CI209 - Inteligência Artificial

Prof. Aurora Pozo
DInf - UFPR
2020/1

Especificação do segundo trabalho prático da disciplina [1]

Trabalho prático 2

Este trabalho é composto por duas partes:

- Implementar o algoritmo de treinamento do modelo Perceptron
- Comparar a sua implementação com o modelo de classificação *Multilayer Perceptron* implementado na biblioteca *scikit-learn*[2]

Você deverá baixar o código-base do trabalho e alterar dois arquivos: *my_perceptron.py* e *ml.py*

Ao final do trabalho, você deverá entender o conceito básico do funcionamento do perceptron e algumas de suas características em problemas de classificação.

Você poderá verificar o resultado da sua implementação utilizando os seguintes comandos:

```
$ python3 app_perceptron.py
$ python3 ml.py
```

[1] Elaborado por Bruno Henrique Meyer e Augusto Lopez Dantas (Prática em docência de Informática)

[2] https://scikit-learn.org/

Divisão do trabalho

Parte 1 *Implementação do perceptron*

Implemente a função *run_perceptron* no arquivo *my_perceptron.py*. A função será utilizada para atualizar os pesos do perceptron em cada época da execução do treinamento do modelo. Nos exemplos desse trabalho serão considerados apenas problemas de classificação binária, onde há apenas duas classes.

Os parâmetros aceitos pela função são:

- *weights*: lista que contém os pesos do perceptron. **O primeiro elemento** do vetor representa o **bias**.
- *data*: Matriz do tipo numpy array. Cada linha representa uma instância (ponto) com 1+N valores, onde N é a quantidade de atributos que serão fornecidos de entrada ao perceptron. O **primeiro elemento** de cada instância será **sempre 1.0** e serve para multiplicar pelo bias, o que simplifica a implementação das operações necessárias.
- *labels*: Vetor de tamanho N que contém o rótulo (0 ou 1) de cada instância. Os índices são relacionados aos índices das instâncias contidas no parâmetro *data*.
- *learning_rate*: Valor que deve ser multiplicado pelo tamanho do "passo" relacionado à atualização dos pesos

A função deverá retornar o vetor de pesos atualizado e a quantidade de erros (instâncias classificadas incorretamente) da época.

Para verificar o funcionamento de sua implementação, execute o comando:

```
$ python3 app_perceptron.py
```

