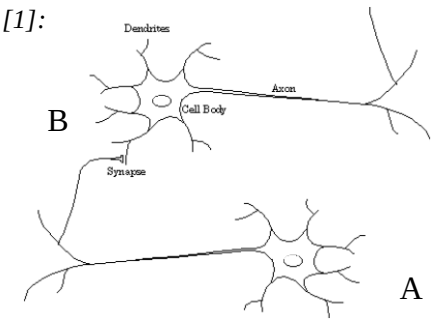
07. Aprendizagem de Hebb e Modelos Auto-Associativos

IC 22/23
ISEC - CPereira

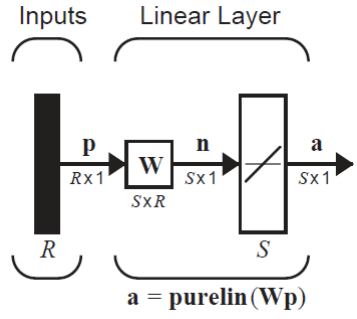
Postulado de Hebb

"**if two interconnected neurons are both "on" at the same time, then the weight between them should be increased**."
Neurons that fire together wire together

Neurónio biológico [1]:



Arquitetura – “Linear associator”



$$\mathbf{a} = \mathbf{W}\mathbf{p}$$

Regra de Hebb

$$w_{ij}^{new} = w_{ij}^{old} + \alpha\, f_i(a_{iq})\, g_j(p_{jq})$$

Sinal pré-sináptico

Sinal Pós-sináptico

Regra de Hebb

Forma Simplificada:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq}$$

Forma Supervisonada:

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}$$

Forma Matricial:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$$

Operação em Batch

$$\mathbf{W} = \mathbf{t}_1\mathbf{p}_1^T + \mathbf{t}_2\mathbf{p}_2^T + \cdots + \mathbf{t}_Q\mathbf{p}_Q^T = \sum_{q=1}^{Q} \mathbf{t}_q\mathbf{p}_q^T \quad \text{(Pesos iniciais nulos)}$$

Forma matricial:

$$\mathbf{W} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T}\mathbf{P}^T$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_Q \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_Q \end{bmatrix}$$

Análise de Desempenho

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left(\sum_{q=1}^{Q} \mathbf{t}_q \mathbf{p}_q^T\right)\mathbf{p}_k = \sum_{q=1}^{Q} \mathbf{t}_q(\mathbf{p}_q^T\mathbf{p}_k)$$

Caso I, Padrões de entrada ortogonais.

$$(\mathbf{p}_q^T\mathbf{p}_k) = 1 \qquad q = k$$
$$= 0 \qquad q \neq k$$

Assim, a saída iguala o valor desejado:

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k$$

Análise de Desempenho

Caso II, entradas normalizadas mas não ortogonais.

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \sum_{q \neq k} \mathbf{t}_q(\mathbf{p}_q^T\mathbf{p}_k)$$

Erro

Exemplo

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Protótipos Normalizados

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} -1 \end{bmatrix} \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 1 \end{bmatrix} \right\}$$

Exemplo

Matriz de Pesos (Regra de Hebb):

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -0.5774 & 0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix}$$

Exemplo

Teste:

$$\mathbf{W}\mathbf{p}_1 = \begin{bmatrix}1.1548 & 0 & 0\end{bmatrix}\begin{bmatrix}-0.5774\\ 0.5774\\ -0.5774\end{bmatrix} = \begin{bmatrix}-0.6668\end{bmatrix}$$

$$\mathbf{W}\mathbf{p}_2 = \begin{bmatrix}0 & 1.1548 & 0\end{bmatrix}\begin{bmatrix}0.5774\\ 0.5774\\ -0.5774\end{bmatrix} = \begin{bmatrix}0.6668\end{bmatrix}$$

Pseudoinversa

Índice de desempenho:

$$\mathbf{W}\mathbf{p}_q = \mathbf{t}_q \qquad q = 1, 2, \ldots, Q$$

$$F(\mathbf{W}) = \sum_{q=1}^{Q} \|\mathbf{t}_q - \mathbf{W}\mathbf{p}_q\|^2$$

Pseudoinversa

Forma Matricial: $\mathbf{W}\mathbf{P} = \mathbf{T}$

$$\mathbf{T} = \left[\mathbf{t}_1\ \mathbf{t}_2\ \cdots\ \mathbf{t}_Q\right]\quad \mathbf{P} = \left[\mathbf{p}_1\ \mathbf{p}_2\ \cdots\ \mathbf{p}_Q\right]$$

