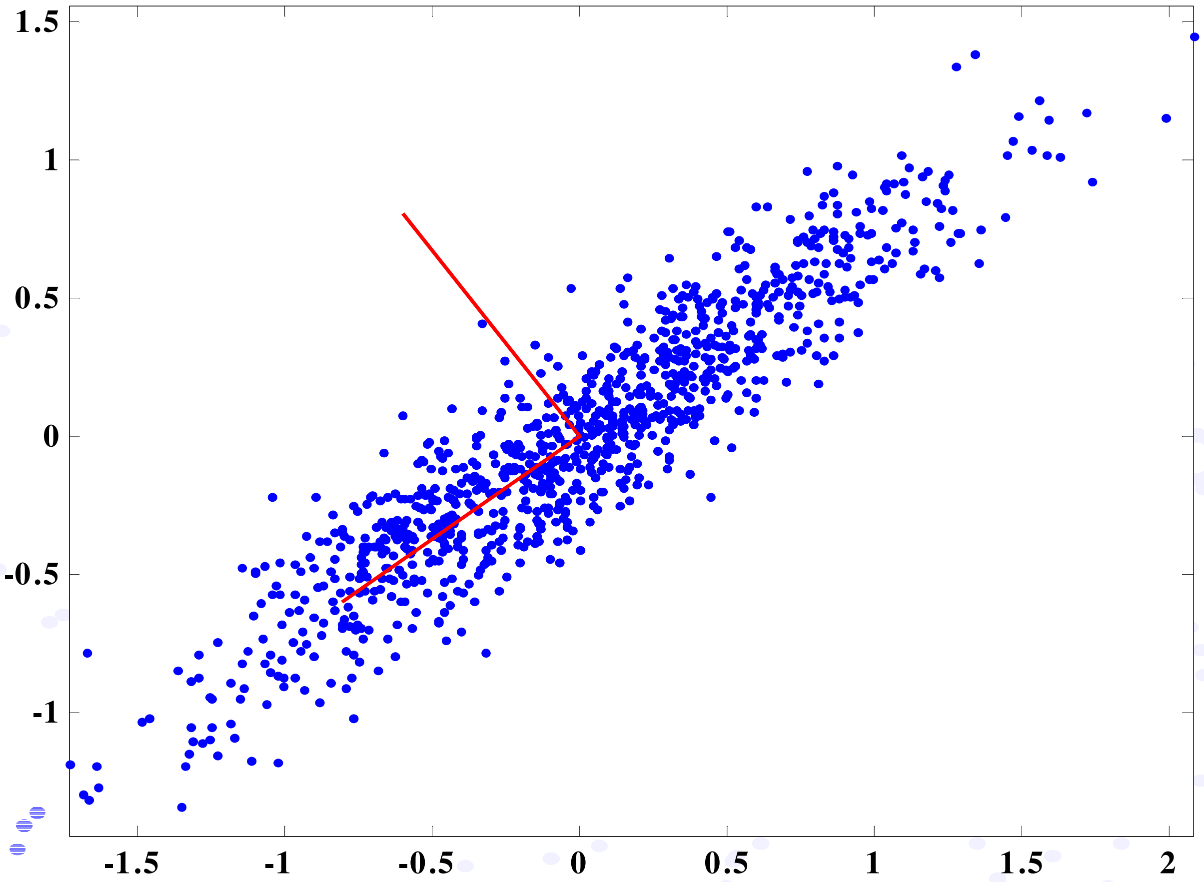
PCA Example



PCA Example



PCA Example



Minimum Reconstruction Error) Maximum Variance

Proof from Diamantaras and Kung

Take a random vector $x=[x_1, x_2, \ldots, x_n]^T$ with $E\{x\} = 0$, i.e. zero mean.

Make the covariance matrix $R_x = E\{xx^T\}$.

Let $y = Wx$ be a orthogonal, linear transformation of the data.

$$WW^T = I$$

Reconstruct the data through W^T.

$$\hat{x} = W^T y = W^T W x$$

Minimize the error.

$$E\{||x - \hat{x}||^2\}$$

Minimum Reconstruction Error) Maximum Variance

$$
\begin{aligned}
J_e &= E\{||x - \hat{x}||^2\} \\
&= E\{\text{tr}[(x - \hat{x})(x - \hat{x})^T]\} \\
&= E\{\text{tr}[xx^T - x\hat{x}^T - \hat{x}x^T + \hat{x}\hat{x}^T]\} \\
&= \text{tr}(R_x) - \text{tr}(R_x W^T W) - \text{tr}(W^T W R_x) - \text{tr}(W^T R_x W) \\
&= \text{tr}(R_x) - \text{tr}(W R_x W^T)
\end{aligned}
$$

$\text{tr}(WR_xW^T)$ is the variance of y

$$
\begin{aligned}
E\{(y - E\{y\})^2\} &= E\{\text{tr}(yy^T)\} \\
&= \text{tr}(W R_x W^T)
\end{aligned}
$$

PCA: Linear Algebra

- Theorem: Minimum Reconstruction, Maximum Variance achieved using

$$W = [\S e_1, \S e_2, \ldots, \S e_m]^T$$

  where e_i is the i^{th} eigenvector of R_x with eigenvalue λ_i and the eigenvalues are sorted descendingly.
- Note that W is orthogonal.

