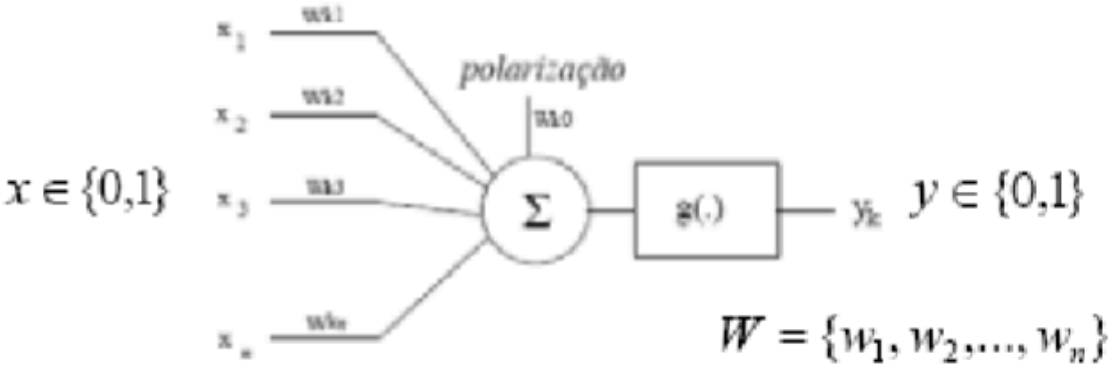
INTELIGÊNCIA ARTIFICIAL

Redes Neurais - 06/10

Estrutura do neurônio de McCulloch-Pitts:



- x_i é a excitação de entrada na sinapse i;
- y_k é a resposta (ou saída) do neurônio k;
- w_{ki} é o peso sináptico da entrada i do neurônio k;
- $g(.)$ é a função de ativação do neurônio.

A entrada é um conjunto de informações de um objeto analisado. O caso explicado na sala de aula foi de uma pessoa, então:

x1 = Gênero . Wg

x2 = Escolaridade . We

x3 = Idade . Wi

x4 = Nacionalidade . Wn

xbias = Bias . Wbias

Peso sináptico é o peso atrelado a cada “sinapse” da entrada.

- O peso diz a importância dos pesos para o resultado final de uma análise.

O somatório do neurônio é o **somatório do produtório.**

- Soma o peso atrelado a entrada x com o valor da entrada de fato.

Existem uma entrada padrão chamada de polarização **(BIAS)** que, normalmente vai ser 1, mas o seu peso que vai fazer a diferença, ou seja:

- Se tivermos 10 entradas, teremos 11 informações → 10 entradas + 1 polarização.

A saída dessa análise é binária, logo:

- Temos uma função de ativação após o somatório so neurônio ser calculado.
- Essa função auxiliar ativa um neurônio com base em uma condição. Por exemplo:
  - Se o somatório for > que 0 a saída é 1.
  - Se o somatório for < que 0 a saída é 0.

Perceptron:

Consirando a função AND:

- Ela é LINEARMENTE separável.
- Considerando que o peso de X é 0,2, de Y é -0,4 e do bias (valor 1) é 0,3:

  X | Y | X ^ Y | Σ | f

  0 | 0 | 0 | 0,3 | 1

  0 | 1 | 0 | |

  1 | 0 | 0 | |

  1 | 1 | 1 | |

  X ^ Y → É a saída esperada.

