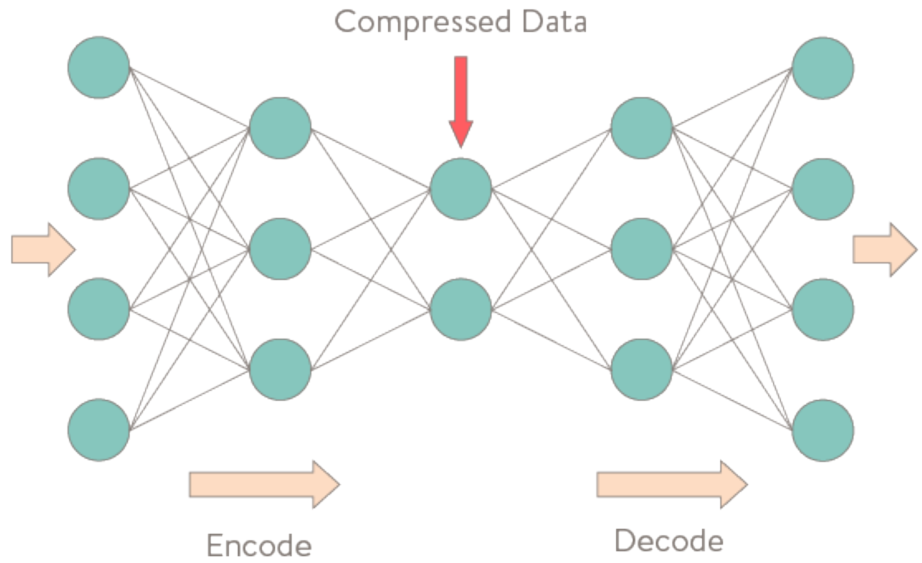


Autoencoders

Qian-Yuan TANG
tangqianyuan@gmail.com

July 13 2017 @CSRC, Beijing

Outline

Background: Dimensionality Reduction

- High Dimensional Descriptions vs Low Dimensional Descriptions
- Principle Component Analysis (PCA)
- The Limits of PCA

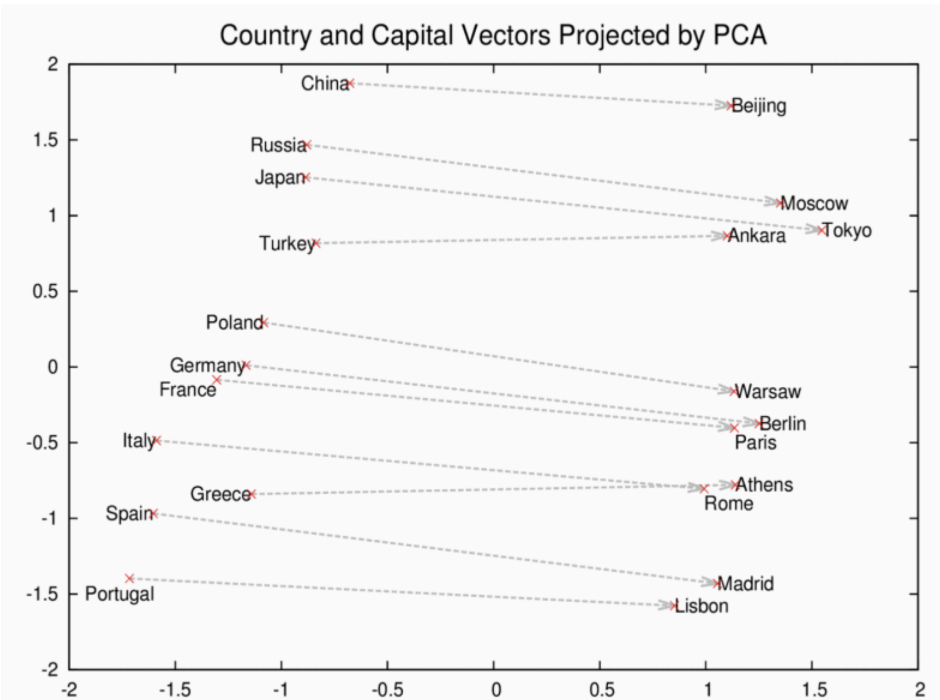
Autoencoders

- Basic Introductions
- Stacked Autoencoder and Deep Learning
- Sparse Autoencoder
- Denoising Autoencoder (DAE)
- Contrastive Autoencoder (CAE)
- Applications of Autoencoders

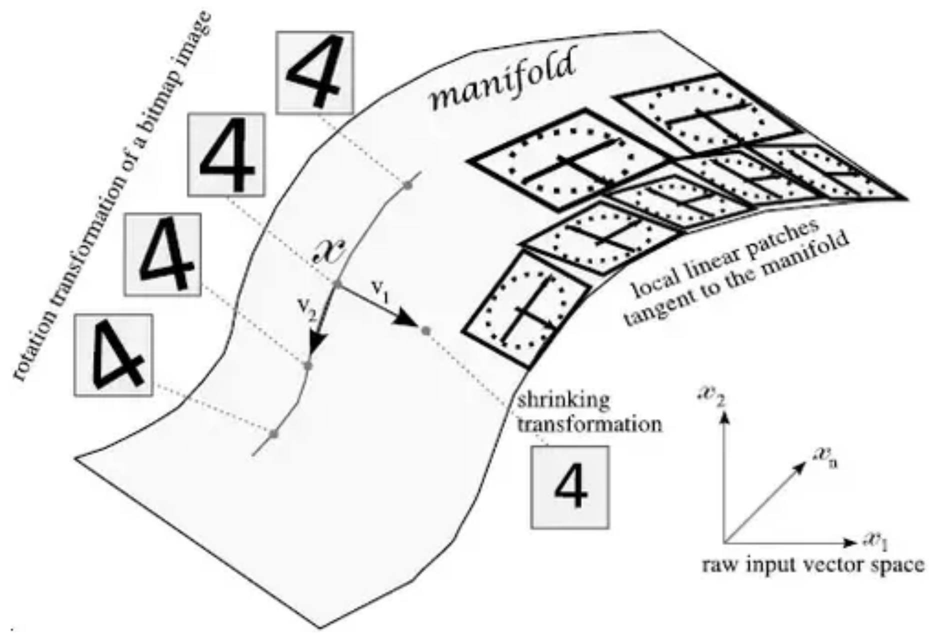
Additional Discussions

- Information Bottleneck Method
- Autoencoders as generative models (VAE and AAE)

