Entendendo e aplicando Inteligência Artificial – Parte I

Adriano Mendes

Sobre

Adriano Mendes

adriano.mendes.c@gmail.com

- Bacharel em Informática, UFJF, 1999
- Pós-graduado em Desenvolvimento de Sistemas - Machado Sobrinho JF, 2001
- MBA em Finanças- FGV, 2008
- CBA em Projetos – IBMEC, 2011
- Mestrado em Inteligência Computacional, UFJF,
  (em andamento)

Experiências

- Desenvolvimento de sistemas
- Gerente de consultoria e implantação
- Diretor de Tecnologia

Conhecimentos

- Sql Server, MySql
- Delphi, C#, Python, JavaScript
- Scikit Learn, TensorFlow, Keras
- Microsoft Azure
- PNL

Comentários

- **O Que é AI?**
  - Algorítimos “inteligentes”;
  - Algorítimos criados para resolver problemas que não tinham solução ou que eram complexos;
  - Fazem uso de
    - Eurísticas
    - Métodos estatísticos
    - Cálculos matemáticos

Comentários

- **Sobre AI:**
  - Um termo também usado em meio acadêmico é Computação Inteligente;
  - Teve seu avanço a medida que a capacidade de computação foi ampliada;
  - Podem se dividir nos métodos usadas pelos algorítimos
    - Algorítimos Genéticos, usando de métodos estocásticos;
    - Aprendizado de máquina, ML, usando de métodos determinísticos e outros;

Inteligência Artificial

Tipos de abordagens

- Estratégia que o algorítimo adota para “aprender”

Nosso foco

Supervisionados

- Usam dados preexistentes para “aprender”;
- Dados: Características + Classe
- Exemplos:
  - Arvores de decisão (D-Tree)
  - Regressões
  - SVM
  - Neural Networks

Não supervisionados

- Não necessitam de dados anteriores para “aprender”;
- Dados: Características
- Exemplos:
  - K-NN
  - K-Mens

Por Reforço

Inteligência Artificial

Tipos de problemas

Supervisionados

Classificação

- Doente ou sadio
- Fraude ou não
- Camisa, Short, Meia
- Detecção de objetos
- Reconhecimento de face

Regressão

- Valor de ação
- Valor do imóvel

Não supervisionados

Agrupamento

- Bons alunos
- Atletas
- Animais

Inteligência Artificial

- **Sobre os dados**
  - Serão definitivos para a qualidade do modelo
  - Devem passar por um processamento prévio para potencializar o aprendizado
  - A qualidade dos dados esta diretamente relacionada a qualidade final da aplicação de um algorítimo para resolver um problema

Inteligência Artificial

- **Sobre os algorítimos**
  - Hoje são vários algorítimos
  - Cada algorítimo tem características específicas
  - Em muitos casos, mais de um algorítimo pode ser usado para resolver um problema
  - Em alguns casos, são o resultado de múltiplas técnicas
  - No geral, para se criar uma aplicação de AI, compara-se algorítimos diferentes, e escolhe-se o que melhor resolve o problema

Inteligência Artificial

Fluxo de trabalho

1. Pré-processamento

- Tratamento dos dados
  - Normalização
  - Binarização
  - Tratamento de outliers
- Redução dos dados
- Extração de características

