**Gustavo Kreuzer Marengo - 835**

**TP555 - AI/ML**

**Lista de Exercícios #3**

**Regressão Polinomial**

1. Suponha que você esteja usando regressão polinomial. Você plota as **curvas de**

**aprendizado** e percebe que há uma grande diferença entre o erro de treinamento e o

erro de validação. O que está acontecendo? Quais são as três maneiras de resolver

isso?

A curva de aprendizado está plotando um algoritmo cujo a função hipótese é um polinomio de grau elevado, havendo uma grande diferença entre as curvas de treinamento e validação. Para resolver esse problema tem-se as seguintes soluções: aumentar o conjunto do treinamento, diminuir a ordem do modelo ou utilizar um modelo com um número menor de exemplos.

**OBS** .: **Curvas de aprendizado** : são gráficos mostrando o desempenho do modelo no

conjunto de treinamento e no conjunto de validação em função do tamanho do conjunto

de treinamento (ou da iteração do treinamento).

2. Exercício de comparação entre as regressões Ridge e LASSO. Dada a seguinte versão

ruidosa da função objetivo y\_noisy = 2 + x + 0.5\*x^2 + n, onde x é um vetor coluna com

M = 100 elementos retirados de uma distribuição aleatória uniformemente distribuída

variando entre -3 e 3 e n é o vetor ruído com M elementos retirados de uma distribuição

aleatória Gaussiana com média 0 e variância unitária. Utilize um polinômio de ordem 90,

padronização de atributos (ou seja, remoção da média e divisão pelo desvio padrão) e

regressão LASSO (utilize a biblioteca SciKit-Learn) com λ variando entre 1e-10 e 1

(utilize *np.linspace* (10\*\*-10, 1, 1000)). Utilizando a função “ *train\_test\_split* ”, divida os

exemplos em um conjunto de treinamento e outro de validação com proporção 70% e

30%, respectivamente. Faça o seguinte

a. Plote um gráfico mostrando a função objetivo e sua versão ruidosa.

b. Crie um loop para testar cada um dos 1000 valores de λ. Para cada novo valor

de λ, treine o modelo, execute a predição e calcule os erros de treinamento e

validação.

c. Para cada iteração do loop, armazene os valores do erro de treinamento e

validação em um vetor.

d. Para cada iteração do loop, verifique se o valor do erro de validação atual é

menor do que o erro de validação mínimo. Se sim, armazene o valor de λ e o

modelo utilizado para aquela iteração. (**D ica** : inicialize a variável contendo o erro

de validação mínimo como: *minimum\_val\_error = float("inf")* ).

e. Plote um gráfico mostrando os erros de treinamento e validação versus os

valores de λ.

f. Baseado no menor valor do erro de validação, qual é o valor ótimo para λ?

g. Dado que você armazenou o modelo que obteve o menor erro de validação,

utilize-o para criar um gráfico que mostre a função hipótese (ou seja, o

mapeamento do atributos de entrada, x, nos valores de saída, y, através do

modelo treinado) e a função objetivo e sua versão ruidosa.

h. Imprima os valores do pesos obtidos durante o treinamento do modelo que

obteve o menor erro de validação ( **Dica** : use o atributo ***named\_steps*** da classe

Pipeline para acessar os objetos que compõem o pipeline:

https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html .

i. Repita os passos anteriores para a regressão de Ridge.

j. O que você percebe com relação aos pesos obtidos com as regressões Ridge e

LASSO?

( **Dica** : Não se esqueça que os parâmetros do escalonamento de atributos, ou seja,

média e devio padrão, são encontrados utilizando-se o conjunto de treinamento. Os

parâmetros encontrados são utilizados para escalonar o conjunto de validação).

( **Dica** : Na instanciação da classe Lasso, configure a tolerância para 1, i.e., tol=1.)

( **Dica** : A documentação do regressor LASSO pode ser acessada através deste link:

https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Lasso.html?highli

ght=lasso#sklearn.linear\_model.Lasso )

( **Dica** : utilize a classe clone para criar uma cópia do modelo que atingiu erro de

validação menor do que o valor mínimo atual.

https://scikit-learn.org/stable/modules/generated/sklearn.base.clone.html )

3. Exercício sobre Early stopping. Dada a seguinte versão ruidosa da função objetivo

y\_noisy = 2 + x + 0.5\*x^2 + x^3 + n, onde x é um vetor coluna com M = 100 elementos

retirados de uma distribuição aleatória uniformemente distribuída variando entre -3 e 3 e

n é o vetor ruído com M elementos retirados de uma distribuição aleatória Gaussiana

com média 0 e variância unitária. Utilize um polinômio de ordem 30 como função

hipótese, padronização de atributos (ou seja, remoção da média e divisão pelo desvio

padrão) e o algoritmo do gradiente descendente em batelada. Utilizando a função

“ ***train\_test\_split*** ”, divida os exemplos em um conjunto de treinamento e outro de

validação com proporção 70% e 30%, respectivamente. Faça o seguinte

a. Plote um gráfico mostrando a função objetivo e sua versão ruidosa.