PCA with Linear Algebra

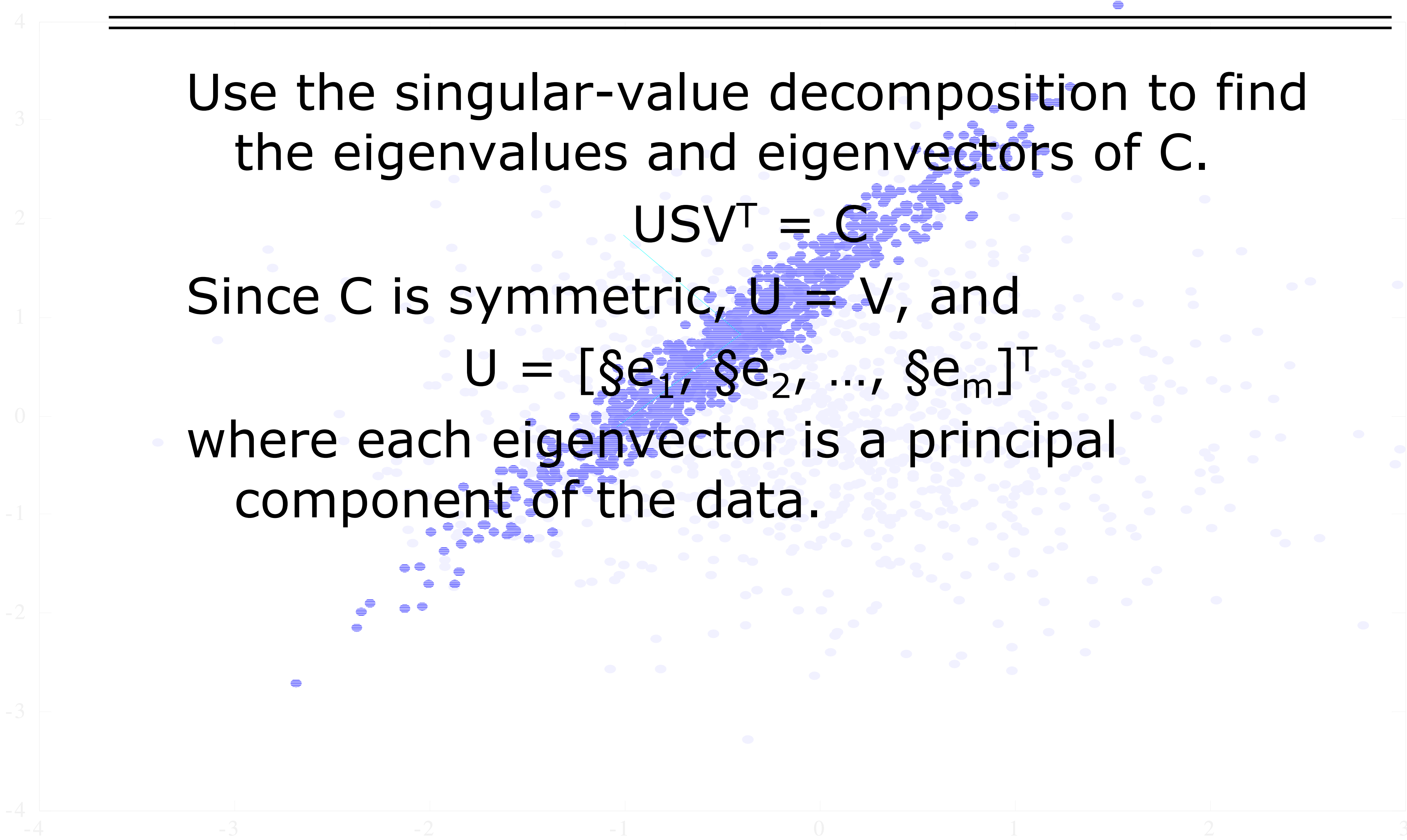
Given m signals of length n, construct the data matrix

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}.$$

Then subtract the mean from each signal and compute the covariance matrix

$$C = XX^T.$$

PCA with Linear Algebra



Use the singular-value decomposition to find the eigenvalues and eigenvectors of C.

$$USV^T = C$$

Since C is symmetric, U = V, and

$$U = [\S e_1, \S e_2, \ldots, \S e_m]^T$$

where each eigenvector is a principal component of the data.

PCA with Neural Networks

- Most PCA Neural Networks use some form of Hebbian learning.

  "Adjust the strength of the connection between units A and B in proportion to the product of their simultaneous activations."

  $$w_{k+1} = w_k + \beta_k(y_k\ x_k)$$

- Applied directly, this equation is unstable.

  $$||w_k||^2\ !\ 1 \text{ as } k\ !\ 1$$

- Important Note: neural PCA algorithms are unsupervised.

