Trabalho Prático 1 Algoritmos II: Geometria computacional

Gabriel Eduardo Lemos dos Santos - 2021032056, Davi Fraga Marques Neves - 2020420575

Departamento de Ciência da Computação - Universidade Federal de Minas Gerais (UFMG) - Belo Horizonte – MG – Brazil

gabriellemos@ufmg.br

davifmn@ufmg.br

Resumo. O objetivo deste trabalho é realizar a implementação de algoritmos de envoltória convexa e verificação de interseção de segmentos para o problema de classificação em aprendizado de máquina supervisionado

1. Introdução

O objetivo deste trabalho é colocar em prática conteúdos aprendidos em sala de geometria computacional vistos de forma teórica sendo eles: algoritmos de envoltória convexa e verificação de interseção de segmentos para o problema de classificação em aprendizado de máquina supervisionado.

A tarefa, em suma, consiste em criar um modelo capaz de classificar pontos tomando como base primeiramente um conjunto de dados que vão alimentar o treinamento do programa e então ser capaz de definir a classe de outro pontos a partir dos parâmetros observados. Assim, para o problema da envoltória é necessário agrupar os pontos em um espaço n-dimensional em classes. O algoritmo utilizado para resolver esse problema será a "Varredura de Grahaam". Ele deverá dizer quais pontos estão dentro de uma envoltória e se há ou não sobreposição de envoltórias. Por fim, o programa deve ser capaz de encontrar os pontos mais próximos entre as duas envoltórias retornando a reta perpendicular ao ponto (a equação dessa reta).

Em seguida, o conjunto de dados separados para realizar os testes será submetido a um classificador que, irá classificá-lo em uma das classes, levando em conta o ponto classificado mais próximo do ponto que deseja-se avaliar.

Por fim, o conjunto de teste será avaliado utilizando algumas métricas, e assim será possível avaliar o quão preciso e coeso é o modelo implementado.

2. Bibliotecas importadas

As bibliotecas importadas neste trabalho foram numpy, math, matplotlib, pandas, seaborn, sklearn e ucimlrepo.

Numpy e math foram utilizadas basicamente para realizar operações matemáticas e de manipulação de estrutura de dados, como transformar listas em arrays numpy, cálculo de distância entre outros.

Matplotlib foi utilizada para plotar os gráficos do código, ou seja, as envoltórias, as retas, os pontos, entre outros, foram realizados por ela.

Pandas foi utilizada para realizar a leitura e tratamento de datasets alguns datasets.

Seaborn e Sklearn foram utilizadas em diversas partes do código, principalmente durante a parte da execução dos datasets. Seaborn possui alguns datasets dentro da própria biblioteca, assim, um dos datasets utilizados, era da própria biblioteca. Sklearn, também possui alguns datasets dentro de si, sendo um deles um dos mais conhecidos, o Iris.

Sklearn também foi utilizado em outras instâncias, como na separação das amostras entre treino e teste. De um modo geral, essa biblioteca realiza operações utilizando funções internas, facilitando o desenvolvimento do código.

Por fim, a biblioteca ucimlrepo é uma biblioteca própria do UC Irvine Machine Learning Repository, um site que possui um número grande e amplamente conhecido de datasets/repositórios. Dessa forma, com o uso da biblioteca deles, é possível realizar o uso dos seus datasets sem precisar de arquivos externos, apenas pelo ID do dataset é possível acessá-lo.

3. Envoltória convexa

Para a realização deste trabalho, busca-se realizar um algoritmo para criar modelos de classificação em aprendizado supervisionado. O problema de aprendizado supervisionado consiste em criar um modelo que, baseado em dados de um conjunto de treinamento, é capaz de representar as características principais dos dados e posteriormente atribuir o valor de uma variável alvo a dados para os quais se desconhece tal valor. Dessa forma, para realizar essa separação, foi utilizado o método da envoltória convexa, o qual consiste em agrupar o máximo de pontos possíveis , utilizando o menor número deles. De forma geral, busca achar os pontos que se encontram “na borda” de cada um dos conjuntos de classes apresentados.

3.1. Interseção de segmentos

Para que o código execute, inicialmente é necessário checar a detecção de interseções entre as envoltórias (classes) que estão presentes no dataset.

Assim, o código implementado realiza três passos para determinar se ocorre a interseção entre duas envoltórias. Primeiramente, é verificado se um ponto "q" está localizado na reta que passa pelos pontos "p" e "r", retornando True se "q" estiver na reta entre "p" e "r" e False caso contrário.

Após isso, é determinado se dois segmentos de linha, um definido por "p1" e "q1" e o outro por "p2" e "q2", se cruzam. Isso é feito por meio de um teste de orientação, verificando se os segmentos de fato se interceptam (verifica-se as coordenadas dos 4 pontos e avalia se ocorre ou não um cruzamento). Se houver interseção, a função retorna True e False em caso contrário.

