TP555 - AI/ML

Lista de Exercícios #5

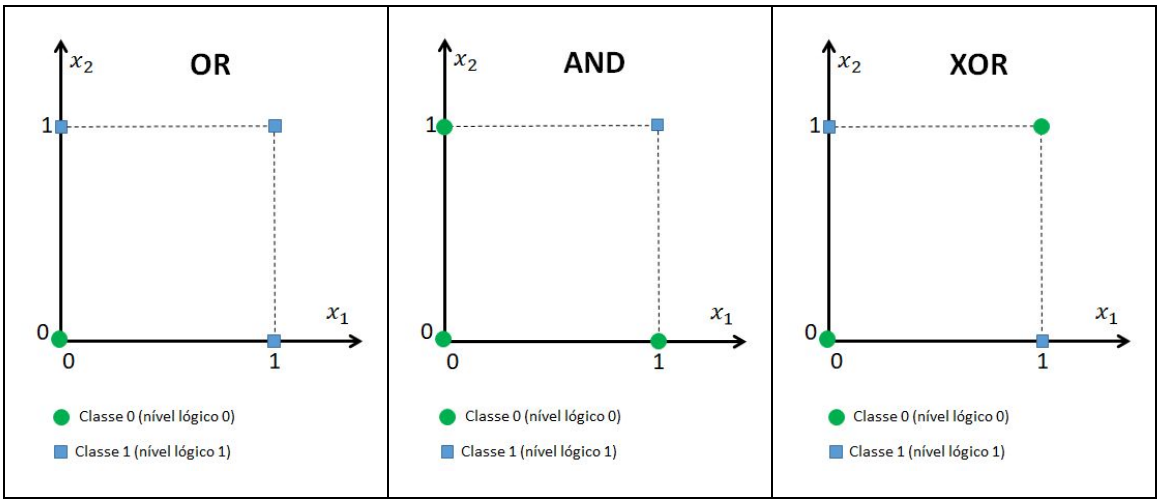
Classificação Linear: Parte 2

1. Dado as funções lógicas: OR, AND e XOR, representadas pelas tabelas e gráficos abaixo. Responda quais podem ser classificados com um classificador linear, ou seja, uma linha reta, que separa as duas classes. Caso algum deles não possa ser separado linearmente, que tipo de classificador seria necessário?

OR		
x1	**x2**	**y**
0	0	0
0	1	1
1	0	1
1	1	1

AND		
x1	**x2**	**y**
0	0	0
0	1	0
1	0	0
1	1	1

XOR		
x1	**x2**	**y**
0	0	0
0	1	1
1	0	1
1	1	0



2. Suponha que você tenha um problema de classificação com múltiplas classes, ou seja, Q > 2, (então y ∈ {1,2,…, Q}). Usando o método ***um-contra-todos***, quantos classificadores de regressão logística diferentes você precisaria treinar para realizar a classificação destas classes?
3. Suponha que você deseje prever, a partir dos atributos x de um tumor, se ele é maligno (y = 1) ou benigno (y = 0). Um classificador de regressão logística gera, para um tumor específico, $h_a(x) = P(y = 1 \mid x;\ a) = 0.7$, portanto estima-se que haja 70% de chance de esse tumor ser maligno. Qual seria a estimativa para $P(y = 0 \mid x;\ a)$, ou seja, a probabilidade de o tumor ser benigno?

4. Considere a regressão logística com 2 atributos, x_1 e x_2. Suponha que $a_0 = 5$, $a_1 = -1$ e $a_2 = 0$, de tal forma que $h_a(x) = f(5 - x_1)$. Encontre e desenhe a fronteira de decisão. Mostre as regiões em que o classificador classifica y=1 (classe positiva) e y=0 (classe negativa).
5. Suponha que você quisesse classificar fotos como externas/internas e diurnas/noturnas. Nesse caso, você deve implementar dois classificadores de regressão logística ou um classificador de regressão Softmax?
6. Utilizando a base de dados ***20 Newsgroups*** da biblioteca scikit-learn, classifique os textos do conjunto de validação (ou testes) em uma das 4 categorias: 'talk.religion.misc', 'soc.religion.christian', 'sci.space' ou 'comp.graphics'. Apresente a ***matriz de confusão*** e as principais ***métricas de classificação*** utilizando a função ***classification_report*** (veja a dica abaixo).
   (**Dica**: Veja como baixar e usar a base de dados no exemplo: ClassifyingTextMultinomialNB.ipynb).
   (**Dica**: Você só precisa baixar as 4 categorias de texto e dividi-las em 2 conjuntos um de treinamento e outro de validação, conforme feito no exemplo acima).
   (**Dica**: documentação da função ***classification_report***: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html).
7. Utilizando a seguinte função hipótese $h_a(x) = f(a_0 + a_1x_1 + a_2x_2)$, onde $f(.)$ é a função de limiar sigmóide (ou logística), e o algoritmo do ***gradiente descendente em batelada com early-stopping***, treine um classificador linear para classificar os seguintes dados. Faça o seguinte
   a. Plote um gráfico mostrando as diferentes classes.
   b. Quantas classes distintas você observa no gráfico acima?
   c. Analisando o gráfico do item (a), que tipo de fronteira de decisão seria necessária para separar essas classes (linear ou não-linear)?
   d. Plote um gráfico com número de épocas versus os erros de treinamento e validação.
   e. Plote a matriz de confusão.
   f. Plote uma figura com as fronteiras de decisão.
   g. Plote o gráfico com a curva característica de operação do receptor (ROC).
   h. Imprima as ***métricas de classificação*** utilizando a função ***classification_report***.
   i. Repita os itens (d) até (h), agora com a seguinte função hipótese:
   $$h_a(x) = f(a_0 + a_1x_1 + a_2x_2 + a_3x_1^2 + a_4x_2^2).$$
   j. Qual a diferença na performance do classificador entre as duas funções hipóteses? (**Dica**: qual das 2 hipóteses confere ao classificador maior precisão de classificação?)