2. Desenvolvimento

- Criação do modelo
- Otimização do modelo
- Validação do modelo

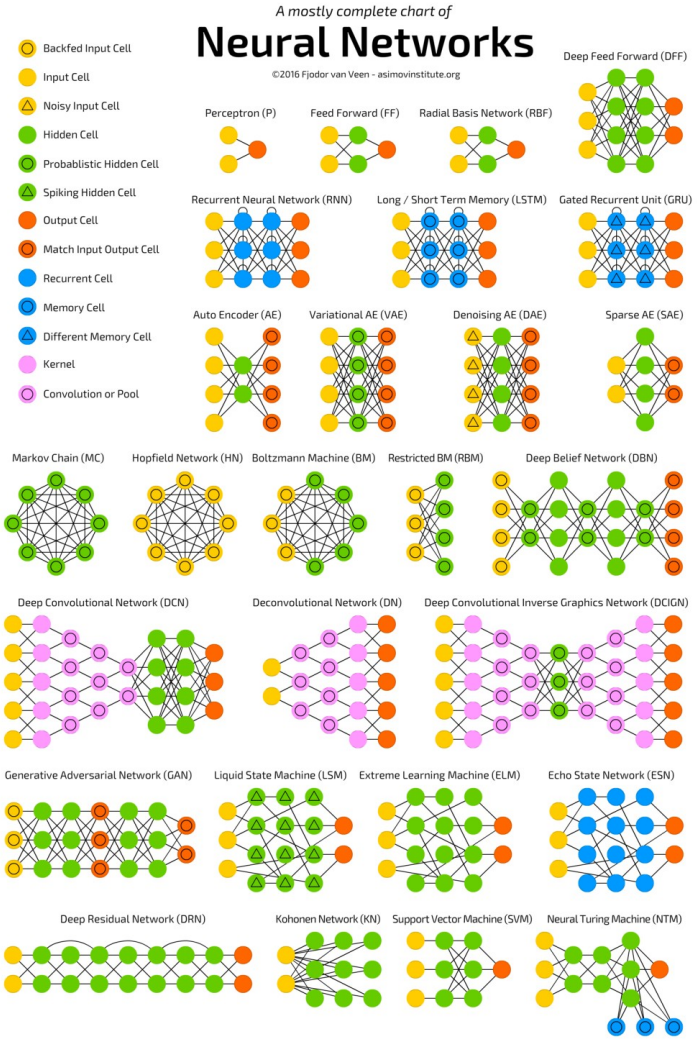
3. Integração e deploy

- Integração com sistemas
- Distribuição e uso
- Monitoramento

Inteligência Artificial

Algorítimos

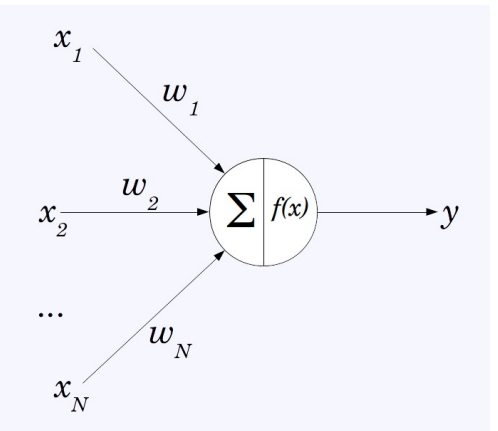
- Regressores
- SVM
- D-Trees
- Redes Neurais & Deep learning



Nota:
Confira mais detalhes em:
- https://www.asimovinstitute.org/neural-network-zoo/
- http://deeplearningbook.com.br/
- https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464
- https://www.digitalvidya.com/blog/types-of-neural-networks/

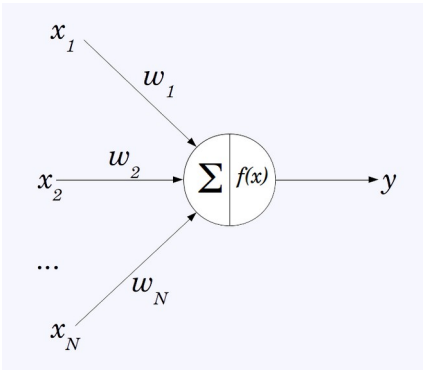
Perceptron

- O Modelo Perceptron, foi desenvolvido nas décadas de 1950 e 1960 pelo cientista Frank Rosenblatt, inspirado em trabalhos anteriores de Warren McCulloch e Walter Pitts.
- Também conhecido por neurônio matemático
- Hoje existem muitos outros modelos mais poderosos e este permite uma compreensão clara de como funciona uma RN em termos matemáticos.
- Trabalha somente com valores numéricos
- Representação:



Nota:
Confira mais detalhes em:
- https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975

Perceptron



PASSO 1: Inicialização
1) Define-se os pesos
2) Taxa de aprendizado, tx_{apr}
3) Define-se o threshold