Por fim, a utiliza as duas funções anteriores para verificar se duas envoltórias convexas, representadas por convex_hull1 e convex_hull2, se intersectam. Ela realiza essa verificação examinando todos os pares de segmentos de linha formados pelas bordas das envoltórias. Se algum par de segmentos se cruzar, a função retorna True, indicando que as envoltórias têm interseção. Caso contrário, a função retorna False, indicando que as envoltórias não se intersectam. Se ocorrer uma interseção entre as duas envoltórias o código realizará um Exception, resultando em um erro e consequentemente o código irá parar.

3.2 Cálculo dos segmentos

Após realizar a envoltória em ambos conjuntos, e verificar que não ocorre interseção entre eles, é necessário implementar um método de separabilidade linear.

Dessa forma, um algoritmo procura a menor distância entre os conjuntos, ou seja, os dois pontos com a menor distância entre os conjuntos, por meio da função closest_points, a qual recebe as duas envoltórias. Ao encontrar esses pontos, uma reta é traçada entre eles, ligando os dois conjuntos.

Por fim, encontra-se o ponto médio dessa reta e, a partir dele, traça-se uma perpendicular. Essa perpendicular divide as envoltórias, caso os dois conjuntos encontrem-se, cada um deles, completamente de um dos lados envoltória, ocorre uma separabilidade linear.

Além disso, juntamente com o cálculo da reta perpendicular,sua função da reta também é calculada e impressa.

3.3 *Plotar* o Gráfico

Utilizando da biblioteca matplotlib, o *plot* do gráfico é realizado. Dessa forma, são plotados as duas envoltórias, a reta que liga ambas envoltórias e a tangente a partir de seu ponto médio.

4. Tratamento dos dados

Os dados utilizados para execução do algoritmo foram obtidos de datasets que não eram bidimensionais. Além disso, era necessário separar amostras de treino e teste.

4.1 PCA (Principal Component Analysis)

O método escolhido e utilizado para transformar os dados em bidimensionais foi o PCA. Em termos simples, o PCA identifica as direções principais ao longo das quais os dados variam mais e projeta os dados originais nesses novos eixos. Isso permite reduzir a dimensionalidade do conjunto de dados, mantendo as informações mais significativas.

A biblioteca Sklearn possui o método PCA implementado, dessa forma foi necessário apenas realizar algumas adaptações para que fosse utilizado de forma correta.

4.1.1 PCA UCI

O PCA UCI é um método/função que realiza o PCA em um dataset UCI. Por meio do ID do dataset, juntamente com o nome da coluna alvo que desejamos analisar, esse método é realizado, retornando um dataset que já foi tratado, pronto para ser utilizado no restante do código.

4.2 Separação das amostras

Como descrito na documentação da atividade, os dados extraídos dos datasets deveriam seguir uma proporção de 70% para o treinamento do modelo, e 30% para que os testes fossem realizados.

Dessa forma, novamente com uso da biblioteca Sklearn, por meio da função train_test_split, os dados são separados seguindo a proporção descrita, retornando dois novos datasets, um de treino e um de teste.

5. Classificador

O classificador implementado para realizar os testes utiliza como parâmetro a distância do ponto que deseja-se classificar, com o conjunto mais próximo à ele. Em resumo, o código classifica e insere uma lista de pontos em dois conjuntos de dados, atribuindo a eles uma classe com base em quais conjuntos são mais próximos. O cálculo dessa distância ocorre por meio da distância euclidiana.

Nessa parte do código é demandado mais tempo de execução, pois é necessário passar por todos os pontos de ambos conjuntos para encontrar aquele mais próximo ao ponto que deseja-se atribuir uma classificação.

6. Modelo

Após passar pelas etapas de tratamento dos dados, o dataset que será usado para o treino (70% da amostra inicial), será passado por parâmetro para a função New_Model, onde será separado, usando suas classes, em dois conjuntos distintos. Após isso, são realizadas as duas envoltórias e a verificação de separabilidade linear conforme descrito nos métodos acima.

Assim, o gráfico será plotado com as duas envoltórias, a reta tangente ao segmento que liga os dois conjuntos, a equação da reta tangente, e a verificação se as envoltórias se cruzam ou não.

Não foi possível obter um modelo para os conjuntos que eram sobrepostos, ou seja, possuíam interseção.

7. Métricas

As métricas foram medidas por 3 parâmetros, sendo eles ‘precisão’, ‘revocação’, e ‘Escore F1’. Todas essas medidas são realizadas por meio de funções internas da biblioteca Sklearn.

Não foi possível obter as métricas dos conjuntos que eram sobrepostos, ou seja, possuíam interseção.

