TP555 - AI/ML

Lista de Exercícios #2

Regressão Linear

1. Qual técnica de regressão linear você usaria se tivesse um conjunto de treinamento com milhares de features (i.e., atributos)? Explique por quais razões você utilizaria esta técnica.
2. Suponha que as features (i.e., atributos) do seu conjunto de treinamento tenham escalas muito diferentes. Qual técnica de regressão linear pode sofrer com isso e como? O que pode ser feito para mitigar este problema?
3. Suponha que você use o ***gradiente descendente em batelada*** e plote o erro de cada época. Se você perceber que o erro aumenta constantemente, o que provavelmente está acontecendo? Como você pode resolver isso?
4. Entre as versões do gradiente descendente (GD) que discutimos (batch, estocástico e mini-batch), qual deles chega mais rapidamente à vizinhança da solução ótima? Qual deles realmente converge? O que você pode fazer para que os outros também convirjam?
5. Em sala de aula, nós discutimos os 3 tipos de algoritmos baseados no gradiente descendente, batch, estocástico e mini-batch, porém, o código do mini-batch foi o único que não foi apresentado. Portanto, neste exercício eu peço que vocês:
    a. Implementem o algoritmo do mini-batch.
    b. Testem suas implementações com a seguinte versão ruidosa da função objetivo y = 2*x1 + 2*x2 + w, onde x1, x2 e w são M = 1000 valores retirados de uma distribuição aleatória Gaussiana normal padrão (i.e, com média 0 e variância igual a 1) e utilizando a função hipótese h = a1*x1 + a2*x2. O objetivo aqui é verificar que o algoritmo do mini-batch também se aproxima do valor ótimo (obtido com a equação normal) quando o valor do passo de aprendizagem é ajustado para seu valor ótimo. (**Dica**: plotem o gráfico das curvas de contorno com o histórico dos valores dos pesos)
    c. Plotem a superfície de erro, a superfície de contorno com os parâmetros a1 e a2 para cada iteração do mini-batch, e o gráfico de iteração versus erro,
    d. Encontrem manualmente (tentativa e erro) o valor ótimo do passo de aprendizagem (**Dica**: utilizem os gráficos da superfície de contorno com os parâmetros a1 e a2 para cada iteração do mini-batch e o gráfico de iteração versus erro para saber se aquele passo é o ótimo. Acessem os links abaixo para entender como vocês podem plotar os gráficos de contorno.),
        i. https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/contour_demo.html#sphx-glr-gallery-images-contours-and-fields-contour-demo-py
        ii. https://www.python-course.eu/matplotlib_contour_plot.php

e. Comparem os resultados do mini-batch com os resultados obtidos com o GD em batelada (batch) e GD estocástico (**Dica**: para a comparação, usem os códigos que estão nos slides da aula e plotem os gráficos da superfície de contorno com os parâmetros a1 e a2 para cada iteração, ou seja, o histórico de atualização dos pesos, e o gráfico de iteração versus o erro para GD em batelada e estocástico).
f. Baseando-se nos gráficos do item anterior, a que conclusões vocês podem chegar quanto ao treinamento dos 3 tipos de gradiente descendente?

6. Dada a seguinte função hipótese e assumindo o erro quadrático médio como função de erro

$$h(x) = a0 + a1*x + a2*x^2.$$

Encontre as equações de atualização dos pesos/parâmetros para esta função. Em seguida, utilizando os vetores x e y definidos abaixo, encontre os parâmetros a0, a1 e a2 através do método da regressão de forma fechada (equação normal) e com ***gradiente descendente em batelada***.

$$y(x) = 3 + 1.5*x + 2.3*x^2 + w,$$

onde x é um vetor coluna com M = 1000 valores retirados de uma distribuição aleatória uniformemente distribuída no intervalo de -5 a 5 e w é outro vetor coluna com M valores retirados de uma distribuição aleatória Gaussiana com média 0 e variância igual a 10.

a. Plote o gráfico do número de iterações versus o erro quadrático médio.
b. Baseado no gráfico acima, encontre manualmente o melhor valor para o passo de aprendizagem. (**Dica**: encontre empiricamente, ou seja, através de tentativa e erro, o valor do passo que faça com que o erro decresça rapidamente nas primeiras épocas e que após algum tempo se estabilize, se tornando quase constante.)

