EA072 – Inteligência Artificial em Aplicações Industriais (2s2024)

Exercícios Computacionais – ECs #10 a #13

Atividade Individual – Peso 3

Data de Entrega do Relatório (conjunto de notebooks IPYNB executados e com respostas para as questões levantadas): 04/12/2024

Miscelânia de técnicas de inteligência artificial

Atividade prática #10: Síntese e interpretação de uma rede bayesiana

Existem muitas opções de pacotes de software para se operar com redes bayesianas. Optamos aqui pelo pyAgrum:

https://pyagrum.readthedocs.io/en/1.10.0/notebooks/01-Tutorial.html

No ambiente Google Colab, carregue o notebook [**EC10_BN.ipynb**] e faça o upload, na área de trabalho, do arquivo [**dados_EC10.txt**]. Propositadamente, não está sendo passada nenhuma informação a priori acerca da natureza do conjunto de dados. Necessariamente, o conjunto de dados deve ser formado por variáveis discretas. Se houvesse variáveis contínuas, elas deveriam passar por algum processo de quantização / discretização, antes de sintetizar a rede bayesiana. Execute o notebook e, com base nos resultados obtidos (e em códigos adicionais, onde pertinente), realize as 5 atividades solicitadas.

Observação referente à Atividade (e): Se fosse "resolver à mão", aqui são necessários vários cálculos a partir das tabelas de probabilidades, enquanto as Atividades (b) e (c) saem direto de campos dessas tabelas. Apenas para exemplificar o que o programa vai fazer para obter a resposta, o resultado sai da seguinte operação, envolvendo campos das tabelas de probabilidades:

$$\Pr\left(C_3 = 2 \,|\, C_5 = 2\right) = \frac{\Pr\left(C_3 = 2 \wedge C_5 = 2\right)}{\Pr\left(C_5 = 2\right)} = \frac{\sum_{j=1,2} \Pr\left(C_3 = 2 \wedge C_5 = 2 \wedge C_4 = j\right)}{\sum_{i,j=1,2} \Pr\left(C_3 = i \wedge C_5 = 2 \wedge C_4 = j\right)}$$

Atividade prática #11: Árvore de decisão operando como classificador: análise de desempenho e interpretabilidade

Observação: A pontuação deste notebook é 50% daquela praticada em todos os notebooks de #01 a #10.

São considerados dois problemas de classificação nesta atividade, sendo que para o primeiro deles é necessário fazer o upload, na área de trabalho, do arquivo [**Social_Network_Ads.csv**]. Ambos os casos de estudo são bem simples, permitindo um acesso mais direto ao comportamento da árvore de decisão. Após executar todo o notebook [**EC11_DTclass.ipynb**], realize as duas atividades requeridas. Cabe destacar que o segundo conjunto de dados já havia sido considerado no EC#06.

Atividade prática #12: Sintonia de hiperparâmetros de um classificador Random Forest usando um algoritmo genético

Observação: A pontuação deste notebook é 50% daquela praticada em todos os notebooks de #01 a #10.

O ajuste de hiperparâmetros é uma parte essencial do pipeline de aprendizado de máquina de alto desempenho. As implementações mais comuns usam uma busca em grade (aleatória ou não) para escolher entre um conjunto de combinações. Nós fizemos uma busca em grade simplificada no EC#02. No entanto, a busca em grade claramente não escala bem e se torna inviável quando aumenta o número de hiperparâmetros, e também quando aumenta o número de valores diferentes para cada hiperparâmetro. Há motivação suficiente, portanto, para propor uma busca mais "esperta", capaz de conduzir a bons resultados usando um custo computacional menor.

Esta atividade usará um algoritmo evolutivo, disponível no pacote Python sklearn-genetic-opt, para encontrar os hiperparâmetros que otimizam uma métrica de validação cruzada. Vamos considerar uma Random Forest como classificador para dígitos manuscritos. A base MNIST não é, em geral, um caso de estudo indicado para árvores de decisão, em virtude de os atributos serem todos numéricos e existirem 784 atributos ao todo. No entanto, aqui vamos operar com uma versão de imagens 8 x 8 e um número reduzido de dados de treinamento, o que conduz a uma execução mais rápida da busca. O foco, portanto, deve estar na verificação do progresso da busca e no fluxo de informação, o qual pode ser adaptado a outros modelos de aprendizado de máquina. Execute o notebook [**EC12_RF_GA.ipynb**] e responda as 2 questões.

Atividade prática #13: Comparação entre modelos de predição de séries temporais

A predição de séries temporais pode ser implementada a partir de um grande número de técnicas, com tratamentos específicos para a série temporal e com objetivos variados. Nesta atividade, vamos empregar redes neurais recorrentes (SimpleRNN e LSTM), não-recorrentes (MLP) e um modelo de regressão linear na predição de duas séries temporais: Sunspot (média mensal de 1749 até 1983) e Unemployment rate (Fonte: U.S. Bureau of Labor Statistics) (de 01/1948 até 10/2024). As predições podem ser de um passo à frente ou de múltiplos passos à frente. Para efeito de comparação, entre as abordagens recorrentes e não-recorrentes, o número de desdobramentos no tempo e o número de atrasos (*tapped delay line*) serão os mesmos. As séries temporais sofrem apenas um escalamento, embora outros tratamentos, como eliminação de tendências, pudessem ser considerados. Não cabe estender os resultados das análises comparativas a outras séries temporais, pois cada série tem suas particularidades e vai desafiar os modelos de aprendizado de formas distintas. Por exemplo, a série Unemployment rate tem um evento anômalo nos dados de teste, pois esses incluem o efeito do lockdown da pandemia causada pelo coronavírus SARS-CoV-2. Já a série Sunspot é bem-comportada, permitindo um bom desempenho de modelos de predição lineares. Faça upload dos 2 datasets e execute o notebook [**EC13_RNNs_MLP_LP_TS.ipynb**] para os dois casos de estudo: Sunspot e Unemployment rate. Apresente, assim, dois notebooks em seu relatório. Responda as 3 questões das células finais em apenas um dos dois notebooks, mesmo que envolva resultados dos dois casos de estudo.