Dimensionality Reduction: Examples (1)

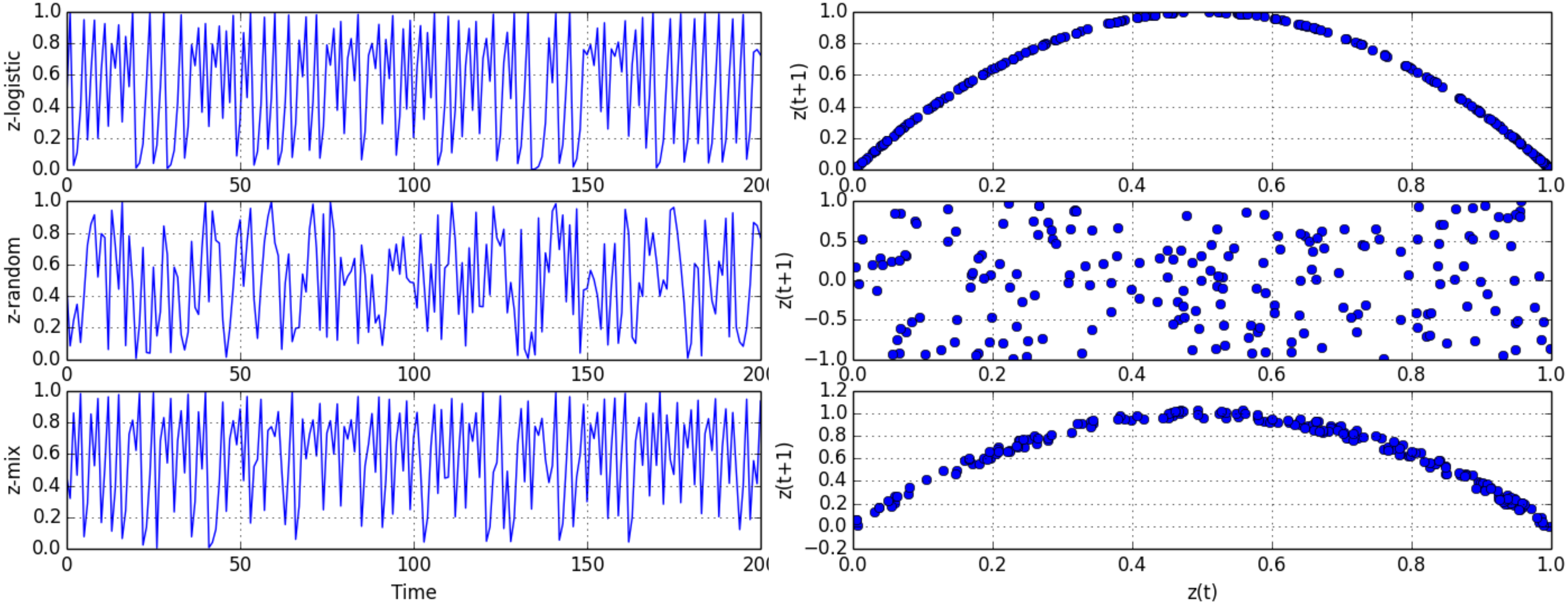


word2vec



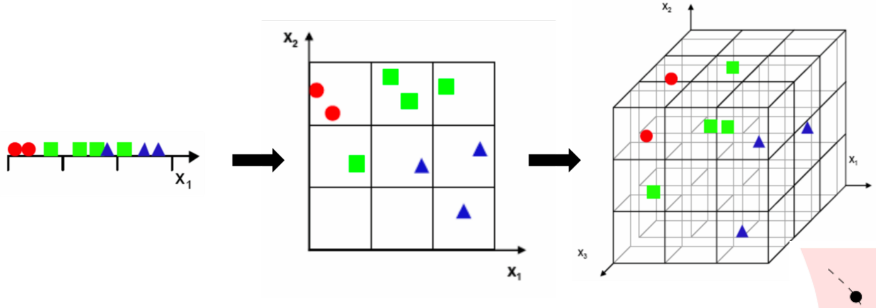
Manifold learning

Dimensionality Reduction: Examples (2)



Dimensionality reduction → Revealing the governing equations of a chaotic system

High-dimensional data vs. Low-dimensional data



The curse of the dimensionality!

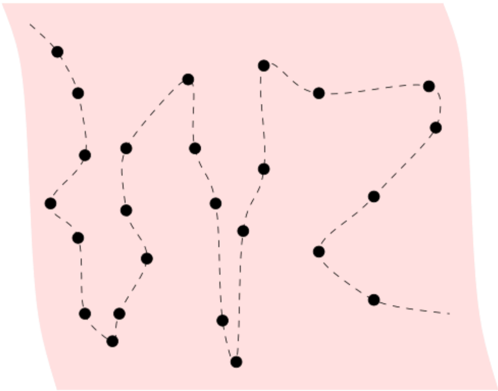


Figure 4.5: Curve or surface?

High-dimensional data

- Detailed descriptions of a system
- Data points become separable (kernel methods)

Low-dimensional data

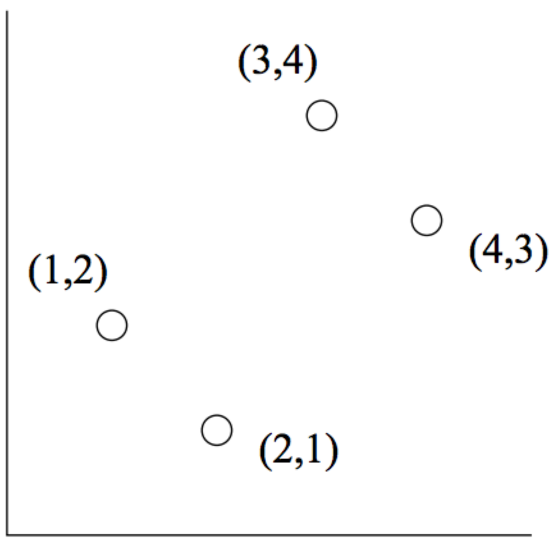
- Coarse-grained (simplified or compressed) descriptions
- Easy to interpret and visualize
- Noise filtered
- Better generalization (possible applications in transfer learning)
- Reduced time and space complexity

How to reduce the dimensionality of data intuitively?

