Universidade Tecnológica Federal do Paraná
Especialização em Ciência de Dados
Métodos e Modelos Avançados em Ciência de Dados
"AT 02 - MLPs"
Prof. Dr. Rafael Gomes Mantovani

Conforme discutido em aula, as Redes Neurais Artificiais, mais precisamente o Perceptron Multicamadas (*Multilayer Perceptron* - MLP), tem a capacidade de generalização de uma função com alto nível de precisão. No entanto, problemas como *overfitting* e *underfitting* ainda podem ocorrer e comprometer o desempenho e a indução adequada de um modelo preditivo.

Objetivo da atividade

Objetivo desta atividade é então avaliar as implementações existentes de MLP em problemas reais, cujos dados podem ser encontrados em repositórios de dados públicos. Além disso, também deseja-se compreender melhor o comportamento do algoritmo.

1 O que fazer?

Encontre uma implementação de MLP adequada na literatura (R/Python) e avalie o desempenho deste algoritmo em algum dataset de sua escolha. Para a seleção de datasets, eu recomendo o uso do OpenML (https://www.openml.org), repositório público de datasets e experimentos com algoritmos de Aprendizado de Máquina. O OpenML tem mais de 3000 datasets (https://www.openml.org/search?type=data) que podem ser filtrados e usados livremente. Caso queiram usar algum outro conjunto de dados do UCI, Kaggle e afins, fiquem à vontade para explorá-los também.

Com o dataset selecionado, execute alguns testes/experimentos para avaliar o algoritmo no problema escolhido. Lembrem-se que definir a topologia da rede (número de neurônios nas camadas) é uma premissa para a construção de uma MLP adequada. Além disso, a escolha da taxa de aprendizagem se mostra importantíssima para a verificação da convergência do algoritmo.

Como trabalho, elabore um relatório técnico (documento pequeno, .pdf) que descreva as etapas realizadas:

- explicar o dataset escolhido;
- mostrar algumas estatísticas interessantes do problema, e justificar porque ele foi escolhido como objetivo de investigação;
- explicar a implementação de MLP escolhida, seus hiperparâmetros, e características gerais de uso;
- descrever a metodologia experimental: como os dados foram manipulados entre treino e teste, qual medida de desempenho foi usada, e quais tipos de investigações você realizou;
- apresentar suas conclusões com relação aos resultados obtidos, e descrever algumas possíveis limitações/vantagen do algoritmo no problema escolhido.

2 Algumas dicas

- MLPs funcionam com datasets que são totalmente formados por dados numéricos. Se eventualmente o dataset escolhido possuir algum atributo categórico, faça o pré-processamento que achar conveniente para deixá-lo com valores numéricos;
- alguns autores sugerem também realizar a normalização dos atributos. Se achar interessante, realize testes com os dados originais e com os dados normalizados. Existe algum benefício em normalizar os dados no seu problema?

- Muitas vezes é mais fácil avaliar/treinar os algoritmos dividindo os dados em apenas duas partições: treino e teste. Chamamos esse processo de *holdout*. Geralmente usamos 60, 70% dos dados no treinamento e o restante no teste. Embora seja simples, esse processo pode apresentar uma alta variância nos resultados. Isto é, partindo da premissa que selecionamos aleatoriamente quem são os exemplos de treino e teste, uma segunda amostragem seguindo a mesma estratégia poderia originar em resultados diferentes (melhores ou piores). Caso queiram explorar, isso é livre, vocês podem explorar a avaliação por validação cruzada (*k-fold cross validation*). Os principais *frameworks* de Aprendizado de Máquina (ver seção seguinte) já possuem funções implementadas que realizam esse processo. Na validação cruzada, os dados são dividido em k partições similares, e todas elas são usadas ao menos uma vez como conjunto de teste. Ou seja, repetimos a criação do modelo (treinar o algoritmo) k vezes. O desempenho final do algoritmo é a média dos valores obtidos em todas as iterações;

- uma análise muito interessante que vocês podem explorar é executar o algoritmo com diferentes quantidade de neurônios na camada oculta. O problema se resolve com mais ou menos neurônios nas representações intermediárias. O mesmo pode ser pensado com a taxa de aprendizagem: valores maiores podem convergir mais rápido (ou se perder no processo), enquanto valores menores requerem mais épocas para conseguir classificar os dados. A taxa de aprendizagem com certeza será um valor sensível a escolha do valor.

- vocês podem apresentar e discutir esses resultados por meio de tabelas, mas o uso de figuras e gráficos é um diferencial, pois estamos falando de ciência de dados :)

3 Alguns links que podem ser úteis

Após muitas instruções, agora entra a parte do *hack* :) Existem algumas ferramentas e links que podem ajudar muito nessa atividade:

- **mlr3** - *Machine Learning in R*: *framework* em R para implementação de soluções de Aprendizado de Máquina. Ele é bem completo, quase tudo já está implementado, o problema pode ser entender como encaixar cada peça. Link: https://mlr3.mlr-org.com;

- **nnet** - pacote em R para MLPs. https://cran.r-project.org/web/packages/nnet/index.html

- **RSNNS** - pacote em para MLPs. https://cran.r-project.org/web/packages/RSNNS/index.html

- **deepnet**: pacote em R para MLPs. https://cran.r-project.org/web/packages/deepnet/index.html

- **scikit learner** - *Machine Learning in Python* - contapartida do mlr mas em Python. https://scikit-learn.org/stable/

- **Perceptron** - implementação do scikit learn para Perceptron https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron

- **MLP** - implementação do scikit Learn para MLPs. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

- **Keras** - framework para implementação de redes neurais e deep learning com Python e Tensor Flow. É possível criar MLPs com as funções e componentes existentes. https://keras.io