PASSO 2: Para cada linha

1. Calcula somatório

2. Calcula f(x)
Função de ativação

3. Calcula Erro

4. Atualiza pesos

Repete até que o erro seja minimizado

Função de ativação

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Calculo de erro

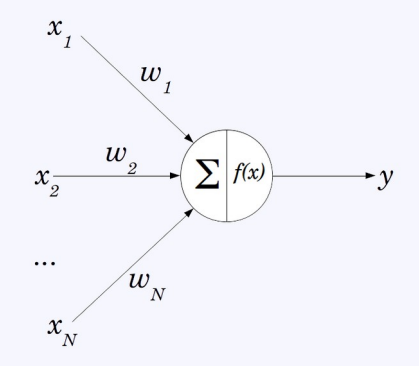
$e = Y_{esperado} - Y_{calculado}$

Atualização dos pesos

1) $w_{i(n+1)} = w_i + (tx_{apr} * Y_{esperado} * e)$

Perceptron

Linha	x1	x2	Classe
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1



1. Calcula somatório
w1, w2, soma produto

0,30	-0,10	0*0,3 + 0*-0,1 = 0
0,30	-0,10	0*0,3 + 1*-0,1 = -0,1
0,30	-0,10	1*0,3 + 0*-0,1 = 0,3
0,30	-0,10	1*0,3 + 1*-0,1 = 0,2

2. Calcula f(x) : Função de ativação

se 0>0,5 então 1 se não 0 => 0
se -0,1>0,5 então 1 se não 0 => 0
se 0,3>0,5 então 1 se não 0 => 0
se 0,2>0,5 então 1 se não 0 => 0

ÉPOCA: 1

3. Calcula Erro

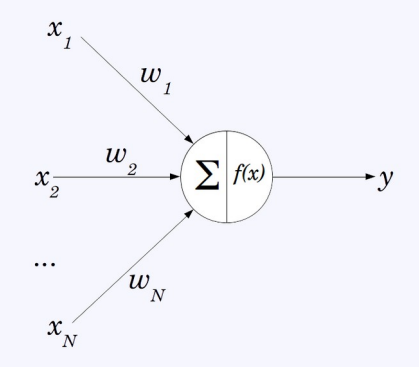
0 - 0 = 0
0 - 0 = 0
0 - 0 = 0
1 - 0 = 1

4. Atualiza pesos

w1(n+1) = 0,3 + (0 * 0,1 * 0) = 0,3	w2(n+1) = -0,1 + (0 * 0,1 * 0) = -0,1
w1(n+1) = 0,3 + (0 * 0,1 * 0) = 0,3	w2(n+1) = -0,1 + (1 * 0,1 * 0) = -0,1
w1(n+1) = 0,3 + (1 * 0,1 * 0) = 0,3	w2(n+1) = -0,1 + (0 * 0,1 * 0) = -0,1
w1(n+1) = 0,3 + (1 * 0,1 * 1) = 0,4	w2(n+1) = -0,1 + (1 * 0,1 * 1) = 0

Perceptron

Linha	x1	x2	Classe
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1



1. Calcula somatório
w1, w2, soma produto

0,40	0,00	0*0,4 + 0*0 = 0
0,40	0,00	0*0,4 + 1*0 = 0
0,40	0,00	1*0,4 + 0*0 = 0,4
0,40	0,00	1*0,4 + 1*0 = 0,4

2. Calcula f(x) : Função de ativação

se 0>0,5 então 1 se não 0 => 0
se 0>0,5 então 1 se não 0 => 0
se 0,4>0,5 então 1 se não 0 => 0
se 0,4>0,5 então 1 se não 0 => 0

ÉPOCA: 2

3. Calcula Erro

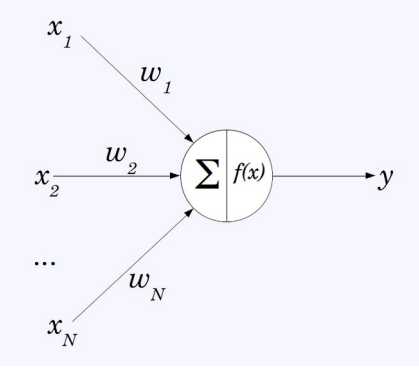
0 - 0 = 0
0 - 0 = 0
0 - 0 = 0
1 - 0 = 1

4. Atualiza pesos

w1(n+1) = 0,4 + (0 * 0,1 * 0) = 0,4	w2(n+1) = 0 + (0 * 0,1 * 0) = 0
w1(n+1) = 0,4 + (0 * 0,1 * 0) = 0,4	w2(n+1) = 0 + (1 * 0,1 * 0) = 0
w1(n+1) = 0,4 + (1 * 0,1 * 0) = 0,4	w2(n+1) = 0 + (0 * 0,1 * 0) = 0
⇒ w1(n+1) = 0,4 + (1 * 0,1 * 1) = 0,5	w2(n+1) = 0 + (1 * 0,1 * 1) = 0,1