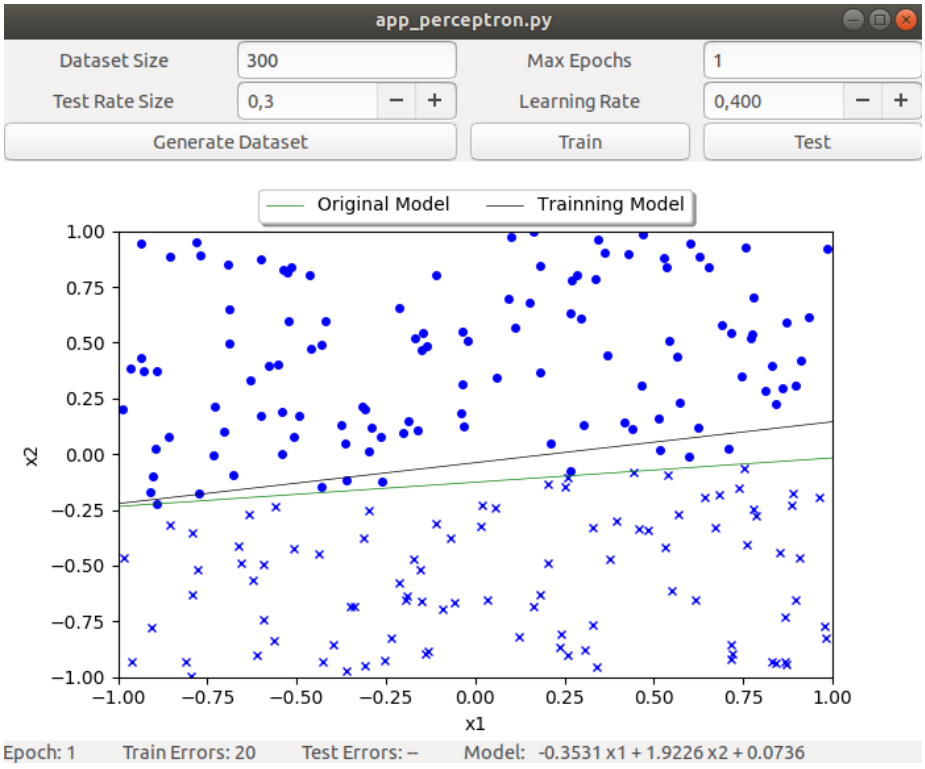

Figura 1: Visualização da execução do *app_perceptron.py*

Na Figura 1 é mostrada a interface para testar o perceptron. Você pode modificar os diferentes parâmetros e selecionar a opção "Train"para iniciar o processo de treinamento do modelo.

A linha verde representa a reta utilizada para gerar a distribuição dos pontos divididos em duas classes. O objetivo do perceptron é encontrar um conjunto de pesos e bias, representado pela linha preta, que seja o mais próximo possível da linha verde.

Parte 2 *Classificação*

O arquivo *ml.py* contém um código que carrega três diferentes bases de classificação. Você deverá utilizar a biblioteca *scikit-learn* para verificar a acurácia do classificador MLP (*Multilayer Perceptron*) nas três bases de dados.

Recomendamos que você use o MLP com diferentes parâmetros, como o número de camadas, usando valores diferentes dos padrões da biblioteca. No relatório você pode comentar se esses parâmetros tiveram impacto na acurácia do classificador.

As bases de dados são divididas na proporção de 3/4 para treinamento e 1/4 para teste. As bases de dados são:

- Iris: Dados reais que representam informações sobre o comprimento das pétalas de três tipos de flores. São 4 características e 150 instâncias. Neste trabalho, serão utilizadas somente 2 das classes e 100 instâncias.
- Artificial: Dados artificiais que contêm 1000 instâncias e 100 características.
- XOR: Dados artificiais de um problema clássico de classificação onde existem 4 agrupamentos de pontos divididos em 2 classes. A base contém duas características e 200 instâncias.

No código disponibilizado já existe uma função que reduz as dimensões de cada base de dados para a visualização (de forma aproximada) dos datasets. A Figura 2 ilustra a distribuição das instâncias de cada base.

Você também deverá analisar um classificador linear que utiliza sua implementação do perceptron, que também será disponibilizado. Seu uso é semelhante aos algoritmos de classificação do *scikit-learn*, e contém os métodos *fit*, *predict* e *score*. O classificador está disponível no arquivo *ml.py* cujo nome do modelo é *PerceptronClassifier*.

Ao final, seu programa deverá informar as acurácias nos conjuntos de teste obtidas em cada base de dados por cada modelo de classificação.

(a) Iris (b) Artificial (c) XOR

Figura 2: Datasets

Parte 3 *Conclusões e Relatório*

Crie um relatório de até 3 páginas reportando o desempenho dos classificadores sobre cada base de dados, também explicando os possíveis motivos para determinado modelo ter uma maior acurácia em determinada base de dados.

Se preferir, utilize figuras, tabelas entre outros recursos. Recomenda-se o uso do LaTeX para construir o seu relatório.

Dicas

- Utilize a biblioteca numpy[3] para facilitar as operações necessárias
- Consulte a documentação da biblioteca *scikit-learn* para verificar os exemplos de uso dos classificadores mencionados neste trabalho

Entrega

Você deverá entregar um arquivo compactado com o nome *login.tar.gz*, onde *login* é o nome do seu usuário no sistema do Departamento de Informática, que contenha os seguintes arquivos:

- *my_perceptron.py*
- *ml.py*
- *login.pdf*

Observações

Você deverá desenvolver e compreender todas implementações feitas no seu trabalho. Não serão admitidos quaisquer tipos de plágio.

[3] `https://numpy.org/`

Dúvidas

Dúvidas poderão ser retiradas por e-mail:

bhmeyer@inf.ufpr.br
aldantas@inf.ufpr.br