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

$$\|\mathbf{E}\|^2 = \sum_i \sum_j e_{ij}^2$$

Pseudoinversa

$$\mathbf{W}\mathbf{P} = \mathbf{T}$$

Minimizar:

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

Se existir inversa de $\mathbf{P}$, $F(\mathbf{W})$ pode ser nula para:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{-1}$$

PseudoInversa

Quando não existe inversa, $F(\mathbf{W})$ pode ser minimizada usando o método da pseudo-inversa:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+}$$

$$\mathbf{P}^{+} = (\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T}$$

Relação com Regra de Hebb

Regra de Hebb

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{T}$$

Pseudoinversa

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+}$$

$$\mathbf{P}^{+} = (\mathbf{P}^{T}\mathbf{P})^{-1}\mathbf{P}^{T}$$

Relação com Regra de Hebb

Se entradas ortonornais:

$$\mathbf{P}^T\mathbf{P} = \mathbf{I}$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \mathbf{P}^T$$

Exemplo

$$\left\{\mathbf{p}_1 = \begin{bmatrix}-1\\1\\-1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}-1\end{bmatrix}\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}1\end{bmatrix}\right\} \qquad \mathbf{W} = \mathbf{T}\mathbf{P}^+ = \begin{bmatrix}-1 & 1\end{bmatrix}\left(\begin{bmatrix}-1 & 1\\1 & 1\\-1 & -1\end{bmatrix}\right)^+$$

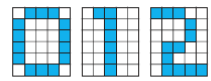
$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \begin{bmatrix}3 & 1\\1 & 3\end{bmatrix}^{-1}\begin{bmatrix}-1 & 1 & -1\\1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0.25 & -0.25\\0.5 & 0.25 & -0.25\end{bmatrix}$$

Exemplo

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{+} = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 0.25 & -0.25 \\ 0.5 & 0.25 & -0.25 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{W}\mathbf{p}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \end{bmatrix} \qquad \mathbf{W}\mathbf{p}_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix}$$

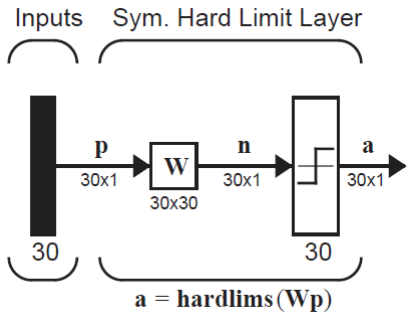
Memória Auto-associativa



$\mathbf{p}_1,\mathbf{t}_1$ $\mathbf{p}_2,\mathbf{t}_2$ $\mathbf{p}_3,\mathbf{t}_3$

$$\mathbf{p}_1 = \begin{bmatrix} -1 & 1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & \ldots & 1 & -1 \end{bmatrix}^T$$

Memória Auto-associativa

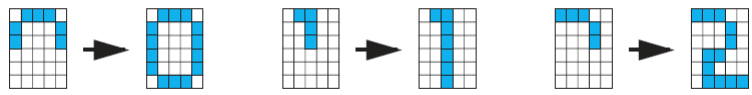


Pretende-se que “a” iguale “p”

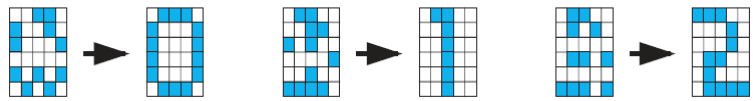
$$\mathbf{W} = \mathbf{p}_1\mathbf{p}_1^T + \mathbf{p}_2\mathbf{p}_2^T + \mathbf{p}_3\mathbf{p}_3^T$$

Memória Auto-associativa

50% ocultos
(ainda consegue recuperar o padrão desejado!)