Perceptron

Linha	x1	x2	Classe
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1



1. Calcula somatório
w1, w2, soma produto

0,50	0,10	0*0,5 + 0*0,1 = 0
0,50	0,10	0*0,5 + 1*0,1 = 0,1
0,50	0,10	1*0,5 + 0*0,1 = 0,5
0,50	0,10	1*0,5 + 1*0,1 = 0,6

2. Calcula f(x) : Função de ativação

se 0>0,5 então 1 se não 0 => 0
se 0,1>0,5 então 1 se não 0 => 0
se 0,5>0,5 então 1 se não 0 => 0
se 0,6>0,5 então 1 se não 0 => 1

ÉPOCA: 3

3. Calcula Erro

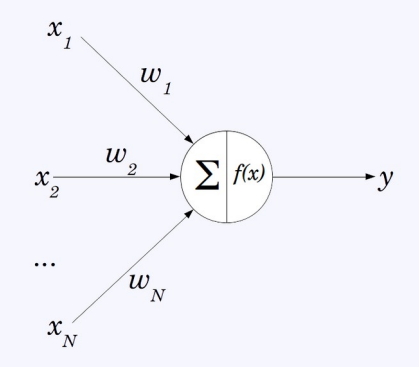
0 - 0 = 0
0 - 0 = 0
0 - 0 = 0
1 - 1 = 0

4. Atualiza pesos

w1(n+1) = 0,5 + (0 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (0 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (0 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (1 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (1 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (0 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (1 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (1 * 0,1 * 0) = 0,1

Perceptron

Linha	x1	x2	Classe
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1



1. Calcula somatório
w1, w2, soma produto

0,50	0,10	0*0,5 + 0*0,1 = 0
0,50	0,10	0*0,5 + 1*0,1 = 0,1
0,50	0,10	1*0,5 + 0*0,1 = 0,5
0,50	0,10	1*0,5 + 1*0,1 = 0,6

2. Calcula f(x) : Função de ativação

se 0>0,5 então 1 se não 0 => 0
se 0,1>0,5 então 1 se não 0 => 0
se 0,5>0,5 então 1 se não 0 => 0
se 0,6>0,5 então 1 se não 0 => 1

ÉPOCA: 4

3. Calcula Erro

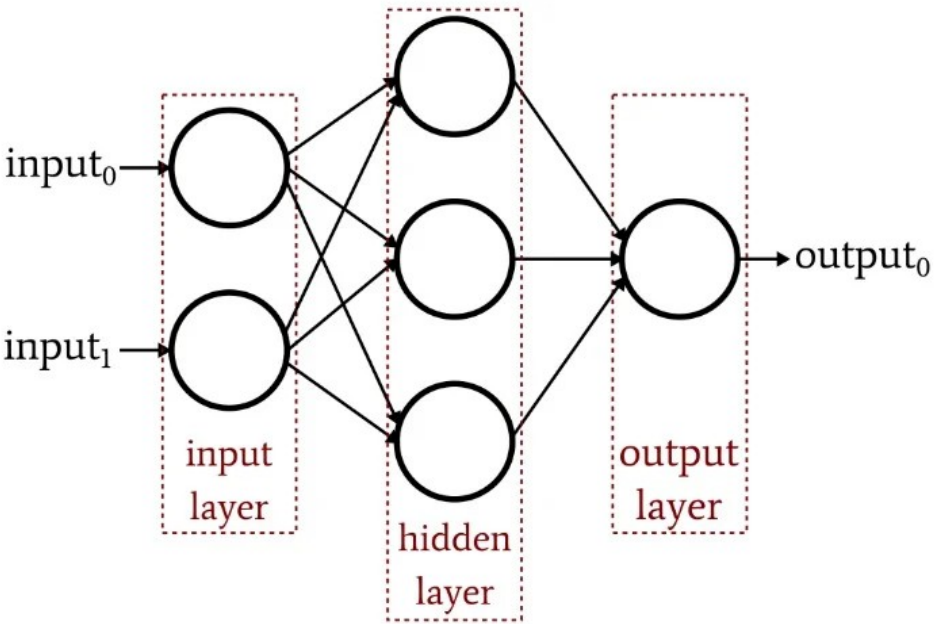
0 - 0 = 0
0 - 0 = 0
0 - 0 = 0
1 - 1 = 0

4. Atualiza pesos

w1(n+1) = 0,5 + (0 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (0 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (0 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (1 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (1 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (0 * 0,1 * 0) = 0,1
w1(n+1) = 0,5 + (1 * 0,1 * 0) = 0,5	w2(n+1) = 0,1 + (1 * 0,1 * 0) = 0,1