- Remove data columns with small variances
- Reducing highly correlated columns

Principle Component Analysis (PCA)

An illustrative example



$$M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \qquad M^{\mathrm{T}}M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix}$$

Eigenvalue eqn. $(30 - \lambda)(30 - \lambda) - 28 \times 28 = 0$

The solution is $\lambda = 58$ and $\lambda = 2$.

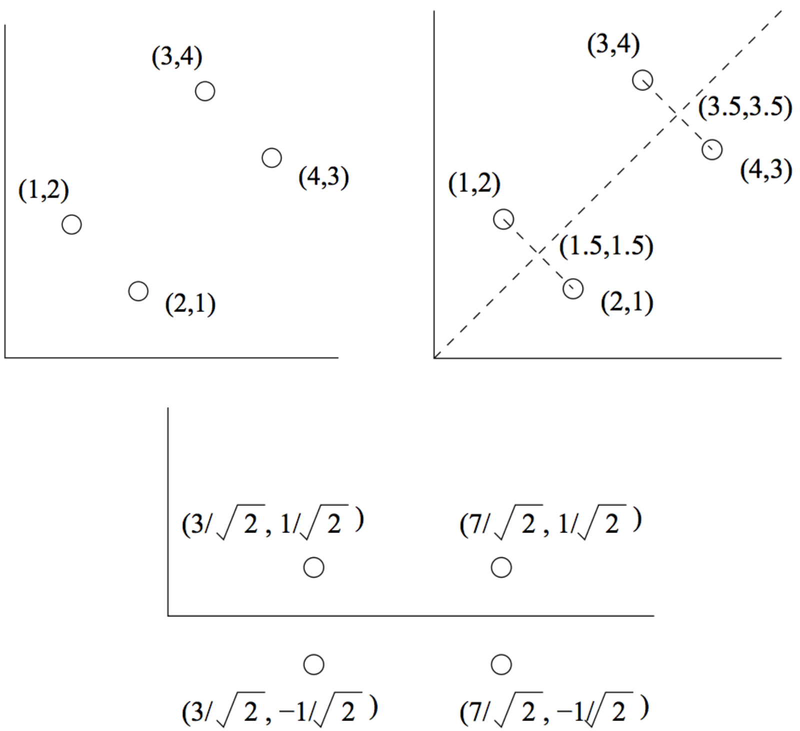
$$\begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 58 \begin{bmatrix} x \\ y \end{bmatrix} \quad x = y. \quad \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

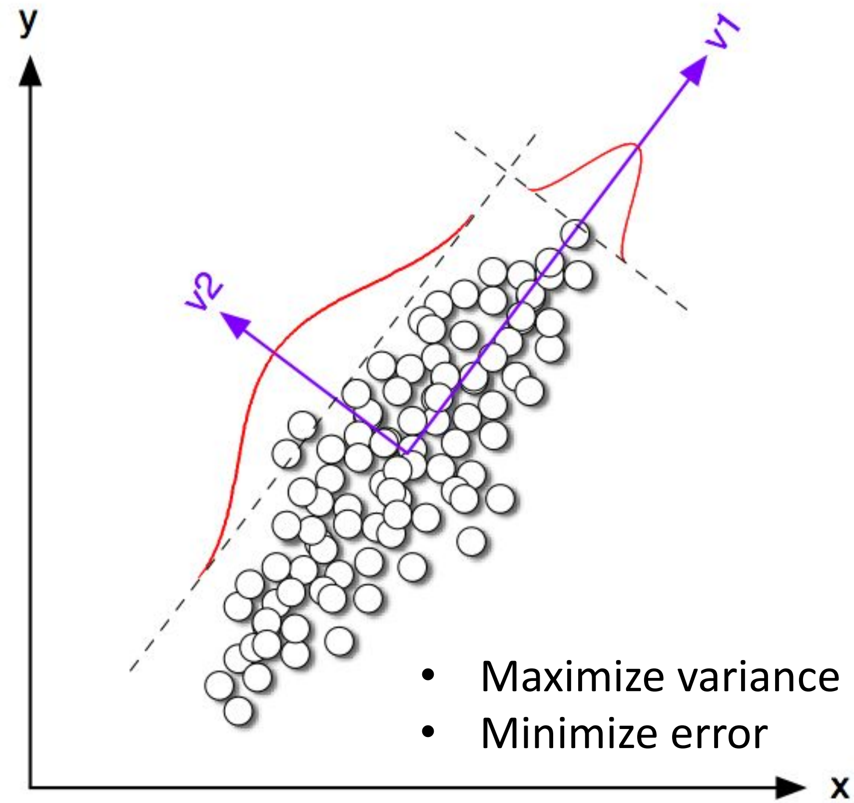
$$\begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 2 \begin{bmatrix} x \\ y \end{bmatrix} \quad x = -y. \quad \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$$

$$E = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \qquad ME = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 1/\sqrt{2} \\ 3/\sqrt{2} & -1/\sqrt{2} \\ 7/\sqrt{2} & 1/\sqrt{2} \\ 7/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Principle Component Analysis (PCA)

An illustrative example





- Maximize variance
- Minimize error

Variance Maximization

$E((\mathbf{u}\cdot\mathbf{x})^2) = E((\mathbf{u}\cdot\mathbf{x})(\mathbf{u}\cdot\mathbf{x})^T) = E(\mathbf{u}\cdot\mathbf{x}\cdot\mathbf{x}^T\cdot\mathbf{u}^T)$

