Introdução a Sistemas Inteligentes

Tópicos em Redes Neurais I: Aprendizado Hebbiano de Rede Auto-Organizada para Análise de Componentes Principais (PCA)

Prof. Ricardo J. G. B. Campello

ICMC / USP

Aula de Hoje

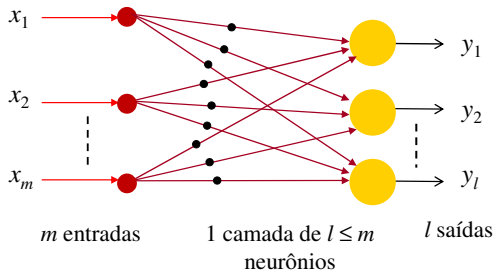
- Motivação para PCA
- Redes Neurais para PCA
  - Algoritmo Hebbiano
- Exemplos
  - Exemplo pedagógico
  - Exemplo de aplicação (compactação de imagens)

Motivação (PCA)

- No quadro...

Redes Neurais para PCA

- RNA com camada única e neurônios lineares é capaz de se **auto-organizar** para extrair as componentes principais:



Redes Neurais para PCA

- As componentes principais y_1, ..., y_l representam uma transformação do padrão de entrada x_1, ..., x_m tal que:
  - a entrada é decomposta de forma não redundante (ortogonal) nas comp. principais, que carregam informação complementar
  - a "porção de informação" da entrada em cada componente principal é decrescente da 1ª para a última componente
  - se reconstruirmos o padrão de entrada utilizando apenas as l primeiras componentes, onde $l < m$, o erro de reconstrução é minimizado (será nulo se $l = m$)

Redes Neurais para PCA

Saídas da rede são descritas individualmente por:

$$y_k = \sum_{i=1}^{m} w_{ki} x_i$$

ou em conjunto (forma vetorial) por:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \Rightarrow \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_l \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \ddots & \vdots \\ w_{l1} & \cdots & w_{lm} \end{bmatrix}$$

Redes Neurais para PCA

Entradas podem ser recuperadas (decodificadas) utilizando os pesos armazenados na rede:

$$\hat{\mathbf{x}} = \sum_{k=1}^{l} y_k \mathbf{w}_k^T$$

onde $\mathbf{w}_k$ é a j-ésima linha da matriz de pesos **W**

Algoritmo Hebbiano Generalizado (GHA)

Inicialize a matriz de pesos **W** com valores aleatórios ≈ 0

Repita para cada padrão de entrada **x**, até a convergência:

Calcule as saídas da rede para $k = 1, ..., l$:

$$y_k = \sum_{i=1}^{m} w_{ki} x_i$$

Atualize os pesos da rede para $k = 1, ..., l$ e $i = 1, ..., m$:

$$\Delta w_{ki} = y_k x_i - y_k \sum_{j=1}^{k} w_{ji} y_j$$

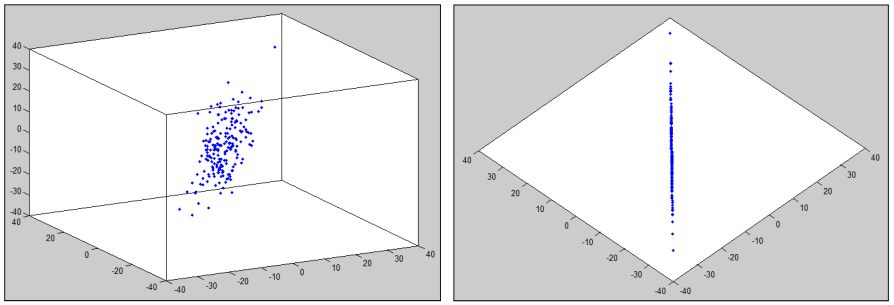
$$w_{ki} = w_{ki} + \eta \Delta w_{ki}$$

Algoritmo Hebbiano Generalizado (GHA)

- A cada iteração do algoritmo, um padrão **x** diferente é apresentado à entrada da rede e um passo de atualização dos pesos é executado
- Se os pesos não tiverem convergido após todos os padrões de treinamento disponíveis tiverem sido apresentados, reapresentam-se os padrões
  - Algoritmo converge se η for uma constante positiva "suficientemente pequena" (depende da aplicação)

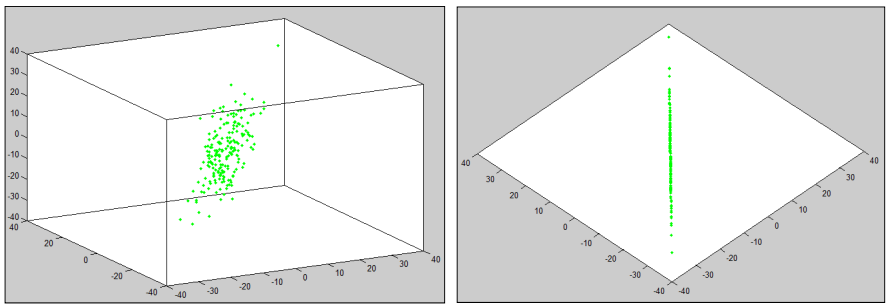
Exemplo 1

- 200 vetores 3D ($m = 3$) com distribuição Normal 2D ao longo de um plano no espaço original:



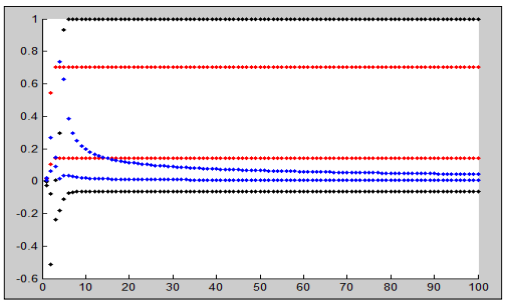
Exemplo 1 (cont.)