  Σ → É o somatório do produtório - entradas vezes os pesos.

  f → Saída encontrada

- Cálculos feitos:
  - Para **Σ:** X . Wx + Y . Wy + Bias . Wbias
    - No primeiro caso: 0 . 0,2 + 0 . -0,4 + 1 . 0,3 = 0,3.
    - No caso acima, pela função auxiliar - por ser maior que 0 - o resultado f = 1. O que está **ERRADO.**

    - Após o calculo acima, utilizamos a seguinte fórmula para ajustar esse erro:
        - Wt+1 = Wt + ta . erro . entrada
            - Wt+1 → Peso novo, após o ajuste.
            - Wt → Peso atual, sem o ajuste.
            - ta → Taxa de aprendizado (Será bem pequena para que o ajuste aconteça devagar)
            - erro → Saída esperada MENOS Saída encontrada.
            - entrada → Entrada que estamos ajustando.
        - Wf1 → primeira saída → Wf1 = 0,2 + 0,3 . (0 - 1) . 0
            - Wf1 = 0,2 + 0 → 0,2
                - Atualizamos o peso de X:
        - Repetimos esse passo a passo para Y e para o BIAS.
        - Atualizamos eles, se necessário.
- Em entradas que podem variar entre muitos valores - que são altos, por exemplo idade -, devemos fazer um “ajuste” nessas entradas para que os cálculos sejam feitos com valores na mesma escala. Isso busca evitar que uma entrada roube no resultado.

Redes Neurais - 08/10

XOR | XNOR → Não podem ser resolvidas por um perceptron simples, pois não são lineares.

Exercícios

Dado um perceptron simples de duas entradas e um bias, cujos pesos são w1 =0,5, w2 = 0,4 e w0 = - 0,3, respectivamente, assinalar a resposta correta:

(a) o perceptron realiza a função NOR

(b) o perceptron realiza a função AND

(c) o perceptron realiza a função OR

(d) o perceptron realiza a função XOR

(e) nenhuma das alternativas

R: (c) o perceptron realiza a função OR

X	Y	X ^ Y	Σ	f		O limiar é: > 0 = 1 | < 0 = 0
0	0	0	-0,3	0	→	0x0,5 + 0x0,4 + 1x-0,3 = -0,3 → 0
0	1	0	0,1	1	→	0x0,5 + 1×0,4 + 1x-0,3 = 0,1 → 1
1	0	0	0,2	1	→	1×0,5 + 0x0,4 + 1x-0,3 = 0,2 → 1
1	1	1	0,6	1	→	1×0,5 + 1×0,4 + 1x-0,3 = 0,6 → 1

MLP - Multilayer Perceptron

Algoritmo Backpropagation:

- Fase 1 - Propagação (fase forward):
  - Os perceptrons recebem cada entrada e realizam o cálculo do **somatório do produtório** de suas entradas e respectivos pesos - **NÃO** podemos esquecer do BIAS.
  - A função auxiliar **NÃO É MAIS A MESMA →** Agora é utilizado uma função **NÃO** linear.
  - As saídas de cada perceptron são passadas para os próximos perceptrons e o processo se repete até a camada da saída final, onde encontramos a resposta e onde o erro é calculado.
- Fase 2 - Retropropagação (fase backward):
  - Ao encontrar um erro, precisamos estimar a contribuição de cada perceptron para esse resultado.

- Começamos a estimar na camada de saída.
    - Se for trivial fazemos: **derivada(função de ativação) * (desejado - real).**
    - Se não for trivial, utilizamos a solução do próximo item.
- Essa é a fórmula mais usual que utilizamos para encontrar esse erro:

Ao invés de usar a diferença absoluta, pode-se utilizar também o **erro médio quadrático (Mean Square Error – MSE) ou Root Mean Square Error – RMSE)**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad \mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(S_i - O_i)^2}$$

Técnicas de Agrupamento - 27/10

Aprendizado Indutivo → Cria uma função e com base nela classifica os testes.

- Descritivo:
    - Associação.
    - Agrupamento.
- Preditivo:
    - Classificação.
    - Regressão.
    - O Random Forest, Backpropagation e Cart resolvem esse problema.

Tipos de agrupamento → Agrupamentos são formados com base em similidaridades

- Particional:
    - K-Means → Vamos ver o que cada indivíduo está em um único grupo.
        - Muito bom quando temos uma clara divisão de grupos.

- Complexidade: O(n*k*i*d)
    - N = Número de instâncias.