PCA with Neural Networks

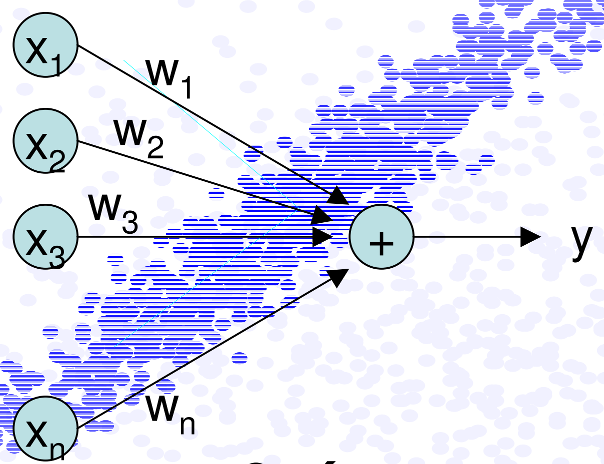
- Simplest fix: normalization.

$$w'_{k+1} = w_k + \beta_k(y_k\, x_k)$$
$$w_{k+1} = w'_{k+1}/||w'_{k+1}||^2$$

- This update is equivalent to a power method to compute the dominant eigenvector and as k ! 1, w_k ! e_1.

PCA with Neural Networks

- Another fix: Oja's rule.
- Proposed in 1982 by Oja and Karhunen.

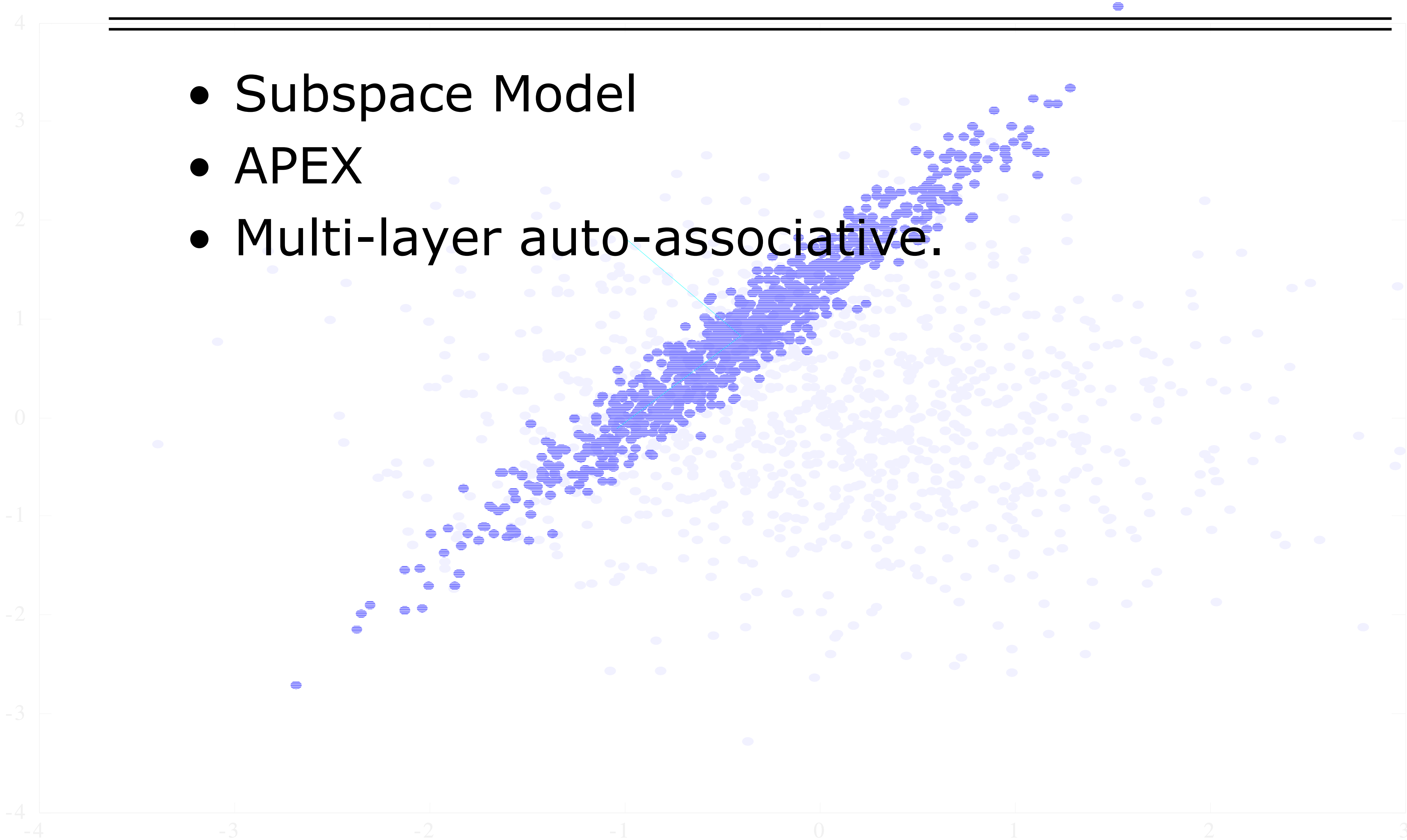


$$w_{k+1} = w_k + \beta_k(y_k\, x_k - y_k^2\, w_k)$$

- This is a linearized version of the normalized Hebbian rule.
- Convergence, as k ! 1, w_k ! e_1.

