TP555 - AI/ML
Lista de Exercícios #10
Tensorflow

1. Após termos discutido sobre o TensorFlow, em sua opinião, quais são as principais vantagens da criação de um grafo de computação, em vez de executar diretamente os cálculos? Quais são as principais desvantagens?

2. O comando `a_val = a.eval(session=sess)` é equivalente a `a_val = sess.run(a)`? **Dica**: crie um grafo e execute as duas instruções em uma sessão e verifique o resultado.

3. O comando `a_val, b_val = a.eval(session=sess), b.eval(session=sess)` é equivalente a `a_val, b_val = sess.run([a, b])`?

4. É possível se executar dois grafos na mesma sessão?

5. Quando uma variável é inicializada? Quando ela é destruída?

6. Qual é a diferença entre um nó do tipo *placeholder* e um nó do tipo variável?

7. O que acontece quando você executa um grafo para avaliar uma operação que depende de um *placeholder*, mas você não fornece nenhum valor para o *placeholder*? O que acontece se a operação não depender do *placeholder*?

8. Como você pode definir o valor de uma variável com qualquer valor desejado durante a fase de execução? **Dica**: use um *placeholder* e verifique como o método `Assign` funciona no link abaixo: https://tensorflow.google.cn/api_docs/python/tf/Variable?hl=zh-cn#assign.

9. Implemente a regressão logística com o algoritmo do gradiente descendente em mini-lotes usando o TensorFlow. Use a seguinte função hipótese:

$$h_a(x) = f(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1^3 + a_6 x_2^3)$$

   onde $f(.)$ é a função de limiar sigmóide (ou logística). Treine e avalie o modelo no conjunto de dados da lua conforme mostrado abaixo. Divida o conjunto de dados em 80% para treinamento e 20% para validação.

```
N = 1000
X, y = make_moons(N, noise=0.1, random_state=42)
```

   Em seguida, faça o seguinte:

   a. Defina o grafo do regressor logístico.

   b. Salve os pontos de verificação usando um objeto da classe `Saver` em intervalos regulares durante o treinamento e salve o modelo final ao final do treinamento.

   c. Restaure o último ponto de verificação na inicialização, se o treinamento foi interrompido.

d. Adicione *summaries* para visualizar as curvas de aprendizado no *TensorBoard*.

e. Imprima o erro (*loss*) do modelo em intervalos regulares durante o treinamento.

f. Ajuste os hiperparâmetros: taxa de aprendizado e tamanho dos mini-lotes para encontrar um erro ótimo de treinamento.

g. Calcule a precisão do modelo com a função `precision_score` da biblioteca `SciKit-Learn`.

h. Plote os dados do conjunto de validação em relação às classes a que foram atribuídos. Ou seja, defina marcadores diferentes para identificar cada uma das classes na figura. Por exemplo, use círculos para denotar exemplos que pertencem à classe 0 e quadrados para denotar exemplos que pertencem à classe 1.

(**Dica:** A função `tf.sigmoid(x, name=None)` calcula a função sigmóide de x elemento-a-elemento.)

(**Dica:** A função `tf.compat.v1.losses.log_loss(labels, predictions, scope=None)` adiciona um termo de perda de log loss ao procedimento de treinamento.)

(**Dica:** Você pode inicializar a variável `theta` utilizando a função `tf.random_uniform`, o link para a documentação desta função é https://www.tensorflow.org/api_docs/python/tf/random/uniform).

10. Modifique o código anterior para implementar a regressão logística utilizando o gradiente descendente estocástico (*SGD*) com TensorFlow.