Classificação e Regressão
Resgate de Vítimas de Catástrofes Naturais, Desastres ou Grandes Acidentes

Lucas Pujol de Souza[1], **Saulo Bergamo**[1]

[1]Departamento de Informática (DAINF) – Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brazil

Abstract. *This paper describes a solution implemented for the development of a regression and classification solution for the rescue of victims of natural disasters, disasters or major accidents. The techniques of decision tree and fuzzy algorithm for classification and neural network for regression were used. The results were satisfactory in two of the solutions, but it needs improvements in the fuzzy logic.*

Resumo. *Este artigo descreve uma solução implementada para o desenvolvimento de uma solução de regressão e classificação para o resgate de vítimas de catástrofes naturais, desastres ou grandes acidentes. Foram utilizadas as técnicas de árvore de decisão e algoritmo fuzzy para classificação e rede neural para regressão. Os resultados foram satisfatórios em duas das soluções, mas necessita de melhorias na lógica fuzzy.*

1. Introdução

O problema proposto é a classificação e regressão para o resgate de vítimas de catástrofes naturais, desastres ou grandes acidentes. O objetivo é utilizar técnicas de aprendizado de máquina para analisar os sinais vitais das vítimas e determinar a gravidade do estado de saúde delas, uma vez que o corpo médico construiu uma fórmula para calcular a gravidade do estado de saúde das vítimas e, também, estabeleceram intervalos que definem as seguintes classes de gravidade mas a fórmula de cálculo e os intervalos de gravidade foram perdidos. Para a resolução deste impasse, utilizou-se a comparação de duas técnicas de classificação (Árvores de decisão e Fuzzy) e a realização de regressão utilizando Redes Neurais. O desempenho será avaliado por meio de métricas apropriadas para cada tipo de tarefa, como RMSE, precisão, recall, f-measure, acurácia e matriz de confusão. Esse problema é importante de ser investigado, uma vez que o uso do aprendizado de máquina pode ajudar em situações de predição e determinação em casos como o simulado nesta tarefa.

2. Fundamentação Teórica

A construção da solução necessita da definição de alguns termos e algoritmos que foram utilizados ou cogitados para a implementação:

2.1. Árvore de Decisão

Uma árvore de decisão é um modelo de aprendizado de máquina utilizado para tomar decisões ou classificar dados com base em uma série de condições. Essa estrutura é construída a partir de um conjunto de regras de decisão hierárquicas que dividem os dados em

grupos menores e mais específicos. No início, todos os dados são considerados como um único grupo. O algoritmo de árvore de decisão seleciona a melhor variável para dividir o conjunto de dados em dois subconjuntos menores. Após isso, o algoritmo cria um nó de decisão correspondente à regra escolhida para dividir os dados. Cada nó representa uma condição ou uma pergunta sobre uma variável específica. Com base na resposta à pergunta do nó, os dados são divididos em subconjuntos menores. Cada subconjunto é roteado para um nó filho correspondente. Os passos de criação e ramificação são repetidos para os subconjuntos resultantes em cada nó filho. Isso cria uma estrutura hierárquica de nós de decisão, onde cada nó representa uma condição que leva a um resultado específico. O processo de divisão é repetido até que certas condições sejam atendidas. Isso pode incluir critérios como profundidade máxima da árvore, número mínimo de amostras em um nó ou pureza mínima. Quando essas condições são alcançadas, os nós terminais da árvore, chamados de folhas, são criados. Cada folha representa uma decisão ou uma classe final para a amostra. Para classificar uma nova amostra, a árvore de decisão percorre os nós com base nas respostas às perguntas e condições em cada nó. Eventualmente, a amostra alcançará uma folha que determina sua classificação ou decisão final [Kingsford and Salzberg 2008].

