TP555 - AI/ML

Lista de Exercícios #5

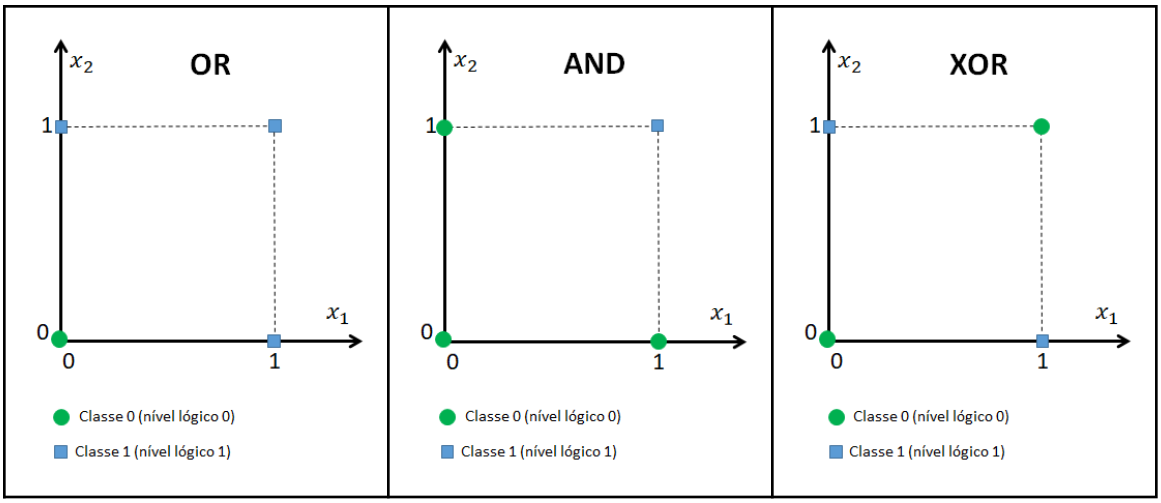
Classificação Linear: Parte 2

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. Implemente a **regra de atualização do perceptron com early-stop** e faça o seguinte:
    A. Classifique o conjunto de dados gerado pelo código abaixo.

```
from sklearn.datasets import make_blobs

# Define a seed.
seed = 42

# Number of examples.
N = 1000

# Create a 2-class dataset for classification.
centers = [[-3.0, 0], [0, 1.0]]
X, y = make_blobs(n_samples=N, centers=centers, random_state=seed)
```

    B. Plote a curva do erro quadrático médio versus as iterações de treinamento.
    C. O erro do modelo converge para algum valor?
    D. Qual a acurácia do do classificador no conjunto total de dados?
    E. Use o código abaixo para gerar um novo conjunto de dados e o classifique.

```
from sklearn.datasets import make_blobs

# Define a seed.
seed = 42

# Number of examples.
N = 1000

# Create a 2-class dataset for classification.
centers = [[-3.0, -1.5], [3, 1.5]]
X, y = make_blobs(n_samples=N, centers=centers, random_state=seed)
```

    F. Plote a curva do erro quadrático médio versus as iterações de treinamento.
    G. O erro do modelo converge para algum valor?
    H. Qual a acurácia do do classificador no conjunto total de dados?
    I. Comparando os dois resultados, o que você pode concluir a respeito do comportamento da **regra de atualização do perceptron** e os dois conjuntos de dados utilizados?

7. 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). Use os seguintes classificadores e compare suas performances: MultinomialNB, BernoulliNB, Softmax e regressão logística com abordagem um contra o resto. Qual deles apresenta a melhor performance.
   (**Dica**: Use apenas as classes disponibilizadas pela biblioteca SciKit-Learn.)
   (**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).
8. Implemente um **regressor logístico** utilizando o algoritmo do ***gradiente descendente em batelada com early-stop***. Em seguida, o teste com a base de dados de mensagens de SPAM usados nos exemplos mostrados em sala de aula.
9. Utilizando a seguinte função hipótese $h_a(x) = f(a_0 + a_1 x_1 + a_2 x_2)$, onde $f(.)$é a função de limiar sigmóide (ou logística), e o algoritmo do ***gradiente descendente em batelada com early-stop***, treine um classificador linear para classificar os dados gerados através da execução do código abaixo.

```
# Import all necessary libraries.
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_circles

# Create concentric circles.
x, y = make_circles(n_samples=1000, random_state=42, noise=0.1, factor=0.2)

# Split array into random train and test subsets.
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=23)
```

   Em seguida, faça o seguinte
   a. Plote um gráfico mostrando as diferentes classes.
   b. Analisando o gráfico do item (a), que tipo de fronteira de decisão seria necessária para separar essas classes (linear ou não-linear)?
   c. Plote um gráfico com número de épocas versus os erros de treinamento e validação.
   d. Plote a matriz de confusão.
   e. Plote uma figura com as fronteiras de decisão.
   f. Plote o gráfico com a curva característica de operação do receptor (ROC).
   g. Imprima as ***métricas de classificação*** utilizando a função ***classification_report***.

h. Repita os itens (c) até (g), agora com a seguinte função hipótese:

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

i. 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***).

