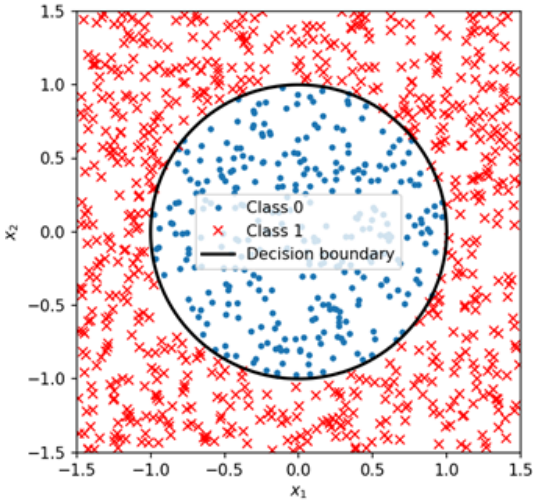
TP555 - AI/ML

Lista de Exercícios #13

Máquina de Vetores de Suporte (SVM)

1. A figura abaixo mostra como dados circulares centrados na origem podem ser linearmente separados mapeando os atributos (x_1, x_2) para as duas dimensões (x_1^2, x_2^2). Mas e se os dados não estiverem localizados na origem e não tiverem uma distribuição circular? E se os dados tiverem uma distribuição em forma de uma elipse, e não um círculo? A equação geral para um círculo (e, portanto, o limiar de decisão) é $(x_1 - a)^2 + (x_2 - b)^2 - r^2 = 0$, e a equação geral para uma elipse é $c(x_1 - a)^2 + d(x_2 - b)^2 - 1 = 0$.
    a. Expanda a equação do círculo e mostre quais seriam os pesos w_i para o limiar de decisão no espaço de atributos quadridimensional (x_1, x_2, x_1^2, x_2^2). Explique por que isso significa que qualquer círculo é linearmente separável neste espaço.
    b. Faça o mesmo para elipses no espaço de atributos de cinco dimensões $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.



2. Construa uma máquina de vetores de suporte que calcule a função XOR. Use valores de +1 e -1 (em vez de 1 e 0) para entradas e saídas, de modo que um exemplo se pareça com ([−1, 1], 1) ou ([−1, −1], −1) . Mapeie a entrada $[x_1, x_2]$ em um espaço que consiste em x_1 e $x_1 x_2$. Desenhe os quatro exemplos de entrada neste espaço e o separador de margem máxima. Qual é a margem? Agora desenhe a linha de separação de volta no espaço de entrada euclidiano original.

3. Considere o problema de separar N exemplos de dados em exemplos positivos e negativos usando um separador linear. Obviamente, isso sempre pode ser feito para N = 2 exemplos em uma linha de dimensão d = 1, independentemente de como os exemplos são rotulados ou de onde estão localizados (a menos que os exemplos estejam no mesmo lugar).
    a. Mostre que isso sempre pode ser feito para N = 3 exemplos em um plano de dimensão d = 2, a menos que sejam colineares.
    b. Mostre que isso nem sempre pode ser feito para N = 4 exemplos em um plano de dimensão d = 2.
    c. Mostre que isso sempre pode ser feito para N = 4 exemplos em um espaço de dimensão d = 3, a menos que sejam coplanares.
    d. Mostre que isso nem sempre pode ser feito para N = 5 exemplos em um espaço de dimensão d = 3.
    e. Tente provar que N exemplos na posição geral (mas não N + 1) são linearmente separáveis em um espaço de dimensão N - 1.
4. Treine um classificador SVM no conjunto de dados MNIST. Visto que os classificadores SVM são classificadores binários, você precisará usar a abordagem **um-contra-todos** para classificar todos os 10 dígitos. Você pode querer ajustar os hiperparâmetros usando pequenos conjuntos de validação para acelerar o processo. Seu modelo deve atingir uma precisão de no mínimo 97% no conjunto de testes. Que precisão você conseguiu alcançar? (**Dica**: escalone os atributos, experimente diferentes tipos de kernels (e.g., linear e RBF), use ***RandomSearch*** para encontrar os melhores valores para os hiperparâmetros ***C*** com uma distribuição recíproca no intervalo de 0.001 a 0.1 e ***gamma*** com uma distribuição uniforme no intervalo de 1 a 10). O módulo ***stats*** da biblioteca ***scipy*** oferece as funções ***reciprocal*** e ***uniform*** que podem ser utilizadas com o ***RandomSearch***).
5. Treine um regressor SVM no conjunto de dados de moradia da Califórnia. (**Dica**: escalone os atributos, experimente diferentes tipos de kernels (e.g., linear e RBF), use ***RandomSearch*** para encontrar os melhores valores para os hiperparâmetros ***C*** com uma distribuição recíproca no intervalo de 0.001 a 0.1 e ***gamma*** com uma distribuição uniforme no intervalo de 1 a 10). O módulo ***stats*** da biblioteca ***scipy*** oferece as funções ***reciprocal*** e ***uniform*** que podem ser utilizadas com o ***RandomSearch***.
6. Use uma SVM para classificar os símbolos das modulações BPSK e QPSK. Use o trecho de código dado abaixo para gerar o conjunto de treinamento. De posse do conjunto de treinamento, faça o seguinte:
    a. Plote os símbolos e suas respectivas classes.
    b. Treine uma SVM no conjunto de treinamento que atinja no mínimo 99% de acurácia neste mesmo conjunto.
    c. Plote a fronteira de decisão encontrada pela SVM e os símbolos.
    d. Use uma SVM com ***kernel linear*** e a seguinte transformação dimensional: $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.
    e. Plote a fronteira de decisão encontrada pela SVM e os símbolos.