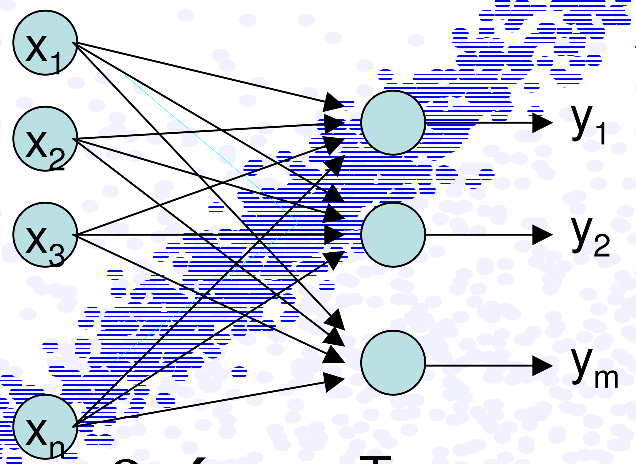
PCA with Neural Networks

- Subspace Model
- APEX
- Multi-layer auto-associative.



PCA with Neural Networks

- Subspace Model: a multi-component extension of Oja's rule.

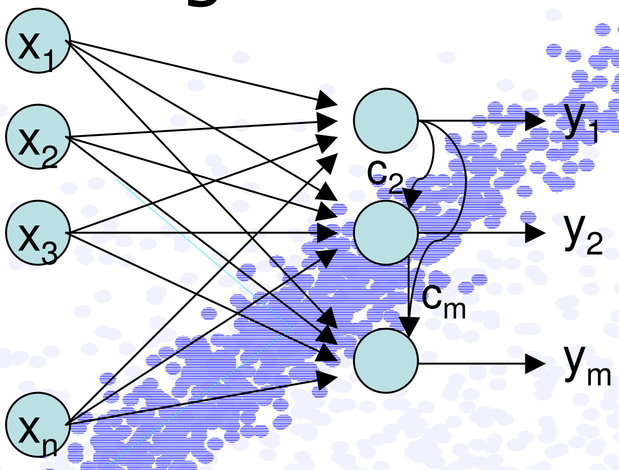


$$\Delta W_k = \beta_k(y_k x_k^T - y_k y_k^T W_k)$$

Eventually W spans the same subspace as the top m principal eigenvectors. This method does not extract the exact eigenvectors.

PCA with Neural Networks

- APEX Model: Kung and Diamantaras



$$y = Wx - Cy \; , \; y = (I+C)^{-1}Wx \text{ ¼ } (I-C)Wx$$

$$C = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ c_{21} & 0 & 0 & \cdots & 0 \\ c_{31} & c_{32} & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ c_{m-1,1} & \cdots & c_{m-1,m-2} & 0 & 0 \\ c_{m,1} & \cdots & \cdots & c_{m,m-1} & 0 \end{bmatrix}$$

PCA with Neural Networks

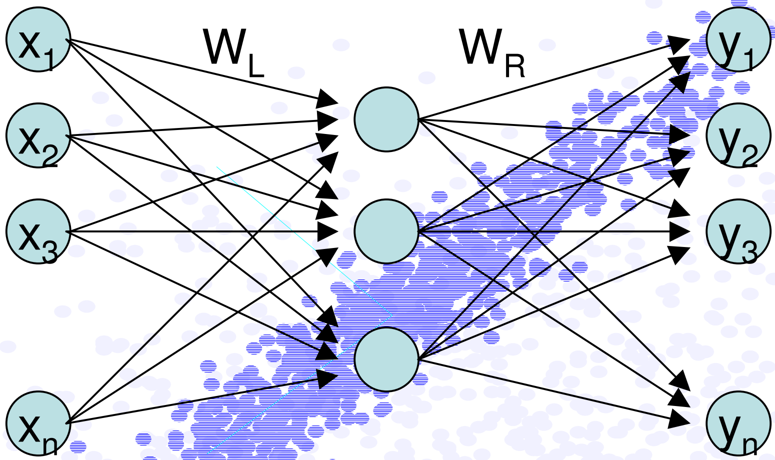
- APEX Learning

$$\Delta w_{i,j,k} = \beta_k (y_{ik} x_{jk} - y_{ik}^2 w_{i,j,k})$$
$$\Delta c_{i,j,k} = \beta_k (y_{ik} y_{jk} - y_{ik}^2 c_{i,j,k})$$

- Properties of APEX model:
  - Exact principal components
  - Local updates, Δw_{ab} only depends on x_a, x_b, w_{ab}
  - "-Cy" acts as an orthogonalization term

PCA with Neural Networks

- Multi-layer networks: bottlenecks



- Train using auto-associative output.

$$e = x - y$$

- W_L spans the subspace of the first m principal eigenvectors.