The matrix $\mathbf{C} = \mathbf{x}\cdot\mathbf{x}^T$ contains the correlations (similarities) of the original axes based on how the data values project onto them

So we are looking for w that maximizes $\mathbf{uCu}^T$, subject to $\mathbf{u}$ being unit-length

Maximize $\mathbf{u}^T\mathbf{xx}^T\mathbf{u}$ s.t $\mathbf{u}^T\mathbf{u} = 1$

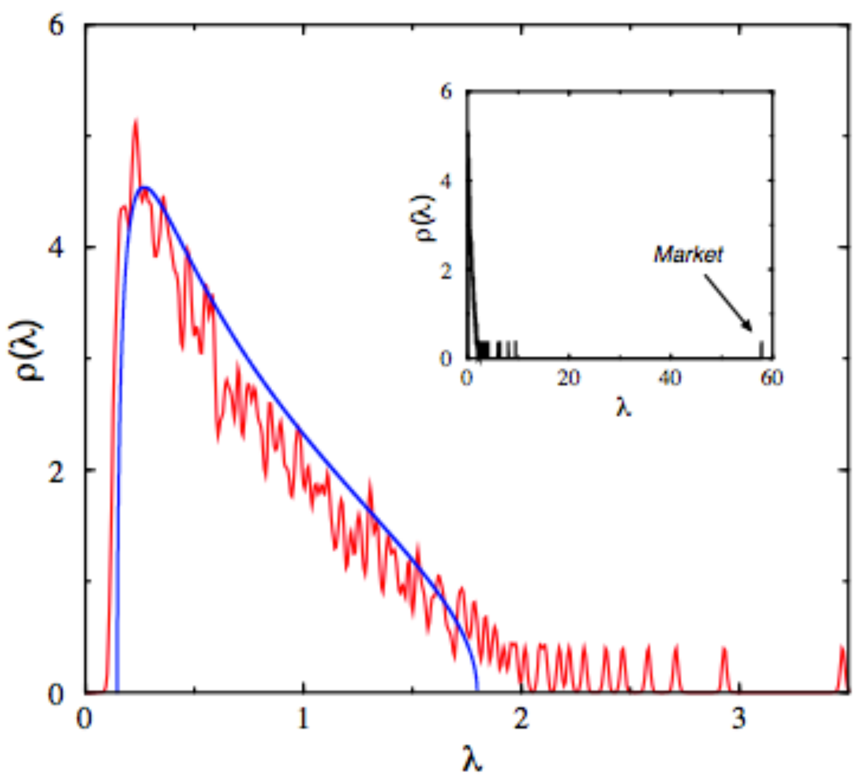
Construct Langrangian $\mathbf{u}^T\mathbf{xx}^T\mathbf{u} - \lambda\mathbf{u}^T\mathbf{u}$

Vector of partial derivatives set to zero

$$\mathbf{xx}^T\mathbf{u} - \lambda\mathbf{u} = (\mathbf{xx}^T - \lambda\mathbf{I})\,\mathbf{u} = 0$$

As $\mathbf{u} \neq \mathbf{0}$ then $\mathbf{u}$ must be an eigenvector of $\mathbf{xx}^T$ with eigenvalue λ

Eigenvalue Distribution



Eigenvalue distribution of a random matrix

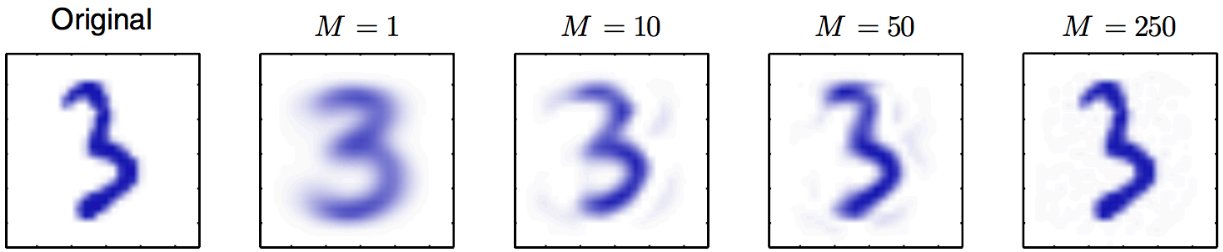
$$\rho(\lambda) = \frac{T}{N} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{2\pi\lambda} \quad \text{if} \quad \lambda_+ \leq \lambda \leq \lambda_-$$

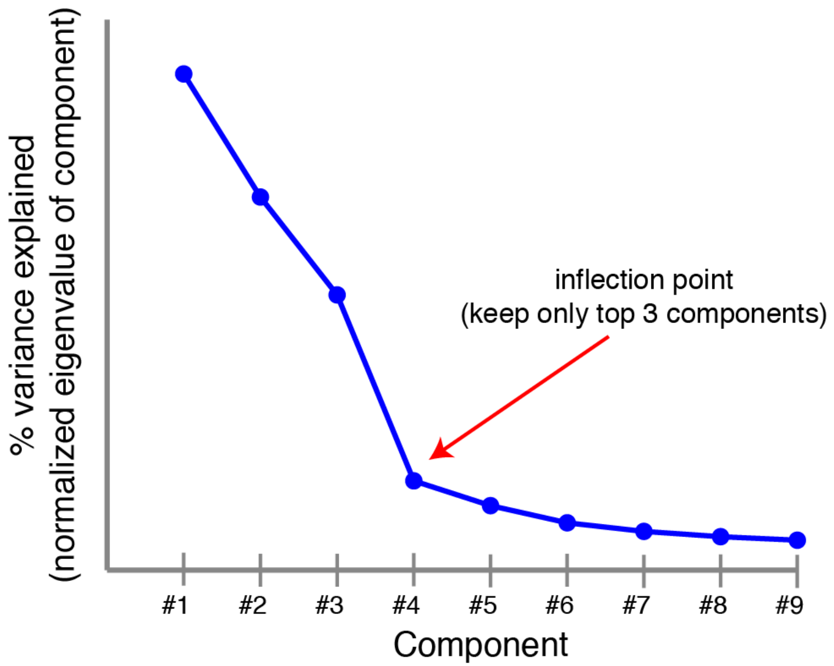
$$\lambda_\pm = \left[1 \pm \sqrt{\frac{N}{T}}\right]^2.$$