2.2. Lógica Fuzzy

A lógica fuzzy, também conhecida como lógica difusa, é uma extensão da lógica booleana tradicional que permite lidar com incerteza e imprecisão. Ela é baseada no conceito de conjuntos fuzzy, nos quais os elementos podem ter pertinência parcial a um conjunto. Em vez de ter apenas conjuntos que são binários (pertencem ou não pertencem), a lógica fuzzy permite que os conjuntos tenham graus de pertinência. No lugar de dizer que um elemento pertence completamente ou não pertence a um conjunto, ele pode pertencer parcialmente com um grau de pertinência entre 0 e 1. Na lógica fuzzy, as variáveis podem ser definidas linguisticamente. Em vez de usar valores numéricos precisos, as variáveis podem ser representadas por termos linguísticos, como ”alto”, ”médio”e ”baixo”. Cada termo linguístico é associado a um conjunto fuzzy correspondente. Cada conjunto fuzzy é definido por uma função de pertinência, que atribui um grau de pertinência a cada elemento em relação a esse conjunto. Essa função mapeia os valores da variável de entrada para graus de pertinência. A lógica fuzzy utiliza operadores fuzzy para combinar conjuntos fuzzy e realizar operações lógicas. Os operadores fuzzy mais comuns são a união, interseção e negação fuzzy. Esses operadores consideram os graus de pertinência dos elementos nos conjuntos fuzzy e produzem um novo conjunto fuzzy resultante. A inferência é realizada utilizando regras fuzzy, que são declarações condicionais que relacionam as variáveis linguísticas e seus conjuntos fuzzy correspondentes. As regras fuzzy são baseadas em lógica condicional e são escritas na forma ”Se A, então B”, onde A e B são declarações fuzzy. Durante a inferência, as regras fuzzy são combinadas para derivar conclusões fuzzy. Após a inferência, as conclusões fuzzy precisam ser convertidas em valores numéricos precisos. Isso é feito através do processo de defuzzificação, que mapeia o conjunto fuzzy resultante em um valor numérico específico [Chen et al. 2001].

2.3. Redes Neurais

Uma rede neural é um modelo computacional inspirado pelo funcionamento do cérebro humano. Ela é composta por um conjunto interconectado de unidades de processamento chamadas de neurônios artificiais ou unidades de processamento, que operam em paralelo

para realizar tarefas de aprendizado e tomada de decisão. A rede neural é organizada em camadas de neurônios, geralmente divididas em três tipos: camada de entrada, camadas ocultas (intermediárias) e camada de saída. A camada de entrada recebe os dados de entrada e a camada de saída produz os resultados finais. As camadas ocultas fornecem a capacidade de aprendizado e extração de recursos. Cada conexão entre os neurônios é associada a um peso numérico que representa a força da conexão. Os pesos são ajustáveis e são inicialmente definidos aleatoriamente. Além disso, cada neurônio possui um valor de bias que representa um deslocamento na ativação do neurônio. Os pesos e biases são os parâmetros que a rede neural aprende durante o processo de treinamento. A informação flui através da rede neural em um processo chamado de propagação direta (feedforward). Os dados de entrada são alimentados na camada de entrada, que passa os sinais para a camada oculta, e assim por diante, até alcançar a camada de saída. Em cada camada, os neurônios combinam os sinais de entrada ponderados pelos pesos e aplicam uma função de ativação para produzir uma saída. Cada neurônio aplica uma função de ativação à sua entrada para determinar sua saída. A função de ativação introduz não-linearidade na rede neural e permite que ela aprenda relações complexas entre os dados. A saída da rede neural é comparada com os resultados esperados, e é calculada uma medida de erro para avaliar o desempenho da rede. Essa medida de erro é geralmente uma função que compara a saída da rede com os valores desejados. A retropropagação do erro (backpropagation) é um algoritmo utilizado para ajustar os pesos e biases da rede neural com base no erro calculado. O algoritmo propaga o erro da camada de saída para trás, ajustando os pesos em cada camada para minimizar o erro. Isso é feito utilizando técnicas de otimização, como o gradiente descendente, para ajustar os pesos de forma iterativa. O processo de retropropagação do erro é repetido por várias iterações ou épocas, com o objetivo de minimizar o erro e ajustar os pesos e biases para melhorar o desempenho da rede. Durante o treinamento, a rede neural aprende a mapear os padrões dos dados de entrada para as saídas desejadas, ajustando gradualmente seus parâmetros. Após o treinamento, a rede neural pode ser usada para fazer previsões ou classificar novos dados de entrada. A informação flui através da rede neural usando os pesos ajustados e as funções de ativação para produzir resultados [Krogh 2008].