Outline

- Principal Component Analysis
  - Introduction
  - Linear Algebra Approach
  - Neural Network Implementation
- Independent Component Analysis
  - Introduction
  - Demos
  - Neural Network Implementations
- References
- Questions

Independent Component Analysis

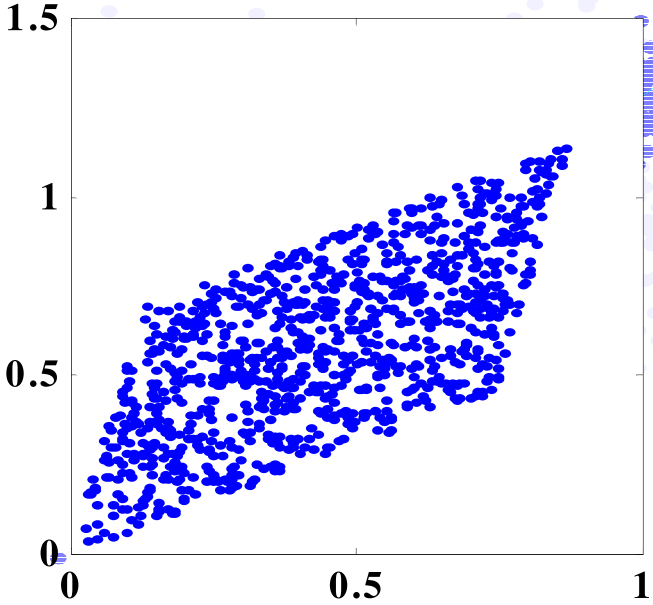
- Also known as Blind Source Separation.
- Proposed for neuromimetic hardware in 1983 by Herault and Jutten.
- ICA seeks components that are independent in the statistical sense.

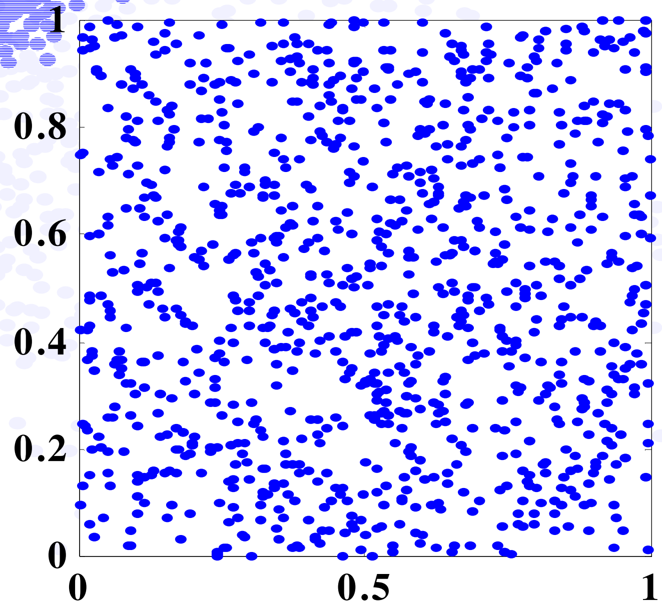
Two variables x, y are *statistically independent* iff P(x Å y) = P(x)P(y).

Equivalently,
E{g(x)h(y)} – E{g(x)}E{h(y)} = 0
where g and h are any functions.

Statistical Independence

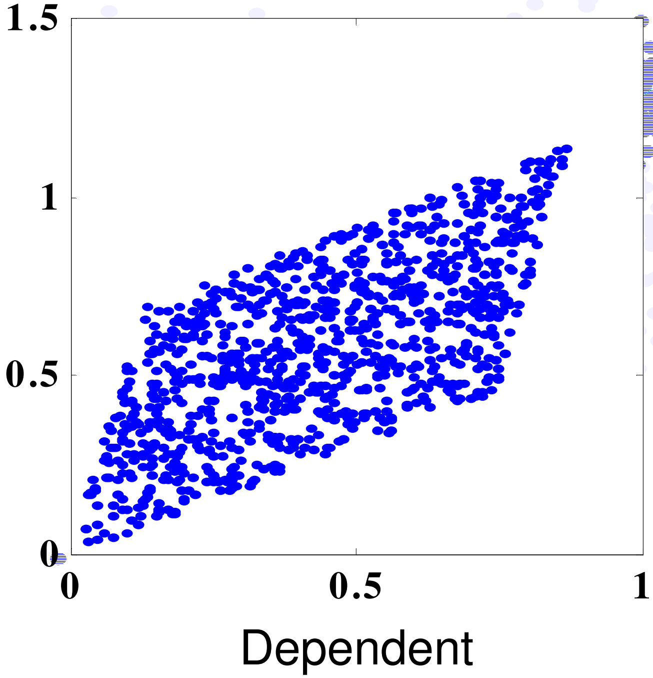
- In other words, if we know something about x, that should tell us *nothing* about y.

