Redes Neurais Artificiais – Projeto Prático 2

Aprendizado Supervisionado no Neurônio Perceptron

Elloá B. Guedes

Escola Superior de Tecnologia
Universidade do Estado do Amazonas
Av. Darcy Vargas, 1200 – Manaus, AM
ebgcosta@uea.edu.br

2 de setembro de 2020

1. Apresentação

Neste segundo projeto prático, os alunos da disciplina de Redes Neurais Artificiais 2020.1 devem organizar-se em equipes de quatro estudantes. Os estudantes devem dispor suas matrículas e efetuar a soma do último dígito das mesmas, aplicando a operação de resto da divisão por 4 ao resultado. Este valor obtido será o identificador dos exemplos que a equipe irá utilizar ao longo do desenvolvimento.

Neste projeto prático, o objetivo é implementar o algoritmo de treinamento mediante Aprendizado Supervisionado do neurônio Perceptron de Rosenblatt aplicado em problemas de classificação. Para tanto, cada equipe deverá elaborar Jupyter Notebooks com o código-fonte deste algoritmo de treinamento desenvolvido na linguagem de Programação Python e fazendo uso das bibliotecas numpy, random, math e matplotlib. Em particular, a biblioteca numpy será de uso **obrigatório** para todas as operações de natureza matricial (multiplicação de matrizes, produto escalar, etc). Neste projeto prático, a biblioteca sci-kit learn **não** deve ser utilizada.

Serão avaliados os seguintes artefatos:

1. **Repositório no GitHub**. De caráter público (privado durante o desenvolvimento, público na ocasião da entrega), incluindo todos os integrantes da equipe e contendo as evidências de código e de progresso, registrando o trabalho dos múltiplos integrantes;

2. **Código-Fonte Python**. Deve ser incluído no repositório sob a forma de um único ou múltiplos Jupyter Notebooks, contendo todas as atividades de programação consideradas para atender ao que se pede no projeto. As células devem demonstrar o resultado da execução do código.

2. Detalhamento da Atividade

A atividade será dividida em três partes, cada uma a ser apresentada a seguir. Todas as atividades tem como entrada um arquivo `txt` com conteúdo em binário descrevendo um `numpy.ndarray` salvo previamente contendo múltiplos exemplos de dimensões $(1,3)$, ou seja, tem dimensões $(m,1,3)$, em que m varia a depender do documento considerado, de 800 a 1000, em média. Cada exemplo representa um ponto no $\mathbb{R}^2$ e o seu respectivo rótulo, isto é, tem-se (x_1,x_2,y_d). Os valores de y_d correspondem às classes discretas e binárias 0 e 1, em que a classe 0 deve ser denotada na cor vermelha e a classe 1 deve ser denotada na cor azul.

2.1. Parte I – Resolvendo um Problema Linearmente Separável

Nesta parte, todas as equipes devem usar o arquivo `dataAll.txt` e construir o algoritmo de treinamento do neurônio perceptron para resolver o problema de classificação proposto. Alguns aspectos devem ser considerados:

1. As equipes devem utilizar a função de ativação degrau com $\vartheta = 0$;
2. O valor da taxa de aprendizado deve ser igual a $\eta = 0{,}1$;
3. O vetor inicial de pesos deve ter seus valores inicializados conforme uma variável aleatória de distribuição uniforme no intervalo, isto é, $w_i \sim U(-0{,}5, +0{,}5)$. O vetor inicial de pesos deve ser impresso no início da execução do algoritmo;
4. A cada época deve ser indicado o número de ajustes feitos no vetor de pesos;
5. Sempre que o vetor de pesos for ajustado, este deve ser impresso;
6. O algoritmo deve executar até a convergência, isto é, até que não haja erros para todos os exemplos presentes no conjunto de treinamento;
7. Ao final, deve-se imprimir:
   (a) O número total de ajustes no vetor de pesos;
   (b) O número de épocas até a convergência;
   (c) O gráfico contendo todos os exemplos do conjunto de dados e a reta que separa as classes obtida como resultado do treinamento do neurônio Perceptron. Respeitar o esquema de cores proposto inicialmente e apresentar a solução de maneira clara neste gráfico.