7. Neste exercício você vai utilizar o arquivo ***training.csv*** onde a primeira coluna são os valores de x (feature) e a segunda de y (label). Baixe o arquivo do endereço: training.csv. 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('training.csv', header=None)

x = df[0].to_numpy()
y = df[1].to_numpy()

fig = plt.figure(figsize=(10,10))
plt.plot(x, y, 'b.')
```

Em seguida, utilize o algoritmo do ***gradiente descendente em batelada*** para encontrar os parâmetros de cada uma das seguintes funções hipóteses.

a. h = a0 + a1*x (polinômio de ordem 1)
b. h = a0 + a1*x + a2*x^2 (polinômio de ordem 2)

c. h = a0 + a1*x + a2*x^2 + a3*x^3 (polinômio de ordem 3)
d. h = a0 + a1*x + a2*x^2 + a3*x^3 + a4*x^4 (polinômio de ordem 4)

Para cada uma das funções hipótese acima faça o seguinte:

a. Encontre os valores ótimos dos parâmetros através do método de forma fechada, i.e., equação normal, ou também conhecida como método dos mínimos quadrados.
b. Encontre as equações de atualização dos parâmetros/pesos assumindo o erro quadrático médio como função de erro.
c. Encontre o valor ótimo do passo de aprendizagem.
d. Plote um gráfico que mostre x vs. y e x vs. h, ou seja, um gráfico comparando os dados originais com a estimativa (i.e., hipótese) da função que gerou y.
e. Plote um gráfico com do número de iterações versus o erro quadrático médio.

Em seguida responda às seguintes perguntas

A. Qual das funções hipótese acima aproxima melhor a função alvo (target), ou seja, qual produz o menor erro ao final do treinamento?
B. Dado que você encontrou os parâmetros que otimizam cada uma das funções hipótese acima (ou seja, você agora tem um modelo treinado que pode predizer o resultado para novos exemplos (generalizar)), use os dados contidos no arquivo predicting.csv e calcule o erro quadrático médio para cada um dos modelos (i.e., função hipótese). Qual função hipótese resulta no menor erro quadrático médio de validação, ou seja, com os dados do arquivo predicting.csv? Crie um gráfico que mostre os erros de treinamento e validação versus a ordem do polinômio. O que você consegue concluir a respeito dos valores plotados?

8. Neste exercício você irá utilizar a esquema de decaimento programado do passo de aprendizagem conhecido como ***decaimento exponencial*** juntamente com o algoritmo do ***gradiente descendente estocástico*** com ***critério de parada*** definido como sendo quando a ***diferença absoluta*** entre o erro da iteração atual e a anterior caia abaixo de 0.001 ou que o número máximo de iterações tenha sido atingido. Defina o número máximo de iterações como

maxNumIter = n_epochs*M,

onde n_epochs = 1 e M = 1000. Dada a seguinte função observável

y_noisy = 2.5*x1 + 1.3*x2 + w,

onde x1, x2 e w são variáveis aleatórias seguindo a distribuição Gaussiana normal padrão, ou seja, com média zero e variância unitária. Gere M = 1000 exemplos a partir desta função observável. Agora faça o seguinte:

a. Plote a superfície de erro considerando a seguinte ***função hipótese***: h(x1,x2) = a1*x1 + a2*x2.
b. Encontre os valores ótimos dos pesos através da fórmula fechada, ou seja, a equação normal.
c. Treine o modelo utilizando apenas o algoritmo do ***gradiente descendente estocástico***.
d. Plote os seguintes resultados:

i. A ***superfície de contorno*** mostrando a evolução da atualização dos pesos.
ii. O gráfico de erro versus número de iterações necessárias para que o critério de parada seja atingido.
iii. O gráfico da variação da atualização (i.e., alpha*vetor_gradiente) versus o número de iterações necessárias para que o critério de parada seja atingido. Mostre a variação da atualização para cada peso em figuras separadas.

e. Agora, treine o modelo utilizando o algoritmo do ***gradiente descendente estocástico*** com os esquema de decaimento programado do passo de aprendizagem conhecido como ***decaimento exponencial***. A fórmula matemática deste esquema é dada por

$$\text{alpha} = \text{alpha\_initial} * \exp(-k * t),$$

Onde k é a taxa de decaimento da exponencial, t é o número da iteração e alpha_initial é o valor inicial do passo de aprendizagem. **OBS**.: Não se esqueça de encontrar os melhores valores (aqueles que façam com que o tempo de convergência e a oscilação do GDE diminuam) de alpha e k.