10. Neste exercício, você irá implementar o ***gradiente descendente em batelada com early-stop*** para a regressão ***Softmax*** (ou seja, para classificação quando Q > 2) sem usar a biblioteca Scikit-Learn para classificar os dados pertencentes a 3 classes diferentes. Utilize um número máximo de iterações igual a 2000, por exemplo. 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

# epsilon é uma valor muito pequeno, e.g., 1e-7, usado para se evitar que quando y_prob for
# zero o resultado não seja infinito.
# epsilon igual a um valor pequeno evita esse tipo de Warning: "RuntimeWarning: divide by
zero encountered in log"
def error_function(x, a, y, epsilon=1e-7):
    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 os vetores de rótulos dos conjunto de treinamento e validação, y_train e y_test, devem ser convertidos para a representação ***one-hot encoding***, veja a função ***to_one_hot*** definida acima).

11. Neste exercício você irá utilizar uma implementação do regressor logístico e as abordagens Um-Contra-o-Resto e Um-Contra-Um para realizar classificação multiclasses. Inicialmente, implemente uma função que realize o treinamento de um regressor logístico com early-stop. Utilize no mínimo 40000 épocas para o treinamento de cada um dos classificadores. Encontre um passo de aprendizagem que seja ótimo para todos os classificadores treinados. Utilize o trecho de código abaixo para gerar exemplos pertencentes a 5 classes.

```
# Number of examples.
M = 1000

# Create 3-class dataset for classification.
centers = [[-5, 0], [0, 1.5], [5, -1], [10, 1.5], [15, 0]]
x, y = make_blobs(n_samples=M, centers=centers, random_state=seed)
```

Em seguida, faça o seguinte:

A. Implemente a abordagem para a classificação multiclasse chamada Um-Contra-o-Resto.
B. Utilizando a abordagem Um-Contra-o-Resto faça a predição utilizando toda a base de dados.
C. Gere a matriz de confusão e calcule o accuracy_score dessa abordagem.
D. Implemente a abordagem para a classificação multiclasse chamada Um-Contra-Um.
E. Utilizando a abordagem Um-Contra-Um faça a predição utilizando toda a base de dados.
F. Gere a matriz de confusão e calcule o accuracy_score dessa abordagem.
G. Qual das duas abordagens apresenta melhor resultado?

12. Neste exercício você usará sua implementação do regressor logístico utilizando o algoritmo do ***gradiente descendente em batelada com early-stop*** para realizar a classificação entre 2 modulações digitais. A base de dados com os sinais InPhase e Quadrature (i.e., I e Q), além do label indicando a modulação estão no arquivo modulations.csv. Baixe o arquivo e o utilize para treinamento e validação do seu classificador. Utilize o trecho de código abaixo para ler as informações do arquivo, elas estão da seguinte forma: primeira coluna amostras InPhase (I), segunda coluna amostras em Quadrature (Q) e a terceira coluna label a modulação, onde 0 indica BPSK e 1 QPSK.

```
def readCSVFile(filename):
  df = pd.read_csv(filename, header=None)
```

```
    i = df[0].to_numpy()
    q = df[1].to_numpy()
    mod = df[2].to_numpy()
    i = i.reshape(len(i),1)
    q = q.reshape(len(q),1)
    mod = mod.reshape(len(mod),)
    return i, q, mod

filename = './modulations.csv'
I, Q, mod = readCSVFile(filename)
y = I + 1j*Q
```

De posse das informações da base de dados, faça o seguinte:

A. Plote um gráfico com as amostras I e Q indicando a qual modulação cada par de amostras I e Q pertence.
B. Analisando o gráfico do item (A), que tipo de fronteira de decisão seria necessária para separar essas classes (linear ou não-linear)?
C. Treine seu regressor logístico com early stopping com pelo menos 30000 épocas. Não se esqueça de encontrar o melhor passo de aprendizagem. **OBS**.: Você irá precisar encontrar uma função hipótese que consiga separar as duas classes. Analise o gráfico da letra (A) para identificar que tipo de função hipótese é necessária.
D. Plote o gráfico mostrando os erros de treinamento e validação versus o número de épocas.
E. Imprima o valor da acurácia do seu modelo, use a função: *accuracy_score*.
F. Plote a matriz de confusão.
G. Plote uma figura mostrando as fronteiras de decisão.
H. Plote o gráfico com a curva característica de operação do receptor (ROC).
I. Imprima as ***métricas de classificação*** utilizando a função ***classification_report***.

13. Dispositivos bluetooth transmitem na faixa de 2.4 GHz e utilizam *frequency hopping*, ou seja, mudam a frequência em que transmitem seus sinais, normalmente, a cada nova transmissão. Isso causa interferência em outras tecnologias, como por exemplo Wi-Fi. Vamos supor que uma determinada faixa de frequência seja dividida em 4 canais e que dispositivos bluetooth sempre tenham o seguinte padrão de transmissão: 0, 1, 2, e 3. Ao final do último canal, i.e., 3, o padrão se repete. Seria interessante se conseguíssemos, a partir da observação do canal atual, saber quais canais podem ser utilizados durante o próximo intervalo de transmissão para que não houvesse interferência.
    a. Você acha ser possível criar um modelo de classificação que dado o canal atual produza em sua saída a **predição do próximo canal que será utilizado**? Se sim, prove isto através de um experimento (crie um notebook Jupyter).
    b. O que aconteceria com seu modelo se agora o padrão de transmissão fosse 0, 3, 1, 0, 2, e 3. Após o último canal, a sequência se repete. Com a nova sequência de canais, você acha ser possível criar um modelo de classificação que ***dado o***

canal atual produza em sua saída a **predição do próximo canal que será utilizado**? Crie um notebook para este experimento.

c. Foi possível se predizer o próximo canal que será utilizado usando a sequência dada no item anterior? Caso não, explique o motivo do modelo não ser capaz de predizer o próximo canal e proponha (ou seja, implemente no notebook) uma possível solução para se resolver este problema.