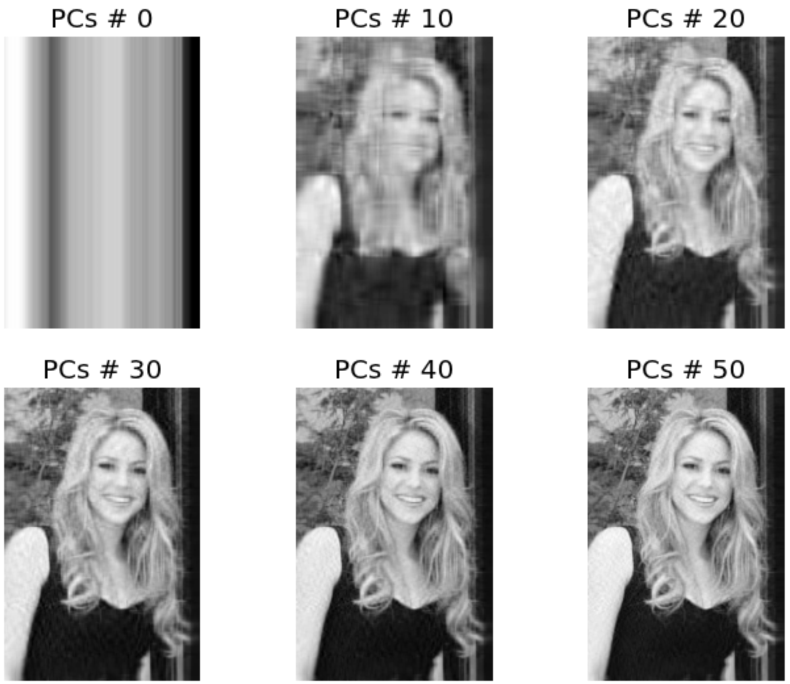
Empirical eigenvalue density for 406 stocks from the S&P 500, and fit using the MP distribution. Note the presence of one large eigenvalue corresponding to the market mode.

(Picture taken from Financial Applications of Random Matrix Theory: Old Laces and New Pieces, Potters et al.)

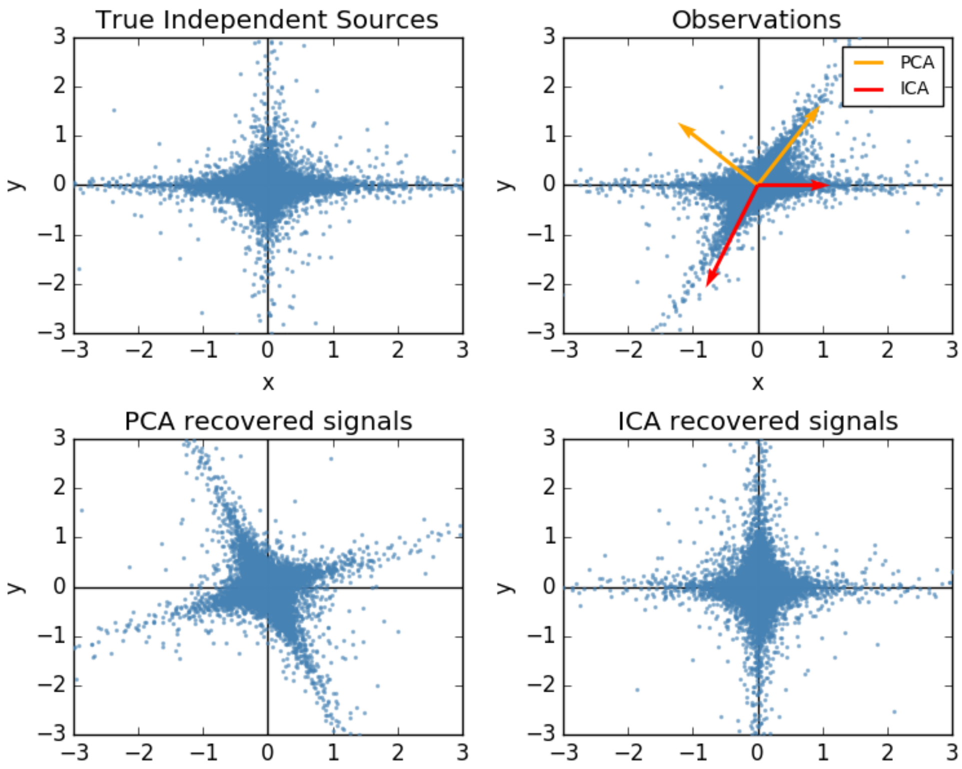
Examples of PCA





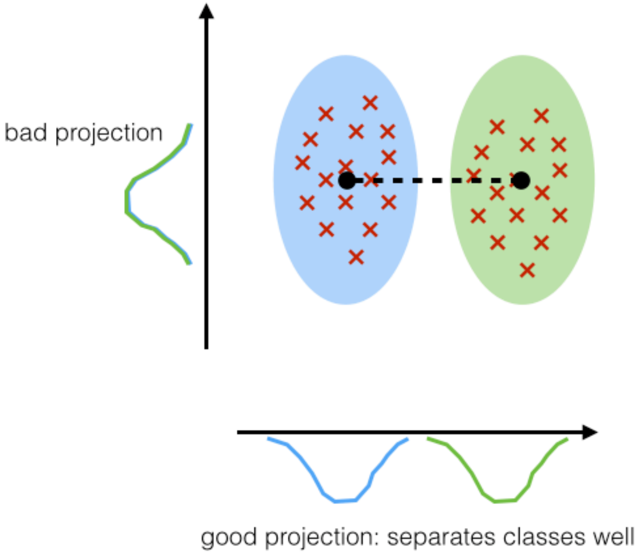


The limits of PCA (1)



Independent Component Analysis (ICA)

