Universidade de São Paulo – USP São Carlos
Instituto de Ciências Matemáticas e de Computação
SCC5809 – Redes Neurais – Profa. Dra. Roseli Romero
Damares Oliveira de Resende – #11022990

Relatório de Atividades – Exercício 3

Nesse exercício foram criadas duas redes neurais diferentes, uma MLP (Multilayer Perceptron) e uma RBF (Radial Basis Function). Ambas redes foram testadas para uma aplicação de classificação não linear na base de dados "Seeds" do repositório UCI.

Pré-processamento da base de dados

Como primeiro passo, a base de dados foi tratada para posteriormente alimentar a rede. Criou-se então o script *NNDataset.py*. Esse script contém funções para carregar a base de dados do repositório online, normalizar os dados e binarizar a saída. O processo de binarização consiste em transformar os rótulos do domínio dos inteiros para o domínio binário. Dessa forma, o rótulo 1 corresponde ao vetor de [1 0 0], o rótulo 2 corresponde ao vetor [0 1 0] e o rótulo 3 ao vetor [0 0 1]. Isso é feito pois a saída da MLP, que é uma sigmoide, está limitada de 0 a 1.

Essa classe também possui um método para separar o conjunto de treinamento e teste de forma estratificada. Assim, a proporção das classes será mantida quando a base de dados for dividida. Essa divisão é feita de forma aleatória. Primeiramente a base é mapeada em 3 diferentes conjuntos, um para cada classe, depois os exemplos em cada conjunto são embaralhados, e por último são divididos em um conjunto de treino e teste. A base de dados ''Seeds" é completamente balanceada e possui 70 exemplos para cada classe. No script implementado 20% dela é usado para teste e 80% para treino, onde o conjunto de teste possui 42 exemplos, 14 de cada classe; e o conjunto de treino possui 168 exemplos, 56 para cada classe.

Implementação dos modelos

O próximo passo consiste em definir a arquitetura das redes. Os modelos estão implementados no script *NNModels.py*. A MLP foi implementada com duas camadas escondidas. A função de ativação é uma sigmoide e o processo de backpropagation é baseado em dois parâmetros externos: o momento, definido em 0.75; e a taxa de aprendizado, definida em 0.3. A primeira camada escondida possui 15 neurônios e a segunda 5. Escolheu-se usar duas camadas pois é sabido que a base de dados não possui uma classificação linear, logo uma MLP de apenas uma camada possivelmente não seria suficiente para classificar os exemplos. A rede roda por 20000 épocas e não possui outra condição de parada.

A RBF foi desenvolvida simplificadamente. A função de ativação é uma gaussiana e os protótipos são escolhidos aleatoriamente para cada classe. Primeiro, os exemplos são clusterizados por classe e depois os exemplos protótipos são aleatoriamente selecionados. Sabe-se que para obter uma performance melhor seria necessário codificar um *kmeans* para definir os centros de forma mais precisa. Porém, sabe-se também que se houverem neurônios suficientes na camada escondida, mesmo que os centros estejam errados, após a entrada ser superdimensionada, baseado no cálculo do spread e na definição da similaridade da entrada com os protótipos, é possível fazer uma classificação razoável. O método de backpropagation para essa rede não foi desenvolvido. Ela é treinada por apenas uma época e mesmo assim consegue classificar bem os exemplos. A RBF foi treinada com 20 protótipos para cada classe, o que totaliza 60 neurônios na camada escondida.

Resultados e conclusão

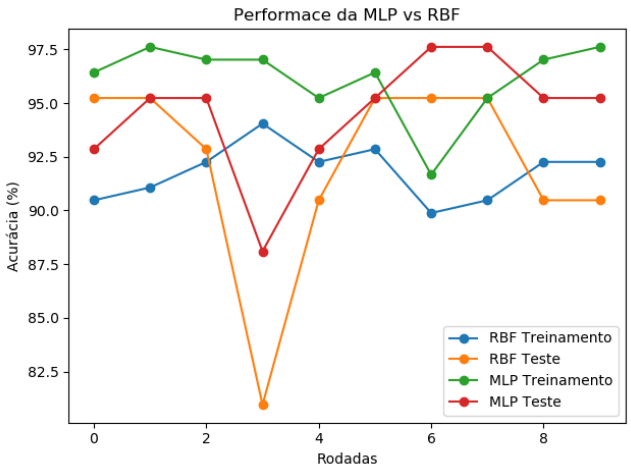

Figura 1: Comparação de Performance

O resultado dos testes é mostrado na Figura 1. O algoritmo foi executado 10 vezes para cada rede. Para cada execução a base de dados foi dividida em treino e teste de forma aleatória. As mesmas bases usadas na RBF são usadas na MLP. A partir da Figura 1 observa-se que o treinamento da MLP possui uma acurácia muito maior que o treinamento da RBF em todas as rodadas. Porém, no teste, a MLP não é superior em todos os casos. Nas rodadas 1 e 5 há um empate e na rodada 0 a RBF possui uma performance melhor. Isso sugere que a RBF é melhor para a generalização de dados que a MLP e consegue classificar mais precisamente dados que nunca foram observados. Outro ponto importante a considerar é o fato de a MLP ter sido implementada com mais cuidado. Ela possui o algoritmo de backpropagation para ajustar os pesos e é executada 2000 mil vezes, enquanto a RBF simplificadamente escolhe de forma aleatória os protótipos e é executada apenas uma vez, ou seja, sem ajuste dos pesos. Possivelmente, se a RBF tivesse sido implementada com todas as propriedades, ela teria uma performance melhor que a MLP.