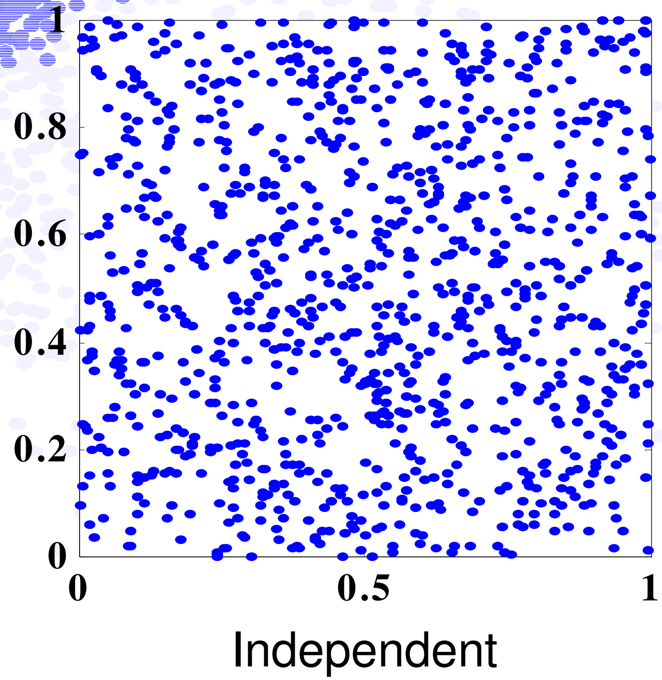




Statistical Independence

- In other words, if we know something about x, that should tell us *nothing* about y.





Independent Component Analysis

Given m signals of length n, construct the data matrix

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}.$$

We assume that X consists of m sources such that

X = AS

where A is an unknown m by m mixing matrix and S is m independent sources.

Independent Component Analysis

ICA seeks to determine a matrix W such that

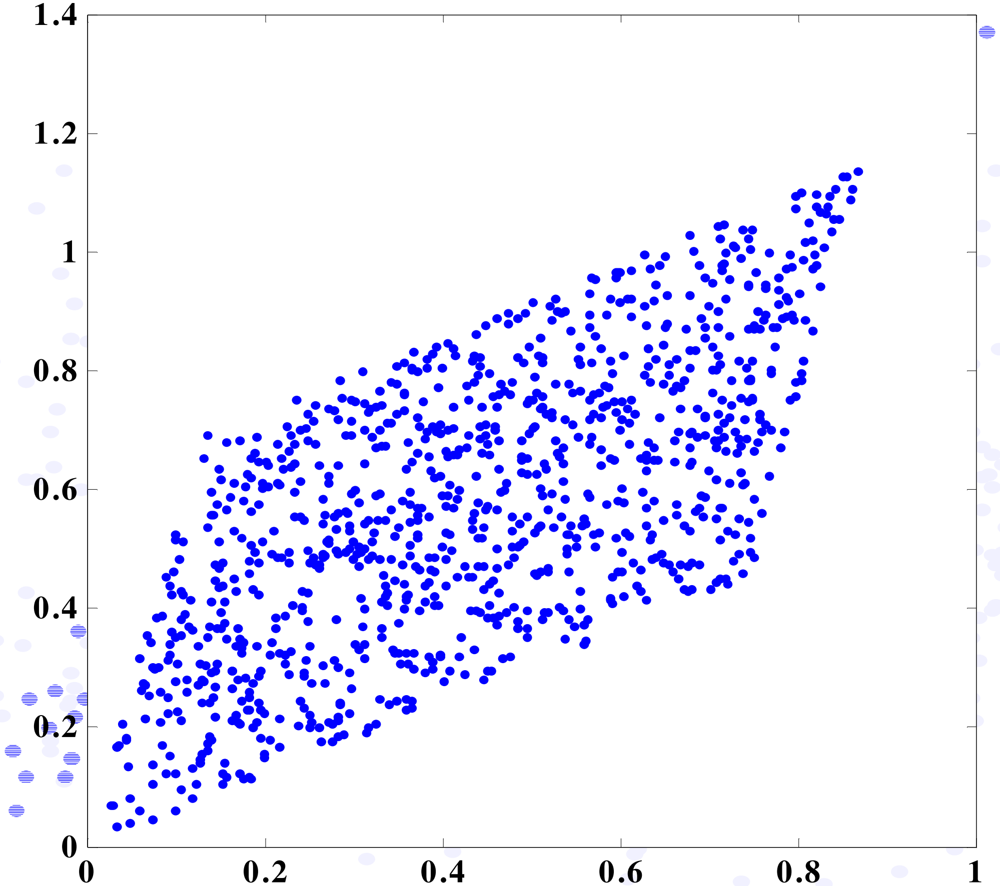
$$Y = WX$$

where W is an m by m matrix and Y is the set of independent source signals, i.e. the independent components.

