**UNIVERSIDADE DE SÃO PAULO**
Instituto de Ciências Matemáticas e de Computação
Bacharelado em Sistemas de Informação

Introdução a Redes Neurais

# Trabalho 2

# Rede Neural Artificial Hopfield com aprendizado Hebbiano

Cainã de Souza D'Ajuda – 8531511

São Carlos – SP

# Introdução

O objetivo deste trabalho é a implementação de uma rede neural de Hopfield que memorize letras utilizando o aprendizado Hebbiano. Foi escolhida a rede de Hopfield discreta e com aprendizado sincronizado, que irá memorizar uma sequência inicial de padrões e não terá que adicionar novas memórias posteriormente.

Para a implementação desta rede neural artificial foi utilizada a linguagem de programação Python, na versão 3.4, com o auxílio das bibliotecas abertas NumPy[1] e Matplotlib[2]. A biblioteca Numpy foi utilizada para aprimorar cálculos de matriz e funções matemáticas gerais que foram aplicadas sobre os pesos e entradas no momento do aprendizado e lembrança. Já a biblioteca Matplotlib foi utilizada apenas para mostrar os padrões em formato de imagem e facilitar a visualização dos resultados e confirmação do aprendizado.

# Arquitetura

A rede de Hopfield consiste de um conjunto de neurônios atrelados a unidades de atraso, onde a saída de cada um é retro-alimentada em todos os outros neurônios, evitando assim a auto-retro-alimentação. Um exemplo deste tipo de rede é apresentado na Figura 1.

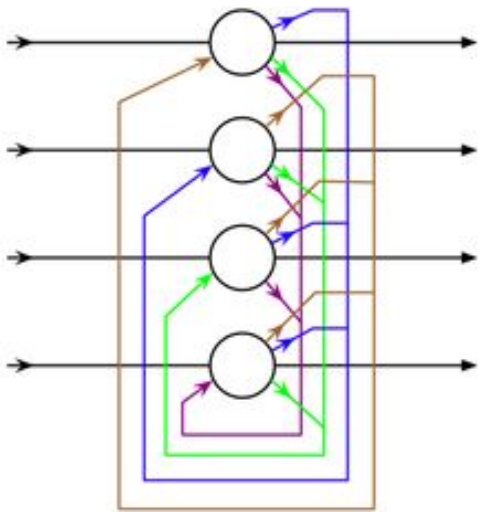

*Figura 1 Exemplo de uma rede de Hopfield*

1 http://www.numpy.org/
2 http://matplotlib.org/

## Treinamento e Resultados

Para o treinamento foram utilizadas representações de letras em matrizes 10x10. Foi testado empiricamente o número máximo de padrões a serem armazenados, resultando em um máximo de até quatro padrões memorizados perfeitamente. Acima de quatro a rede começa a confundir ou juntar padrões, resultando em memórias distorcidas. Para uma memorização melhor foram também utilizados padrões não muito próximos, assim um dos testes que teve melhor resultado foi utilizando as letras H, I, K e S. Para este conjunto de letras foi possível trocar 15 de 100 bits e a rede conseguiu reproduzir a letra por completo sem erros.

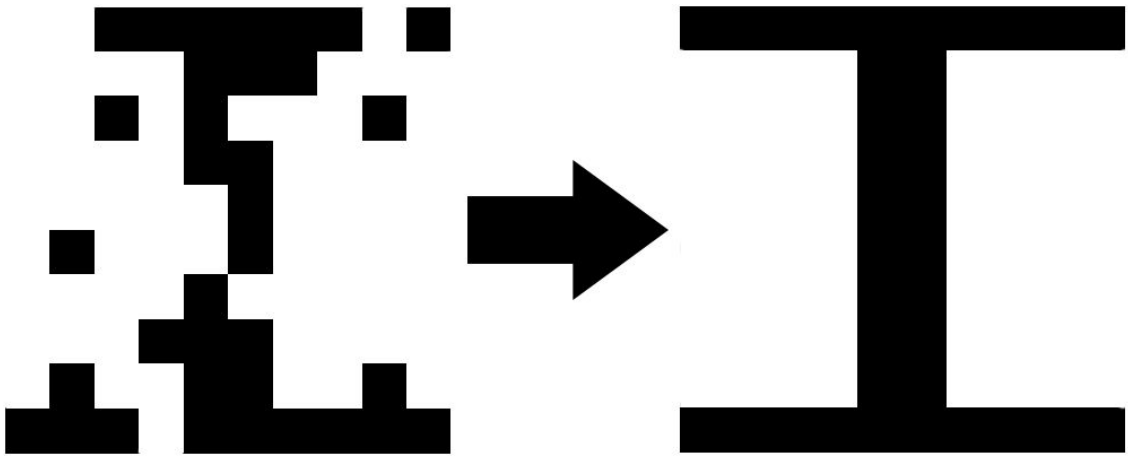

*Figura 2 Exemplo de entrada e saída da rede*

# Execução

**Dependências:**

**- python3-numpy**

**- python3-matplotlib**

Para a execução do trabalho primeiramente é necessário a instalação das dependências já informadas, os nomes podem variar dependendo da distribuição linux utilizada.

Para a execução, vá até o diretório que contenha os aquivos e execute o seguinte comando (via terminal):

***$ python3 main.py***

Posteriormente o software irá lhe informar três feedbacks, que consistem no estado que se encontra sua execução, sendo eles:

**- Treinamento**

**- Verificação de aprendizado**

**- Teste com ruído**

Após isso serão geradas os algumas imagens no diretório referentes aos testes. Essas imagens consistem em 4 tipos:

- **Padrão (‘padrao_X.png’)**

  - Imagens geradas a partir do vetor de característica usado como padrão de aprendizado

- **Verificação de aprendizado (‘lembranca_padrao_X.png’)**

  - Imagens geradas a partir das ‘lembranças’ dos padrões aprendidos pela rede. Essas imagens são utilizadas para verificar se a rede aprendeu corretamente.

- **Teste (‘teste_X.png’)**

  - Imagens com ruído para a realização dos testes.

- **Resultado (‘resultado_teste_X.png’)**

  - Imagens de resultado dos testes, essas imagens representam o padrão identificado pela rede para imagem com ruído.

Os resultados são identificados pelo numero do arquivo, por exemplo:

**teste_1.png → resultado_teste_1.png**