TP555 - AI/ML

Lista de Exercícios #3

Regressão para Modelos Não-Lineares

1. Exercício sobre regressão não-linear. Dada a seguinte função não-linear

   y = a0*x^(a1)

   Como você poderia linearizar essa função (lembre-se que a nova função é linear em relação aos pesos e não em relação ao atributo)? Qual é a função hipótese que você poderia usar para encontrar os pesos da função geradora? Imagine que você só tem acesso aos vetores x e y. Supondo que a0 = 1.4 e a1 = 3.0 e x é igual a um vetor com M = 1000 amostras retiradas de uma distribuição Uniforme no intervalo [0.0, 10.0), faça o seguinte:
   a. Plote o gráfico da função não-linear (i.e., x versus y).
   b. Linearize a função não-linear.
   c. Plote o gráfico da função linearizada, i.e., x versus y.
   d. Utilizando a função hipótese que lineariza a função-geradora,
      - Plote a superfície de erro.
      - Encontre os valores ótimos dos pesos da função hipótese utilizando a equação normal.
      - Encontre os pesos da função hipótese utilizando o algoritmo do gradiente descendente em batelada com ***critério de parada*** definido como sendo quando a ***diferença absoluta*** entre o erro da iteração atual e a anterior caia abaixo de 0.00001 ou que o número máximo de épocas tenha sido atingido. Faça o número máximo de épocas igual a 1000. (**OBS**.: não se esqueça de encontrar o melhor valor para o passo de aprendizagem).
      - Plote a superfície de contorno mostrando o histórico de atualização dos pesos e seus valores ótimos.
      - Plote o gráfico de erro versus número de épocas.
   e. Quais os valores dos pesos encontrados pelo o algoritmo do gradiente descendente em batelada?
2. 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?

   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).
3. 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. (**Dica**: 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 todos os valores de λ, ou seja, os 1000 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?highlight=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)

4. 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.

5. 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 o modelo gerador, 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.

6. 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). O arquivo contém a versão ruidosa da função original, ou seja o modelo gerador ao qual ruído é adicionado. 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 ***reg_poli.csv*** e os valores encontrados com o modelo para os valores de x vindos do arquivo csv, ou seja, use o modelo para "prever" os valores de y com os valores de x vindos do arquivo.