f. Qual a acurácia obtida com esta transformação?
g. Veja que a transformação dimensional acima resulta num espaço de atributos com 5 dimensões. Você acha que os dados poderiam ser separados com uma transformação que resultasse num espaço de atributos de menor dimensão? Se sim, use esta nova transformação para separar os dados. (**Dica**: não é necessário se obter uma alta acurácia.).

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import erfc
from sklearn.svm import SVC

np.random.seed(42)

# Definition of several utility functions.
def mod(bits, modtype):
    symbols = []
    ip = np.zeros((len(bits),1), dtype=complex)
    inc = 0
    if(modtype==2):
        symbols_bpsk = [-1.0 + 1j*0.0, 1.0 + 1j*0.0]
        for b in bits:
            ip[inc] = symbols_bpsk[b[0]]
            inc += 1
        # Normalization of energy to 1.
        symbols = (1.0/np.sqrt(1.0))*ip
    elif(modtype==4):
        symbols_qpsk = [-1.0 - 1j*1.0, -1.0 + 1j*1.0, 1.0 - 1j*1.0, 1.0 + 1j*1.0]
        for b in bits:
            ip[inc] = symbols_qpsk[b[0]]
            inc += 1
        # Normalization of energy to 1.
        symbols = (1.0/np.sqrt(2.0))*ip
    elif(modtype==16):
        symbols_16qam = [-3.0 - 1j*3.0, -3.0 - 1j*1.0, -3.0 + 1j*3.0, -3.0 + 1j*1.0,
                         -1.0 - 1j*3.0, -1.0 - 1j*1.0, -1.0 + 1j*3.0, -1.0 + 1j*1.0,
                         +3.0 - 1j*3.0, +3.0 - 1j*1.0, +3.0 + 1j*3.0, +3.0 + 1j*1.0,
                         +1.0 - 1j*3.0, +1.0 - 1j*1.0, +1.0 + 1j*3.0, +1.0 + 1j*1.0
                        ]
        for b in bits:
            ip[inc] = symbols_16qam[b[0]]
            inc += 1
        # Normalization of energy to 1.
        symbols = (1.0/np.sqrt(10.0))*ip
    else:
        print('Error: Modulation not implemented.')
```

```
    return symbols

# Number of symbols to be transmitted.
N = 1000

# Generate BPSK symbols.
bits_bpsk = np.random.randint(0, 2, (N, 1))
# Modulate the binary stream into BPSK symbols.
bpsk = mod(bits_bpsk, 2)

# Generate QPSK symbols.
bits_qpsk = np.random.randint(0, 4, (N, 1))
# Modulate the binary stream into QPSK symbols.
qpsk = mod(bits_qpsk, 4)

X = np.zeros((N,1), dtype=complex)
y = np.random.randint(0, 2, (N,))
for i in range(N):
    if y[i] == 0:
        X[i] = bpsk[i]
    else:
        X[i] = qpsk[i]

# Create Es/N0 vector.
EsN0dB = 15
EsN0Lin = 10.0**(-(EsN0dB/10.0))

# Pass symbols through AWGN channel.
noise = np.sqrt(EsN0Lin/2.0)*(np.random.randn(N, 1) + 1j*np.random.randn(N, 1))
X = X + noise

X = np.c_[np.real(X),np.imag(X)]
```