8. Datasets

Os datasets, os resultados obtidos pelas métricas, os gráficos gerados, estão disponíveis no notebook entregue, por isso, no relatório será apresentado apenas um breve resumo dos resultados, juntamente com uma pequena análise.

Os datasets que foram obtidos a partir do UC Irvine Machine Learning Repository foram lidos e tratados com a biblioteca própria deles (descrito acima). Alguns foram obtidos de bibliotecas do próprio python, como o Seaborn.

No notebook, bem como nas referências bibliográficas, é possível saber qual a fonte dos datasets.

8.1 Íris

O dataset "íris" é amplamente conhecido e pode ser encontrado em diversas fontes. Nesse código, ele foi obtido a partir da biblioteca Seaborn.

Por se tratar de um código "base", é fácil de trabalhar com ele. Dessa forma, ao obter suas métricas, todas foram 1.0, ou seja, um acerto de 100% para todos os casos.

8.2 Heart-Disease

O dataset contém informações sobre pacientes e sobre as condições de seus sistemas cardiovascular.

Para esse dataset foram utilizados 2 casos diferentes, um levando em conta o sexo do paciente e o outro o índice de BFS (fast blood sugar) no sangue. No entanto, em ambos casos o resultado foi o mesmo.

Por ser um dataset da área da saúde, a separação entre as classes, de uma forma precisa, é quase que impossível. Dessa forma, ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset.

8.3 Rice

O dataset contém informações sobre os tipos de arroz. Nesse dataset, ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset.

8.4 Câncer de Mama

O dataset foi obtido a partir da biblioteca Sklearn. Novamente ocorre o problema do dataset "Heart-Disease", pois, por se tratar de um dataset da área da saúde, a separação entre as classes, de uma forma precisa, é quase que impossível. Dessa forma, ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset.

8.5 Wine

Outro dataset obtido a partir da biblioteca Sklearn. No notebook é possível ver que o plot dos pontos no gráfico ocorre de forma "misturada", não sendo possível realizar uma separação por classes.

Dessa forma, ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset.

8.6 Penguins

O dataset "penguins" foi obtido a partir da biblioteca Seaborn. Assim como com o dataset Iris, uma espécie foi retirada (Chinstrap) e foram utilizados apenas indivíduos do sexo masculino (havia muita variação e gerava resultados ruins).

Assim, utilizando as duas espécies restantes, foi possível gerar um modelo.

8.7 Automobile

O dataset “automobile” contém informações sobre diversos modelos de automóveis, tais quais tipo do combustível, preço, tipo do chassi, entre outros.

Para esse dataset foram realizados dois testes, um levando em conta o tipo de combustível utilizado e outro avaliando se o carro possui ou não “turbo”.

Para ambos casos não foi possível gerar um modelo, tendo em vista que as envoltórias se cruzavam.

8.8 Glass

O dataset “glass” contém informações sobre a identificação de tipos de vidro. Assim como no dataset das sementes, existem diversas classes de vidro podendo portanto gerar diferentes resultados para quais tipos forem selecionados.

No notebook foram selecionados os tipos 2 e 5, e por serem linearmente separáveis, foi possível criar um modelo para essas 2 classes.

8.9 Letter

O dataset “letter” identifica a qual das letras do alfabeto, a letra que deseja-se classificar, mais se assemelha. Como abrange todas as letras do alfabeto, existem 26 classes distintas, no entanto, foram selecionadas duas, “A” e “Z”.

Com essas duas letras, ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset. Esse resultado poderia ser diferente dependendo das letras selecionadas para realização do modelo

8.10 Income

O dataset income agrupa, por renda, seus indivíduos. Para realizar o experimento, selecionamos as classes “male” e “female”, separando por gênero e gerando portanto duas classes.

Novamente ocorre a interseção entre as classes e por isso não é possível criar um modelo para esse dataset.

9. Conclusão

Pode-se concluir, pelas métricas obtidas, que o modelo tem uma taxa de acerto muita alta para os dados que são linearmente separáveis (acima de 90% em todos os casos). Como nesse trabalho, o algoritmo não conseguia realizar seu objetivo caso houvesse uma interseção entre os conjuntos, grande parte dos dados não puderam ser avaliados (é notável no plot dos gráficos o quanto suas classes se misturam).

No entanto, para aqueles que puderam ser avaliados, os números obtidos foram muito bons. Ainda assim, eles só eram possíveis uma vez que os datasets possuíam classes muito distintas, e que podiam ser linearmente separáveis, fato esse que, como visto em na maioria dos casos, não ocorre no mundo real.

10. Referências bibliográficas

https://www.python.org/

https://scikit-learn.org/

https://archive.ics.uci.edu/

https://seaborn.pydata.org/

https://chat.openai.com/