      K = Número de clusters (grupos).

      I = Número de iterações.

      D = Número de atributos.
- Existe o **nebuloso**, onde um indivíduo do teste pode participar de diversos grupos.
- Distâncias intra-clusters são minimizadas, já as entre clusters são maximizadas.
    - Distância de Manhattan.
- 1a interação:
    - Chuta os Ks (instâncias).
    - Calcula a distância entre cada ponto com os Ks escolhidos (centróides).
- 2a interação:
    - Calcula a média entre as distâncias das instâncias do grupo para cada atributo. Os centróides antigos participam do cálculo dessa média.
    - Marca os novos centróides como + em cada grupo.
    - Calcula a distância entre todas as instâncias aos novos centróides, criando um **NOVO** grupo.
- 3a interação e seguintes:
    - Refaz o passo 2.
    - O critério de parada é quando X% que estão mudando de grupo. “Até que somente 1% dos objetos mudam de clusters”, onde X = 1.

- Vamos pensar que utilizamos o K-Means e encontramos 3 grupos:
    - De fato essa base tem 3 grupos?
        - Não! Afinal, se ele encontrou 3, é porque foi definido K = 3.

        - Uma solução é utilizar esse algoritmo diversas vezes com diversos Ks, calculando a qualidade de cada uma dessas “respostas” e utilizando a melhor.
    - Como posso dar um nome ao grupo?
        - Roda uma árvore, gerando regras que ajudam a dar uma ideia sobre o nome do grupo.
- Hierárquico:
- Por densidade:
    - DB Scan.
- Baseado em modelo:
    - SOM.

12/11

Dados inconsistentes → São aqueles dados que possuem valores conflitantes em seus atributos.

- Duas instâncias com todos atributos iguais, mas com o resultado diferente.
    - Exemplo: Dois pacientes com todos os seus sintomas iguais, mas com doenças diagnosticadas diferentes.
- Quanto mais reduzimos a quantidade de atributos, maior a chance de termos dados inconsistentes.
- Retiramos todos os dados inconsistentes. Se temos duas instâncias em conflito, retiramos as duas.

Dados redundantes → São aqueles dados que possuem valores semelhantes em seus atributos.

- Uma instância é redundante quando ela é muito semelhante a uma outra instância do mesmo conjunto de dados.
- Podemos se referir a **instâncias** ou **atributos**.
    - Dados redundantes em relação ao atributo X.
    - Dados redundantes em relação a instâncias completas.

        - Exemplo: Pacientes com todos ou quase todos sintomas iguais, com diagnósticos iguais.
    - Retiramos todos os dados redundantes, exceto uma instância deles.
    - A redundância de um atributo está relacionada à sua **correlação** com um ou mais atributos do conjunto de dados.
        - Dois atributos estão correlacionados quando apresentam um perfil de variação semelhante.
            - Exemplo: Alunos que tem mais **horas de estudo** possuem maiores **notas**. (Atributos de entrada!)
            - Se a correlação ocorrer entre um **atributo de entrada** e um **atributo de saída,** o atributo de entrada terá uma **grande** influência na predição do valor do atributo de classe (saída).
                - Exemplo: Uma instância com o atributo de entrada “quebrou regra do tipo” está **muito** correlacionada ao atributo de saída “quebrou regra?”. Afinal, ter um valor positivo no de entrada, diz 100% sobre o valor do de saída.
- **Dados ausentes →** O ideal é **NÃO** ter dado ausente, mas na vida real **quase sempre** temos eles.
    - **Próxima aula.**

17/11

O que fazer com dados ausentes?

- Podemos eliminar as instâncias com dados ausentes, mas somente quando realmente é possível removê-la. **(Não é prática e não acontece muito no mundo real)**
    - Ao remover, podemos ficar com um conjunto muito pequeno, o que torna ruim essa opção.
- Podemos definir manualmente em alguns casos específicos. **(Não é prática e não acontece muito no mundo real)**
    - Não acontece muito, pois nem sempre é possível preencher os dados de um conjunto. “Forjá-los” nem sempre manterá a base interessante para trabalho.