b. Encontre, manualmente, o melhor valor para o passo de aprendizagem.

c. Execute o treinamento por 1000 épocas.

d. Para cada época, armazene em um vetor os valores do erro de treinamento e

validação.

e. Para cada época, verifique se o valor do erro de validação atual é menor do que

o erro de validação mínimo. Se sim, armazene o modelo utilizado para aquela

época, ou seja, os valores dos pesos, e o valor do erro de validação para aquela

época. ( **Dica** : inicialize a variável contendo o erro de validação mínimo como:

*minimum\_val\_error = float("inf")* ).

f. Plote um gráfico mostrando os erros de treinamento e validação versus o

número de épocas.

g. Dado que você armazenou o modelo que obteve o menor erro de validação,

utilize-o para criar um gráfico que mostre a função hipótese (ou seja, o

mapeamento do atributos de entrada, x, nos valores de saída, y, através do

modelo treinado) e a função objetivo e sua versão ruidosa.

4. Exercício que utiliza validação cruzada. Usando o arquivo ***covid19.csv*** , onde onde a

primeira coluna são os valores de x (i.e., atributo) representando o número de dias

desde o primeiro caso confirmado de COVID-19 e a segunda coluna são os valores de y

(i.e., objetivo ou rótulo), representando o número de casos de COVID-19 ativos. Leia o

conteúdo do arquivo, ou seja, os vetores x e y, com os seguintes comandos:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(‘./covid19.csv', header=None)

x = df[0].to\_numpy()

y = df[1].to\_numpy()

x = x.reshape(len(x),1)

y = y.reshape(len(y),1)

fig = plt.figure(figsize=(10,10))

plt.plot(x, y, 'b.')

Em seguida, faça o seguinte

a. Plote os valores do arquivo, i.e., um gráfico mostrando o dias desde o primeiro

caso versus o número de casos ativos.

b. Encontre uma aproximação polinomial que represente bem os dados do arquivo.

Para encontrar a melhor aproximação, utilize os seguintes métodos: validação

cruzada holdout (com 80% do conjunto original para treinamento e 20% para

validação), validação cruzada k-fold (com k=10 folds), validação cruzada

leave-p-out (com p=1) e curvas de aprendizado. Analise polinômios com ordem

variando de 1 até 12.

c. Em seguida, de posse da melhor ordem de polinômio que aproxima os dados do

arquivo csv, treine o modelo com todos os dados do arquivo csv. Utilize

padronização de atributos com a classe StandardScaler da biblioteca

SciKit-Learn.

d. De posse do modelo treinado, crie um vetor x variando de 1 a 70 com

incrementos de 1 em 1, i.e., número de dias desde o primeiro caso registrado até

70 dias depois, e faça a predição do número de casos ativos até 70 dias após o

primeiro caso registrado.

e. Sabendo que o número total de leitos de UTI no Brasil é de 40600 (aqui vamos

supor que nenhum leito está ocupado no momento), preveja em quantos dias

desde o início do primeiro caso registrado no Brasil (26-02-2020) o número de

leitos total seria atingido.

5. Neste exercício você vai utilizar o arquivo ***reg\_poli.csv*** onde a primeira coluna são os

valores de x (atributo) e a segunda de y (objetivo ou rótulo). Após, leia o conteúdo do

arquivo, ou seja, os vetores x e y, com os seguintes comandos:

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(‘./reg\_poli.csv', header=None)

x = df[0].to\_numpy()

y = df[1].to\_numpy()

x = x.reshape(len(x), 1)

y = y.reshape(len(y), 1)

fig = plt.figure(figsize=(10,10))

plt.plot(x, y, 'b.')

Em seguida

a. Apresente o gráfico de x versus y, mostrando os **pontos** amostrados do modelo

gerador.

b. Encontre uma aproximação polinomial que represente bem os dados do arquivo.

Para encontrar a melhor aproximação, utilize os seguintes métodos: validação

cruzada holdout (com 70% do conjunto original para treinamento e 30% para

validação), validação cruzada k-fold (com k=10 folds), validação cruzada

leave-p-out (com p=1) e curvas de aprendizado. Analise polinômios com ordem

variando de 1 até 12.

c. Em seguida, de posse da melhor ordem de polinômio que aproxima os dados do

arquivo csv, treine o modelo com todos os dados do arquivo csv. Utilize

padronização de atributos com a classe StandardScaler da biblioteca

SciKit-Learn.

d. Plote um gráfico que mostre os pontos ruidosos do arquivo csv e os valores

encontrados com o modelo para dos valores de x vindos do arquivo csv