f. Plote todos os resultados listados no item (d) para esta implementação do gradiente descendente estocástico com decaimento exponencial.
g. Imprima os valores dos pesos encontrado com a equação normal, com o GDE e o GDE utilizando decaimiento exponencial.
h. Qual das duas versões precisa de menos iterações para que o critério de parada seja atendido?
i. Qual valor dos pesos obtidos pelo GDE e GDE com decaimento exponencial é próximo ao do obtido com a equação normal?

DICA: Para que você tenha o mesmo comportamento para ambas versões do algoritmo do GDE (sem e com o decaimento exponencial), “reinicie” o gerador de números pseudo-aleatórios antes do laço de repetição que implementa cada uma das versões do GDE, com a função seed no módulo random da biblioteca NumPy: np.random.seed(42).

9. Neste exercício você irá aplicar o ***escalonamento mín-max*** aos atributos de treinamento de um modelo de regressão linear. Dada a seguinte função observável

$$\text{y\_noisy} = x1 + x2 + w$$

onde x1 é um vetor coluna com M amostras retiradas de uma distribuição Uniforme no intervalo [-5.0, 5.0), x2 é um vetor coluna com M amostras retiradas de uma distribuição Uniforme no intervalo [-0.5, 0.5) e w é também um vetor coluna com M amostras retiradas de uma distribuição Gaussiana Normal com média 0 e variância unitária. Gere um conjunto de treinamento com M = 1000 exemplos. Utilize o algoritmo do **gradiente descendente em batelada** com a seguinte função hipótese

$$h(x1,x2) = a1*x1 + a2*x2,$$

com a1 e a2 iniciais iguais a -10 e -10, respectivamente.

a. Sem aplicar escalonamento de features aos exemplos de treinamento, plote a superfície de erro, a superfície de contorno com os parâmetros a1 e a2 encontrados durante as iterações (ou seja, o histórico de valores que o algoritmo

encontra durante o treinamento do modelo) para o conjunto de treinamento e o gráfico de erro quadrático médio versus o número de iterações. **OBS.1**: Não se esqueça de encontrar, manualmente, o valor ótimo para o passo de aprendizagem. Este deve ser o maior valor possível que não faça o algoritmo oscilar entre os vales. **OBS.2**: Não se esqueça também, de calcular o valor ótimo dos pesos, através da equação normal, e plotá-los no gráfico de contorno juntamente com o histórico dos pesos.

b. Aplique a ***normalização min-máx*** aos atributos de treinamento, plote a superfície de erro, a superfície de contorno com os parâmetros a1 e a2 encontrados durante as iterações e o gráfico de erro quadrático médio versus o número de iterações. **OBS.1**: Não se esqueça de encontrar, novamente, o valor ótimo para o passo de aprendizagem. Este deve ser o maior valor possível que não faça o algoritmo oscilar entre os vales. **OBS.2**: Não se esqueça também, de calcular novamente o valor ótimo dos pesos, através da equação normal, e plotá-los no gráfico de contorno juntamente com o histórico dos pesos.
c. Existe diferença entre os formatos da superfície de erro sem e com escalonamento?
d. Houve diminuição do número de iterações necessárias para a convergência?
e. Se a ***padronização*** fosse aplicada ao invés da ***normalização***, como ficaria o formato da superfície de erro e o número necessário de iterações para a convergência do algoritmo?

10. Neste exercício você irá aplicar ***escalonamento de features*** aos dados de treinamento e teste. Dada a seguinte função objetivo (ou modelo gerador)

$$y = x1 + x2,$$

onde x1 é um vetor coluna com M amostras retiradas de uma distribuição Gaussiana com média 0 e desvio padrão unitário e x2 é um vetor coluna com M amostras retiradas de uma distribuição Gaussiana com média 10 e desvio padrão igual a 10. Gere dois conjuntos de dados (aleatórios), com M = 1000 amostras cada. Um dos conjuntos será utilizado para treinamento e o outro para teste, ou seja, validação do modelo treinado. Ou seja, você deve criar um conjunto y_treinamento = x1_treinamento + x2_treinamento e outro conjunto y_validação = x1_validação + x2_validação. Use sementes diferentes para gerar os conjuntos de treinamento e validação (e.g., np.random.seed(42) e np.random.seed(84)). Utilize o **gradiente descendente em batelada** com a seguinte função hipótese