3. Metodologia

A solução apresentada foi elaborada na linguagem Python. O menu inicial apresenta para o usuário uma escolha entre um dos três algoritmos: dois de classificação (árvore de decisão e fuzzy) e um de regressão (rede neural). O código pode ser acessado através do link: https://github.com/saulobergamo/sort_and_regress.

3.1. Classificação

3.1.1. Árvore de decisão

Através do uso da biblioteca Scikit-learn para treinar um classificador de árvore de decisão e fazer predições, o código carrega dados com rótulos e sem rótulos, cria uma árvore de decisão usando Scikit-learn, treina a árvore com os dados de treinamento, faz predições nos dados sem rótulos, imprime os resultados e salva-os em um arquivo. Além disso, ele exibe a árvore de decisão plotada.

3.1.2. Fuzzy

Com o uso da biblioteca Scikit-fuzzy para construir um sistema de controle fuzzy utilizando lógica de conjuntos fuzzy e inferência baseada em regras, o algoritmo carrega os dados de treinamento e teste, cria variáveis linguísticas para os atributos de entrada e saída, define regras fuzzy, cria um sistema de controle fuzzy e faz predições fuzzy para os dados de teste. As previsões são escritas em um arquivo de saída.

3.2. Regressão

3.2.1. Rede Neural

O algoritmo realiza regressão neural usando a biblioteca Scikit-learn. Um objeto MLPRegressor é criado para representar o modelo de regressão neural. É especificado o número e o tamanho das camadas ocultas usando o parâmetro hidden_layer_sizes. Outros parâmetros como a função de ativação (activation), número máximo de iterações (max_iter), taxa de aprendizado (learning_rate), e taxa de aprendizado inicial (learning_rate_init) também são definidos. Os valores previstos são formatados como strings com 4 casas decimais e o ponto decimal é substituído por uma vírgula. Os resultados são salvos em um arquivo de saída.

4. Resultados e análise

4.1. Árvore de Decisão

No algoritmo de árvore de decisão, a solução implementada obteve uma acurácia de 92,12%, acima da média de todas as soluções apresentadas (78,68%).

4.2. Fuzzy

Na solução fuzzy, o resultado foi uma acurácia de 56.06%. A média geral de todas as soluções foi de 67,21%, indicando um desempenho abaixo do esperado.

4.3. Rede Neural

Para a avaliação do resultado da regressão por rede neural, foi utilizada a métrica de RMSE (Root Mean Square Error). Para calcular o RMSE, primeiro calcula-se o erro quadrático para cada ponto de dados, que é a diferença entre o valor previsto e o valor real, elevada ao quadrado. Em seguida, calcula-se a média desses erros quadráticos. Por fim, tira-se a raiz quadrada dessa média para obter o RMSE. O RMSE é uma medida de dispersão dos erros do modelo, ou seja, quanto menor o valor do RMSE, melhor é o ajuste do modelo aos dados observados. Ele fornece uma estimativa da magnitude média dos erros de previsão e é expresso na mesma unidade da variável de resposta. Na solução apresentada, o RMSE foi de 3,89. Esse valor está bem abaixo da média de todas as soluções (6,19).

5. Conclusões

Os algoritmos implementados apresentaram um ótimo resultado. Entretanto, a solução fuzzy ficou abaixo da média de outras soluções, indicando uma possível área de melhoria. Em uma outra oportunidade, poderiam ser avaliadas estratégias como analisar os conjuntos fuzzy usados no algoritmo, verificar se as regras fuzzy estão completas e bem definidas para o problema em questão ou revaliar a técnica de defuzzificação utilizada.

Referências

Chen, G., Pham, T. T., and Boustany, N. (2001). Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems. *Applied Mechanics Reviews*, 54(6):B102–B103.

Kingsford, C. and Salzberg, S. L. (2008). What are decision trees? *Nature biotechnology*, 26(9):1011–1013.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 26(2):195–197.

6. Apêndice

Para executar o código, siga as instruções abaixo:

Tenha python3 e pip instalado.

Execute o comando ”python3 main.py”sem as aspas.

Caso exista uma dependencia inexistente, execute ”pip install -dependência pendente-”, sem aspas. Substitua o termo -dependência pendente- por qual for necessária.

As dependências necessárias são:

python versão 3.8

matplotlib

sklearn

csv

pandas

skfuzzy