Padrões com ruído (7 pixels)



exemplos em “*Neural network design demonstration*”
https://pypi.org/project/nndesigndemos/ [1]

Variações da Regra de Hebb

Básica : $\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$

Com *Learning Rate*: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T$

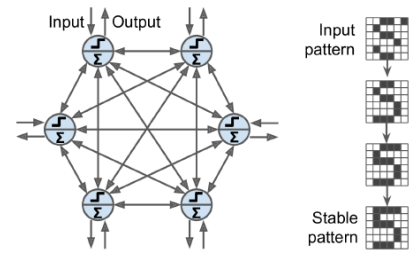
Smoothing: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1-\gamma)\mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T$

Regra Delta: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha(\mathbf{t}_q - \mathbf{a}_q)\mathbf{p}_q^T$

Não Supervisionada: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{a}_q \mathbf{p}_q^T$

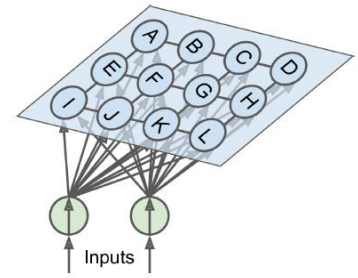
Outras Arquiteturas

- Redes de Hopfield
  - Redes de memória associativa:
    - Aprende com alguns padrões de exemplo (treino)
    - Quando se apresenta um novo padrão, produz o padrão aprendido (no treino) mais próximo.
  - Rede totalmente conectada
    - O algoritmo de treino baseia-se na regra de Hebb



Outras Arquiteturas

- Self-Organizing Maps
  - Usadas para produzir uma representação de baixa dimensão de um conjunto de dados de dimensão elevada.
  - Os neurónios são distribuídos num mapa - normalmente 2D para facilitar a visualização.
    - Cada neurónio tem uma ligação "pesada" para cada entrada.

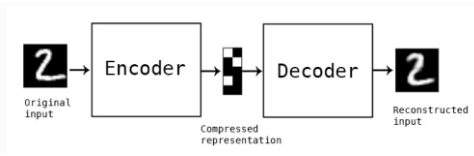


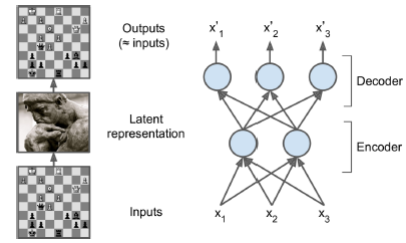
Outras Arquiteturas

- …
  - Treino
    - O algoritmo de treino é não supervisionado - faz com que todos os neurónios compitam entre si.
      - i) Os pesos são inicializados aleatoriamente.
      - Ii) É escolhida aleatoriamente uma instância de treino.
      - Iii) Todos os neurónios calculam a distância entre seu vetor de peso e o vetor de entrada,
      - iV) O neurónio de menor distância "ganha" e ajusta seu vetor de peso para ficar mais próximo do vetor de entrada, tornando mais provável que ganhe competições futuras
        - também pode recrutar neurónios vizinhos, que também atualizam os seus vetores de peso mas em menor escala

Outras Arquiteturas

- Auto-Encoders
  - Redes capazes de aprender representações densas dos dados de entrada, designadas de representações latentes ou codificações, sem qualquer supervisão.
  - As representações têm normalmente uma dimensionalidade muito inferior aos dados de entrada, tornando estes modelos úteis para redução de dimensionalidade.





- https://blog.keras.io/building-autoencoders-in-keras.html

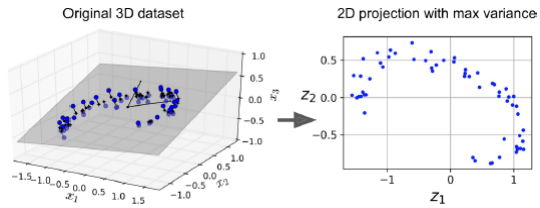
[2] Géron, A. (2019)

Outras Arquiteturas

- …
  - Realizam PCA se usarmos apenas funções de ativação lineares e função de custo MSE!

```
from tensorflow import keras
encoder = keras.models.Sequential([keras.layers.Dense(2, input_shape=[3])])
decoder = keras.models.Sequential([keras.layers.Dense(3, input_shape=[2])])
autoencoder = keras.models.Sequential([encoder, decoder])
autoencoder.compile(loss="mse", optimizer=keras.optimizers.SGD(lr=0.1))

history = autoencoder.fit(X_train, X_train, epochs=20)
codings = encoder.predict(X_train)
```



- *[2] Géron, A. (2019)*

Referências

- *[1] Neural Networks Design, Martin T. Hagan, Howard B. Demuth, Mark H. Beale, Cap. 7*
- *[2] Géron*, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.