$$h = a1*x1 + a2*x2,$$

com a1 e a2 iniciais iguais a -20 e -20, respectivamente. Para todos os casos abaixo, treine os modelos com o mesmo número máximo de iterações, por exemplo, 2000 iterações e um critério de parada que faça o algoritmo parar quando o erro de treinamento entre duas épocas consecutivas for menor do que 0.001, ou seja, o algoritmo irá parar se o erro for menor do 0.001 ou se atingir o número máximo de iterações. Pede-se

a. Sem aplicar nenhum escalonamento de features aos exemplos de treinamento, plote a superfície de erro, a superfície de contorno com os parâmetros a1 e a2

encontrados durante as iterações (ou seja, o histórico de valores que o algoritmo encontra durante o treinamento do modelo) para o conjunto de treinamento e o gráfico de erro quadrático médio versus o número de iterações para os conjuntos de treinamento e teste. **OBS.1**: Não se esqueça de encontrar, manualmente, o valor ótimo para o passo de aprendizagem. **OBS.2**: Não se esqueça de encontrar o valor ótimo dos pesos e plotá-los no gráfico de contorno com o histórico dos pesos.

b. Aplique a normalização min-máx às features de treinamento e teste, plote a superfície de erro, a superfície de contorno com os parâmetros a1 e a2 encontrados durante as iterações para o conjunto de treinamento e o gráfico de erro quadrático médio versus o número de iterações para os conjuntos de treinamento e teste. **OBS.1**: Não se esqueça de encontrar o valor ótimo para o passo de aprendizagem. **OBS.2**: Não se esqueça que o conjunto de testes é normalizado com os valores mín-máx encontrados para o conjunto de treinamento. **OBS.3**: Não se esqueça de encontrar o valor ótimo dos pesos/parâmetros e plotá-los no gráfico de contorno com o histórico dos pesos.
c. Aplique a padronização às features de treinamento e teste, plote a superfície de erro, a superfície de contorno com os parâmetros a1 e a2 encontrados durante as iterações para o conjunto de treinamento e o gráfico de erro quadrático médio versus o número de iterações para os conjuntos de treinamento e teste. **OBS.1**: Não se esqueça de encontrar o valor ótimo para o passo de aprendizagem. **OBS.2**: Não se esqueça que o conjunto de testes é padronizado com os valores de padronização encontrados para o conjunto de treinamento. **OBS.3**: Não se esqueça de encontrar o valor ótimo dos pesos/parâmetros e plotá-los no gráfico de contorno com o histórico dos pesos.
d. Repita os itens b e c aplicando desta vez a normalização min-máx e a padronização, respectivamente, também aos targets/rótulos (ou seja, os valores de y).
e. Baseado nos resultados anteriores o que você pode concluir a respeito do escalonamento de features? (**Dica**: Comente a respeito das formas das superfícies de erro, dos números de iterações necessárias para se alcançar o ponto ótimo, isso se ele é alcançado, da diferença entre o erro quadrático médio obtido para o conjunto de treinamento e o obtido para o conjunto de testes (são similares ou diferentes), da diferença entre os valores do erro quadrático médio para os 3 casos acima, i.e., sem escalonamento e com os 2 tipos de escalonamento com e sem escalonamento dos labels (qual resulta no menor erro? Escalonar os labels traz algum benefício? Como ficam as superfícies de erro quando se escalona os labels?), e o que mais você achar interessante comentar. Quanto mais detalhada sua análise dos resultados, melhor será sua avaliação neste exercício.)

11. 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,
    i. Plote a superfície de erro.
    ii. Encontre os valores ótimos dos pesos da função hipótese utilizando a equação normal.
    iii. 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).
    iv. Plote a superfície de contorno mostrando o histórico de atualização dos pesos e seus valores ótimos.
    v. Plote o gráfico de erro versus número de épocas.
e. Quais os valores dos pesos encontrados pelo o algoritmo do gradiente descendente em batelada?