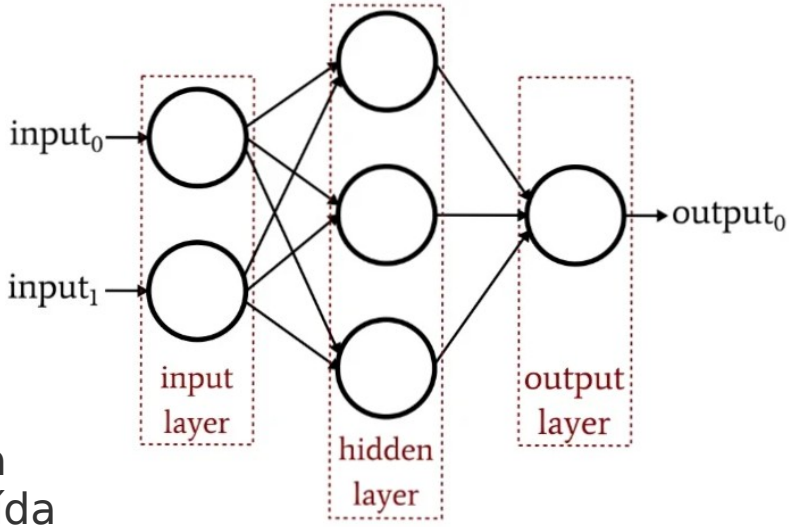
Multilayer Perceptron

- É uma das primeiras redes neurais
- É a união de vários perceptrons completamente conectados, camada a camada
- Representação:
  - Permitindo múltiplas camadas intermediárias
  - Permitindo múltiplos outputs



Nota:
Confira mais detalhes em:
- https://en.wikipedia.org/wiki/Multilayer_perceptron

Multilayer Perceptron



Passo 1: Inicialização
a. Atribuir valores aleatórios para os pesos e limites

Passo 2: Ativação
a. Calcular os valores dos neurônios da camada oculta
b. Calcular os valores dos neurônios da camada de saída

Passo 3: Treinar os Pesos
a. Calcular os erros dos neurônios das camadas de saída e oculta
b. Atualizar os pesos dos neurônios das camadas de saída e oculta

Passo 4: Iteração
a. Repetir o processo a partir do passo 2 até que satisfaça o critério de erro

Experimento 01

- Ideia
  - Realizar o reconhecimento de dígitos utilizando MLP
- Recursos
  - Linguagem Python[1] via distribuição Anaconda[2] e frameworks Scikit-Learn e Scikit-Image
  - Usando a base de dados MNIST[3]
  - Fazendo uso de Jupyter Notebooks usando a IDE do VS Code[4]
- Repositório:
  - GitHub: https://github.com/adrianomendesc/2020-SNCT-SEMPEX

(1) https://www.python.org/
(2) https://www.anaconda.com/
(3) http://yann.lecun.com/exdb/mnist/index.html ou https://www.openml.org/d/41063
(4) https://code.visualstudio.com/ e https://code.visualstudio.com/docs/python/data-science-tutorial

Referências

SITES

- Python.org
  https://www.python.org/https://www.python.org/
- Anaconda
  https://www.anaconda.com/
- OpenML.org
  https://www.openml.org/
- Data Science Academy
  https://www.datascienceacademy.com.br/pages/home
- Towards Data Science
  https://towardsdatascience.com
- IA Experts
  https://iaexpert.academy/

Referências

Livros

- Katia Faceli, **Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina** Gen, 2011
- Russell Stuart, **Artificial Intelligence, A modern approach**,3ª edição, Pearson, 2010
- Jake VanderPlas, **Python Data Science Handbook**, O'Reilly, 2016
- Joel Gruss, **Data Science from Scratch**, O'Reilly, 2015