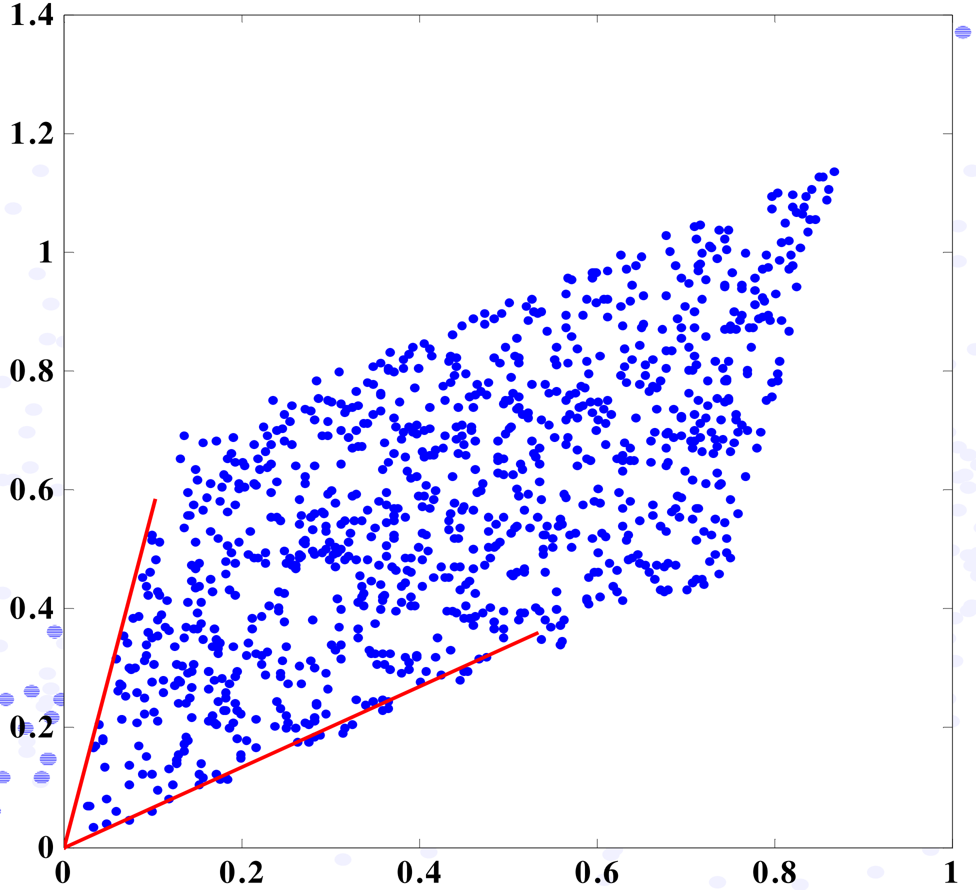
$$W \text{ ¼ } A^{-1} \text{) } Y = A^{-1}AX = X$$

- Note that the components need not be orthogonal, but that the reconstruction is still linear.