- PCA Neural com $l = 2$, η = 10^{-4} e 100 passagens completas pelos 200 objetos
- Dados recuperados:



Exemplo 1 (cont.)

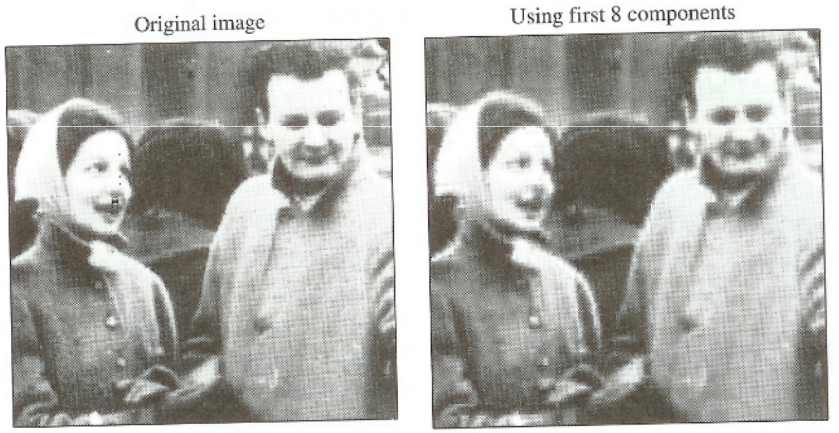
- Recuperação quase perfeita
  - Erro quadrático médio de recuperação: MSE = 0,6
  - Redução de dimensionalidade dos dados: 3D $\rightarrow$ 2D
- Pesos:



Exemplo 2 (Haykin, 1999)

- Compressão de uma imagem 256 x 256 (65.536) pixels com 256 tons de cinza (0 a 255)
- 1024 blocos de 64 pixels (8 x 8 pixels)
- Cada bloco = vetor com 64 entradas (0 a 255)
  - Logo, 1024 vetores para treinar a rede
- Rede com $m = 64$ entradas e $l = 8$ neurônios / saídas

Exemplo 2 (Haykin, 1999)



Exemplo 2 (Haykin, 1999)

- Outra imagem compactada e recuperada com as 8 primeiras componentes principais obtidas a partir da mesma rede treinada com a figura anterior

original

recuperada (PCA anterior)

Exemplo 3

Dimensions = 206

Principais Aplicações PCA

- Redução de Dimensionalidade (Extração de Características)
  - para algoritmos de aprendizado de máquina, reconhecimento de padrões e data mining
  - p. ex., decomposição espectral com dezenas ou centenas de componentes pode gerar umas poucas características relevantes
- Compressão
  - vide exemplos 2 e 3 anteriores

Exercício

- Seja o conjunto de apenas 2 padrões ($m = 3$):
  - $\mathbf{x}_1 = [10 \;\; 10 \;\; 10]^T$
  - $\mathbf{x}_2 = [-10 \;\; -10 \;\; -10]^T$

  e a seguinte matriz de pesos iniciais:

  $$\mathbf{W} = \begin{bmatrix} -0.03 & -0.03 & -0.03 \\ -0.01 & -0.01 & -0.01 \\ 0.02 & 0.02 & 0.02 \end{bmatrix}$$
- Faça $l = 3$, $\eta = 10^{-3}$ e complete a 1ª passagem do algoritmo GHA pelos dados (iniciada no slide a seguir com o padrão $\mathbf{x}_1$) apresentando o padrão $\mathbf{x}_2$ à rede

Exercício

$\mathbf{x}_1$	y	$\Delta\mathbf{W}(:,1)$	$\Delta\mathbf{W}(:,2)$	$\Delta\mathbf{W}(:,3)$	$\mathbf{W}(:,1)$	$\mathbf{W}(:,2)$	$\mathbf{W}(:,3)$
10	-0.9	-8.9757	-8.9757	-8.9757	-0.0389	-0.0389	-0.0389
10	-0.3	-2.9910	-2.9910	-2.9910	-0.0129	-0.0129	-0.0129
10	0.6	5.9748	5.9748	5.9748	0.0259	0.0259	0.0259

- Em seguida, faça também manualmente outra passagem completa de atualização dos pesos com os dois padrões
- Complete o exercício computacionalmente executando um total de 2000 passagens. Avalie o que ocorre com as 3 componentes principais e discuta os resultados

Referências

- Haykin, S. "Neural Networks: A Comprehensive Foundation", Prentice Hall, 2nd Edition, 1999*

* Possuem versão em Português