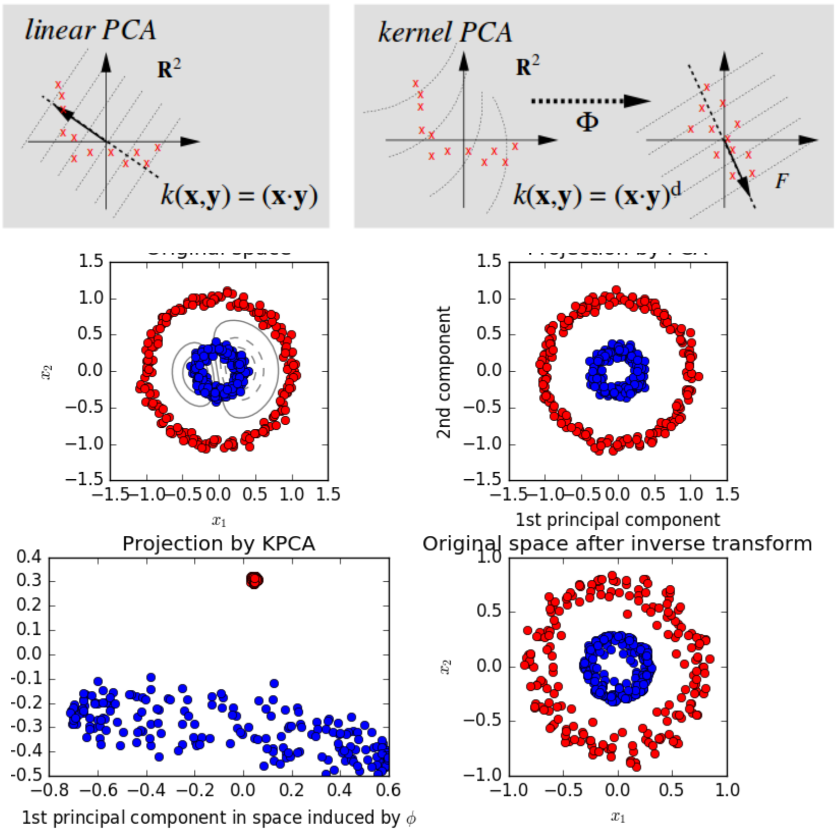
maximizing the component axes for class-separation



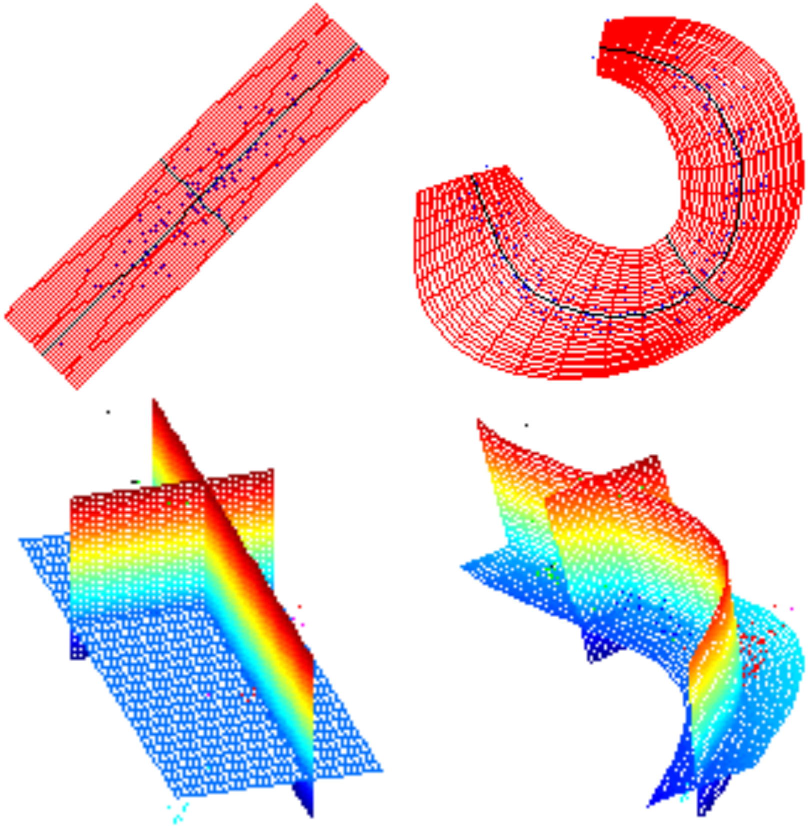
Supervised learning

Linear discriminant analysis (LDA)

The limits of PCA (2)