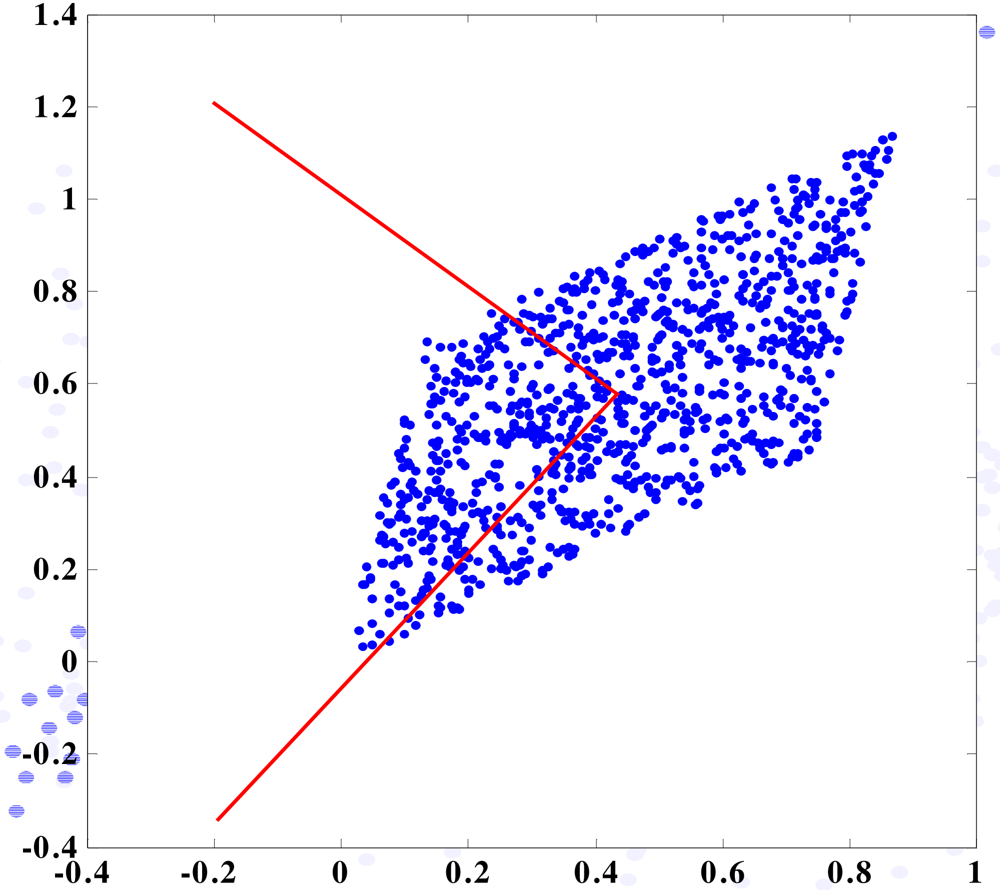
ICA Example



ICA Example

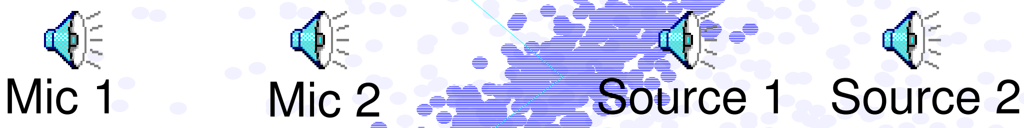


PCA on this data?



Classic ICA Problem

- The “Cocktail” party. How to isolate a single conversation amidst the noisy environment.



http://www.cnl.salk.edu/~tewon/Blind/blind_audio.html

More ICA Examples

More ICA Examples

Notes on ICA

- ICA cannot "perfectly" reconstruct the original signals.

If $X = AS$ then

1) if $AS = (A'M^{-1})(MS')$ then we lose scale
2) if $AS = (A'P^{-1})(PS')$ then we lose order

Thus, we can reconstruct only without scale and order.

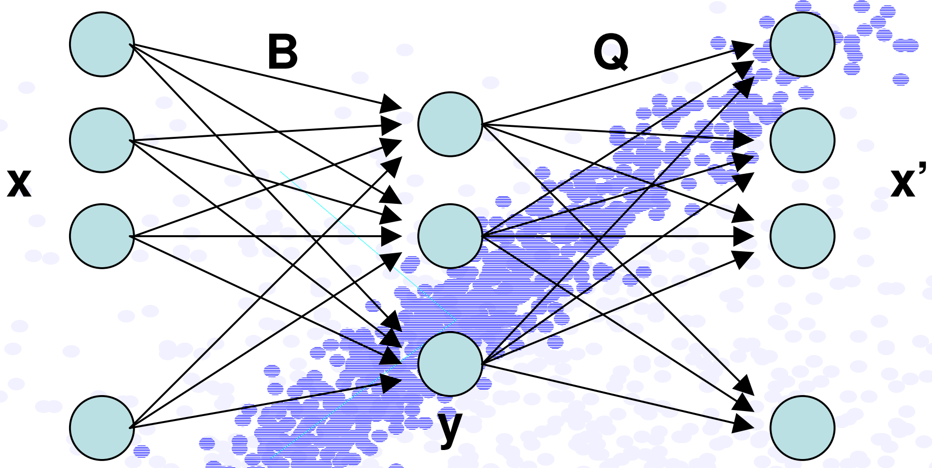
- Examples done with FastICA, a non-neural, fixed-point based algorithm.

Neural ICA

- ICA is typically posed as an optimization problem.
- Many iterative solutions to optimization problems can be cast into a neural network.

Feed-Forward Neural ICA

General Network Structure



1. Learn B such that y = Bx has independent components.
2. Learn Q which minimizes the mean squared error reconstruction.

Neural ICA

- Herault-Jutten: local updates

$$B = (I+S)^{-1}$$

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

$$g = t,\ h = t^3;\ g = \text{hardlim},\ h = \text{tansig}$$

- Bell and Sejnowski: information theory

$$B_{k+1} = B_k + \beta_k[B_k^{-T} + z_k x_k^T]$$

$$z(i) = \partial/\partial u(i)\ \partial u(i)/\partial y(i)$$

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

Recurrent Neural ICA

- Amari: Fully recurrent neural network with self-inhibitory connections.

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

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

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

$$\frac{d\widehat{W}}{dt} = -\mu(t)[I + \widehat{W}][\Lambda - f(y(t)) g^T(y(t))].$$

References

- Diamantras, K.I. and S. Y. Kung. *Principal Component Neural Networks*.
- Comon, P. "Independent Component Analysis, a new concept?" In *Signal Processing*, vol. 36, pp. 287-314, 1994.
- FastICA, http://www.cis.hut.fi/projects/ica/fastica/
- Oursland, A., J. D. Paula, and N. Mahmood. "Case Studies in Independent Component Analysis."
- Weingessel, A. "An Analysis of Learning Algorithms in PCA and SVD Neural Networks."
- Karhunen, J. "Neural Approaches to Independent Component Analysis."
- Amari, S., A. Cichocki, and H. H. Yang. "Recurrent Neural Networks for Blind Separation of Sources."

Questions?