   (**Dica**: utilize como base o notebook ClassificationOfTwoLinearlySeparableClasses.ipynb, nele você vai encontrar a

implementação do algoritmo do ***gradiente descendente em batelada com early-stopping***).
(**Dica**: não se esqueça de encontrar o melhor valor para o ***passo de aprendizagem***).

8. Neste exercício, você irá implementar o ***gradiente descendente em batelada com early-stopping*** para a regressão ***Softmax*** (ou seja, para classificação quando Q > 2) sem usar a biblioteca Scikit-Learn para classificar os dados pertencentes à 3 classes diferentes. Utilize o trecho de código abaixo para gerar os dados das 3 classes. Esse trecho também plota uma figura mostrando os dados pertencentes às 3 classes.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.datasets import make_blobs
import seaborn as sns

# make 3-class dataset for classification
centers = [[-5, 0], [0, 1.5], [5, -1]]
x, y = make_blobs(n_samples=1000, centers=centers, random_state=42)

idx0 = np.argwhere(y == 0)
idx1 = np.argwhere(y == 1)
idx2 = np.argwhere(y == 2)

fig = plt.figure(figsize=(5,5))
plt.plot(x[idx0,0], x[idx0,1], '.', label='Class 0')
plt.plot(x[idx1,0], x[idx1,1], 'rx', label='Class 1')
plt.plot(x[idx2,0], x[idx2,1], 'ko', label='Class 2')
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.legend()
plt.show()
```

Você pode utilizar, se desejar, as seguintes funções auxiliares.

```
def to_one_hot(y):
    n_classes = y.max() + 1
    m = len(y)
    Y_one_hot = np.zeros((m, n_classes))
    Y_one_hot[np.arange(m), y] = 1
    return Y_one_hot

def softmax(logits):
    exps = np.exp(logits)
    exp_sums = np.sum(exps, axis=1, keepdims=True)
    return exps / exp_sums

def error_function(x, a, y, epsilon):
```

```
    logits = x.dot(a)
    y_prob = softmax(logits)
    error = -np.mean(np.sum(y * np.log(y_prob + epsilon), axis=1))
    return error

def classifier(x, a):
    logits = x.dot(a)
    y_prob = softmax(logits)
    c = np.zeros((len(y_prob), 1))
    for i in range(0, len(y_prob)):
        c[i, 0] = np.argwhere(y_prob[i,:] == y_prob[i,:].max())[0]
    return c

def predict_prob(x, a):
    logits = x.dot(a)
    y_prob = softmax(logits)
    h1 = y_prob
    h0 = 1 - h1
    h = np.c_[h0, h1]
    return h
```

Utilizando-se a seguinte função hipótese $h_a(x) = f(a_0 + a_1 x_1 + a_2 x_2)$, pede-se

A. Divida o conjunto de dados em 75% para treinamento e 25% para validação.
B. Plote um gráfico com número de épocas versus os erros de treinamento e validação.
C. Plote a matriz de confusão.
D. Plote uma figura mostrando as fronteiras de decisão.
E. Imprima as ***métricas de classificação*** utilizando a função ***classification_report***.

(**Dica**: utilize como base o notebook ClassificationOfTwoLinearlySeparableClasses.ipynb, nele você vai encontrar a implementação do algoritmo do ***gradiente descendente em batelada com early-stopping***).

(**Dica**: não se esqueça de encontrar o melhor valor para o ***passo de aprendizagem***).

(**Dica**: lembre-se que o vetor de rótulos, y, deve ser convertido para a representação ***one-hot encoding***, veja a função ***to_one_hot*** definida acima).