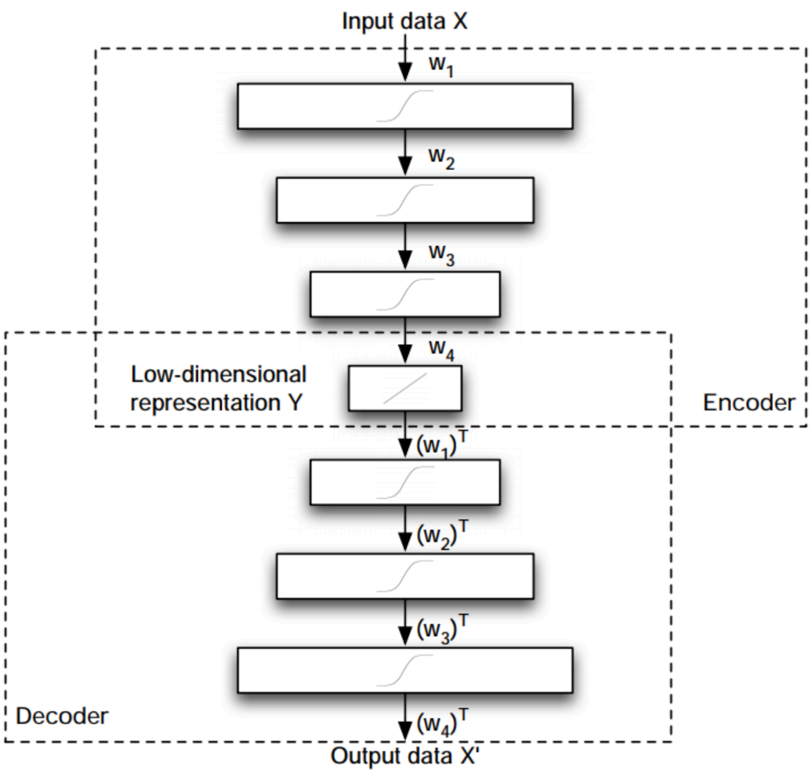


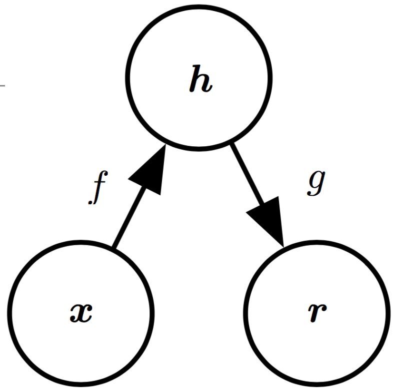
Kernel PCA



Nonlinear PCA

Autoencoders





- **Basic setups**
  - Input x --- encoder --- decoder --- output r
  - Encoder: h = f(x)
  - Decoder: r = g(h) = g(f(x))
  - Minimizing loss function: L(x, g(f(x))

- **Discussion**
  - When will an autoencoder give the same result as PCA? (Error function)
  - How to train a autoencoder? (gradient descent)

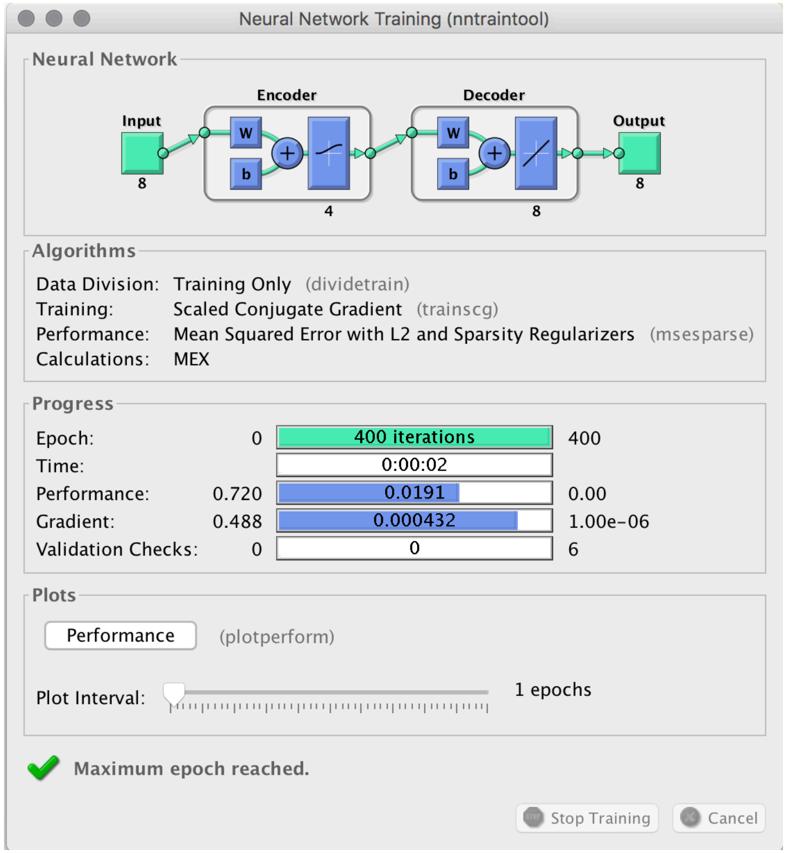
Your First Autoencoder

openExample('nnet/TrainAutoencoderWithSpecifiedOptionsExample')

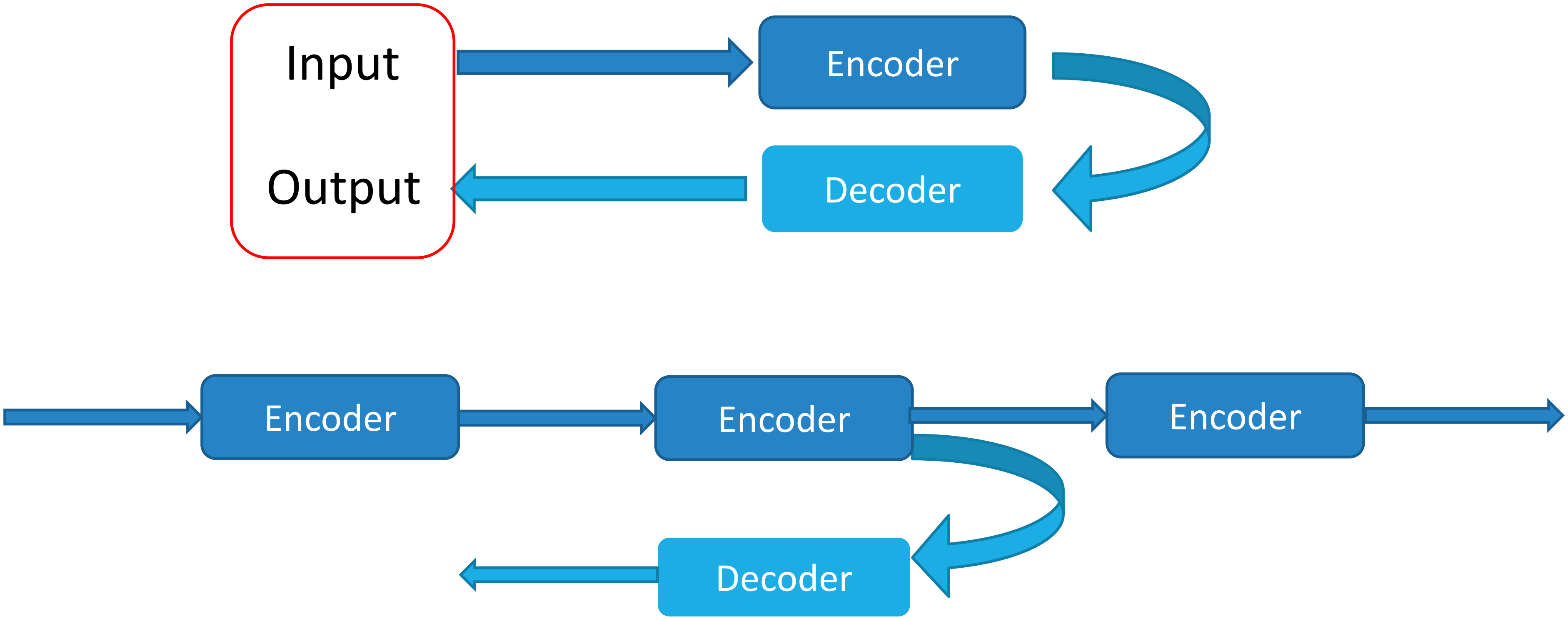
```
%% Train Autoencoder with Specified Options
% Load the sample data.

% Copyright 2015 The MathWorks, Inc.

X = abalone_dataset;
%%
% |X| is an 8-by-4177 matrix defining eight attributes for 4177 different
% abalone shells: sex (M, F, and I (for infant)), length, diameter, height,
% whole weight, shucked weight, viscera weight, shell weight. For more
% information on the dataset, type |help abalone_dataset| in the command
% line.
%%
% Train a sparse autoencoder with hidden size 4, 400 maximum epochs, and
% linear transfer function for the decoder.
autoenc = trainAutoencoder(X,4,'MaxEpochs',400,...
'DecoderTransferFunction','purelin');
%%
% Reconstruct the abalone shell ring data using the trained autoencoder.
XReconstructed = predict(autoenc,X);
%%
% Compute the mean squared reconstruction error.
mseError = mse(X-XReconstructed)
```