2.2. Parte II – Experimentação

Nesta segunda parte, cada equipe deverá usar o seu respectivo identificador de exemplos para trabalhar com um arquivo específico. Por exemplo, se a equipe tem identificador 3, deve considerar o arquivo `data3.txt`.

A equipe deve aproveitar o algoritmo construído na Parte I e executar 100 repetições do mesmo para as seguintes configurações: $\eta \times I = \{0{,}4, 0{,}1, 0{,}01\} \times \{(-100, +100), (-1, +1), (-0{,}5, +0{,}5)\}$ em que I é o intervalo a ser utilizado para a distribuição uniforme do valor dos pesos. Assim, há 9 configurações a serem testadas, cada uma delas com 100 iterações.

Para cada configuração, deve-se apresentar um único gráfico contendo as entradas e a solução obtida, para mostrar que todas as configurações, ainda que distintas, levam à convergência. Respeitar as sugestões de ilustração indicadas anteriormente.

Para cada configuração em suas 100 execuções, obter a média e o desvio padrão da quantidade de ajustes efetuados no vetor de pesos e do número de épocas até a convergência. Dispor tais resultados sobre a forma de uma tabela ou gráfico e discutir se há uma configuração melhor ou pior que as demais ou se elas são equivalentes, considerando também os recursos computacionais utilizados.

2.3. Parte III – Validação *Holdout* em Problema Não-Linearmente Separável

Todas as equipes devem considerar o arquivo `dataHoldout.txt` e apresentar um gráfico inicial que evidencie que este problema não é linearmente separável.

Em seguida, os exemplos devem ser aleatoriamente divididos em duas partições, uma delas contendo 70% dos exemplos (treinamento) e outra contendo 30% (teste). Embora o problema não seja linearmente separável, vamos utilizar os dados de treinamento para obter uma reta de separação das classes com o neurônio Perceptron (solução possível). O neurônio em questão tem função de ativação degrau com $\vartheta = 0$ e os valores de η e de inicialização de pesos devem seguir as recomendações da literatura. Execute o algoritmo por 100 épocas, mas a cada época apresente os exemplos disponíveis com conjunto de treinamento em ordem aleatória.

Efetue a previsão da saída deste neurônio para todos os exemplos do conjunto de teste, comparando-a com a saída desejada e responda ao que se pede:

1. Apresente a matriz de confusão das previsões efetuadas para o conjunto de testes;
2. Qual a acurácia da solução proposta para os dados do conjunto de treinamento inicialmente fornecido?
3. Nos mesmos termos da questão anterior, obtenha os valores de precisão, revocação e F-*Score*;
4. A partir destas métricas, discorra acerca da qualidade desta solução perante o conjunto de testes.

Apresente dois gráficos com a solução obtida pelo neurônio Perceptron, mas um deles contendo os dados de treinamento e o outro contendo os dados de teste. Disponha tais gráficos lado a lado.

3. Tecnologias e Sugestões

Para a realização desta tarefa, é obrigatório o uso da linguagem de programação Python 3.6+ e das bibliotecas `pandas`, `numpy`, `random` e `matplotlib`. Soluções que fizerem uso de listas para manipula-

ção dos exemplos e cálculos matriciais são desencorajadas, devendo ser completamente evitadas.

Para aquelas equipes com restrições de *hardware*, recomenda-se o uso do Google Colab, disponível em: <http://colab.research.google.com/>. O notebook produzido ao final deve ser incluído no repositório GitHub. Para as demais equipes, recomenda-se o uso do gerenciador de pacotes Anaconda e a utilização de ambientes virtuais `conda env`.

4. Critérios de Avaliação

Os critérios de avaliação levarão em conta a organização do repositório, a qualidade do código produzido, a completude das tarefas solicitadas, a documentação, a qualidade textual das respostas em relação ao seu conteúdo e em termos de utilização da norma culta, coesão, coerência, o respeito aos prazos e a colaboração da equipe na elaboração do projeto.

5. Links Úteis

- <https://numpy.org/>
- <https://numpy.org/doc/stable/>
- <https://matplotlib.org/3.3.0/api/_as_gen/matplotlib.pyplot.scatter.html>
- <https://jakevdp.github.io/PythonDataScienceHandbook/04.02-simple-scatter-plots.html>