- Podemos utilizar algum método ou heurística para automaticamente definir valores para os campos ausentes, mas isso pode deixar a nossa base distorcida. **(Tem problemas quanto a distorção da variação da base, pois, ao utilizar a média e mediana, a variável é "achatada", colocando instâncias em lugares indevidos)**
- Utilizar algoritmos de AM (Aprendizado de Máquina) que lidam internamente com dados ausentes.
    - Missforest
    - KNNimputer

- **Balanceamento →** É necessário balancer base antes de utilizá-las, pois pode acontecer de que uma classe seja completamente entendida, mas uma segunda classe tenha 100% de erro.
    - Se essa segunda classe tiver poucas instâncias comparadas a primeira classe, a acurácia geral vai ficar muito boa, mas esse modelo não vai ser útil pra nada.
- **Como resolver?**
    - Redefinir o tamanho do conjunto de dados.
        - Adicionar instâncias à classe minoritária. (Métodos **oversampling**)
            - Existe o risco de adicionar instâncias representem situações que nunca ocorrerão, induzindo um modelo inadequado.
            - Pode ocorrer o problema de **overfitting,** em que o modelo é superajustado em relação aos dados de treinamento.
            - Método **SMOTE.**
        - Remover instâncias da classe majoritária. (Métodos **undersampling)**
            - Ao remover essas instâncias da base da classe majoritária, é possível que dados de grande importância sejam perdidos, gerando um modelo errado.
            - Pode ocorrer o problemas de **underfitting,** em que o modelo é superajustado em relação aos dados de treinamento.
            - Método **RandomUnderSampler**.

        - **Under-over →** Reduzimos da majoritária e acrescentamos na minoritária, encontrando as duas classes no “meio do caminho”.
    - Utilizar diferentes custos de classificação para as diferentes classes.
    - Induzir um modelo para uma classe.

19/11

- Passo a passo para o pré-processamento:
    1. **Visualização** de dados (Seleção da base e compreensão da base):
        a. Nominal Normal (Até 10 “opções” nos atributos) → Barras
        b. Nominal de Alta Cardinalidade (Cidades, Países) → Mapa
        c. Numérica → Histograma
    2. **Seleção** dos atributos:
        a. Observamos a **Correlação** de atributos → Muitos atributos correlacionados não são necessários, como Data de Nascimento e Idade. Podemos remover alguns para simplificar o processo.
        b. Removemos aqueles atributos que não são relevantes para o objetivo, seja ele classificação, agrupamento, etc.
        c. Selecionamos aqueles atributos que são relevantes, seja de forma empírica (a partir de algum estudo na área, conhecimento de especialistas, a própria visualização dos dados, etc) ou utilizando algum método computacional, como a entropia.
    3. **Codificação:**
        a. Após codificar, é bom “olhar pra trás” para a etapa de visualização, pois é possível ter prejudicado a qualidade da nossa base.
        b. Label encoder.
        c. One hot encoder.
        d. Podemos imputar a discretização nesse momento.
    4. Eliminação de **Inconsistências e Redundâncias**.
    5. Eliminação de **Outlier:**
        a. Boxplot ou outros métodos.

    b. Atenção para os outliers extremos quando existem muitos outliers na base.

6. **TRAIN TEST SPLIT:**
    a. Divide a base para obter o conjunto de treino e o conjunto de treino.
    b. X% para teste.
7. **Imputação de dado ausente →** Fazemos no conjunto de treino e no teste**:**
    a. Utilizamos machine learning.
8. **Normalização/Padronização →** Fazemos no conjunto de treino e no teste.
9. **Balanceamento →** Fazemos apenas no conjunto de treino**:**
    a. Sem balanceamento.
    b. Com over.
    c. Com under.
    d. Com over-under.