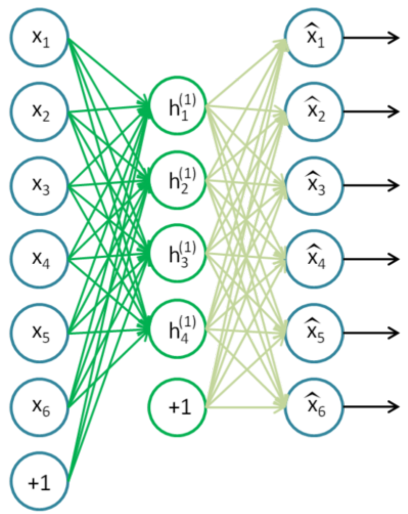
Stacked Autoencoders



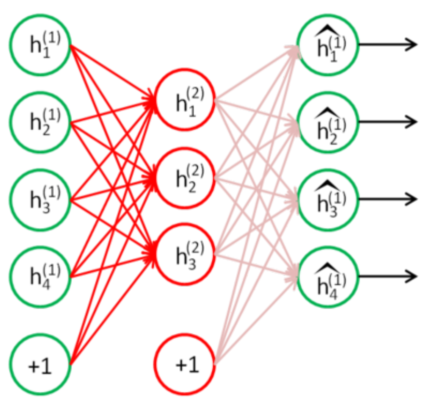
Greedy layer-wise training: trains the parameters of each layer individually while freezing parameters for the remainder of the model.

Stacked Autoencoders and Deep Learning

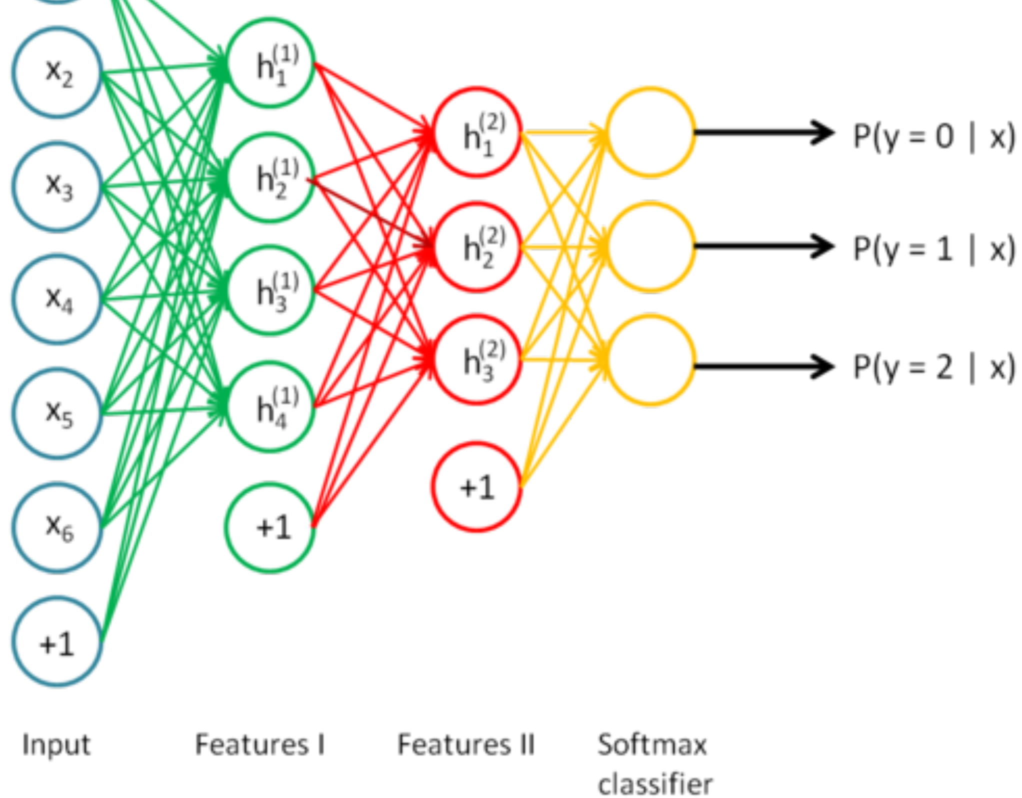
Train 1st Layer



Train 2nd Layer



- Unsupervised pretraining
- Supervised learning



http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders