Manual de usuário para o toolbox associado ao EFC 1 – IA353

O primeiro aspecto a ser considerado é a escolha de um problema de classificação/regressão que admita ser abordado a partir de dados disponíveis. Os dados disponíveis devem ser devidamente normalizados (média zero e variância unitária) ou escalados para o intervalo [−1, +1], por exemplo. Não é necessário fazer isso para os dados em 'train.mat' (gerado pelo programa ⟨**gen_data.m**⟩, conforme descrito mais abaixo), mesmo percebendo que eles excursionam num intervalo 4 vezes maior que o sugerido. Os dois fatores que mais impactam no desempenho das redes neurais MLP são: (1) entradas, podendo ser uma apenas, que excursionam em intervalos muito extensos (por exemplo [−10000, +10000]); e (2) entradas que excursionam em intervalos muito diferentes entre si (por exemplo, uma no intervalo [−1, +1] e outra no intervalo [−0.00001, +0.00001]).

Para a execução do toolbox em Matlab, todos os dados disponíveis para treinamento devem ser colocados em um único arquivo. Se a dimensão do espaço de entrada for m e a do espaço de saída for r, os N dados disponíveis para treinamento devem ser dispostos em duas matrizes: (1) a matriz X de dimensão $N \times m$ com as entradas e a matriz S de dimensão $N \times r$ com as saídas. Uma vez definidas essas duas matrizes no ambiente do Matlab, o comando a seguir permite gerar o arquivo que deve ser utilizado como entrada do próximo passo de execução do toolbox.

Comando do Matlab: **`save filename X S`**

O usuário é quem escolhe o nome do arquivo (a ser utilizado no campo ⟨filename⟩ acima), sendo que a extensão será '.mat', indicando um arquivo binário que pode ser lido pelo Matlab. Observação: Pode-se também salvar os dados em arquivo ASCII, mas esta opção não é considerada no toolbox.

Para que o usuário possa realizar ao menos uma execução do toolbox, sem precisar dispor dos seus próprios dados de treinamento e sem precisar executar nenhum dos pré-processamentos acima, é disponibilizado o programa ⟨**gen_data.m**⟩, o qual, ao ser executado, produz dados artificiais de entrada-saída de uma rede neural MLP com 2 entradas, uma saída e 5 neurônios na camada intermediária. Os pesos são fixos e definidos arbitrariamente. Esses dados artificiais podem ser gerados com ou sem ruído. Sugere-se fortemente o emprego de ruído na geração dos dados. Esses dados gerados artificialmente são gravados num arquivo denominado 'train.mat'.

O próximo passo é executar o programa ⟨**gen_k_folds.m**⟩. Será solicitado o nome do arquivo, o qual deve ser colocado na forma 'filename.mat' ou 'filename', entre aspas simples. Também será solicitado o número de pastas k e a porcentagem dos dados que deve ser reservada para uso na síntese do ensemble. Esses dados reservados serão chamados de **dados de teste** e uma sugestão é que eles correspondam a 15% do total de dados (informar 15 e não 0,15 como entrada do programa). Automaticamente, o programa vai gerar k arquivos denominados filename1.mat, filename2.mat, ..., filenamek.mat, onde cada um desses arquivos conterá um conjunto de amostras do arquivo filename.mat, sendo que a distribuição de amostras pelos arquivos é aleatória e a mais equilibrada possível. Exemplo: Dividir 253 amostras em 10 pastas fará com que as 3 primeiras pastas fiquem com 26 amostras cada, enquanto que as 7 últimas pastas ficarão com 25 amostras cada.

Dispondo das *k* pastas, é possível executar o procedimento denominado *k-fold cross-validation*, que é descrito no enunciado da Questão 7 do EFC 1. Para tanto, é necessário executar o programa ⟨**nn1h_k_folds.m**⟩. A sigla nn1h vem do inglês *neural network with one hidden layer*. Este programa vai solicitar o nome-raiz das pastas (neste caso, 'filename'), o número de pastas (neste caso, *k*), o número de neurônios na camada intermediária (neste caso, um valor entre 5 e 20) e o tipo de geração dos pesos iniciais da rede neural. Recomenda-se, neste último item, o uso da opção 1. Só utilize a opção 2 caso queira iniciar o treinamento com os mesmos pesos já utilizados num treinamento anterior, estando certo de que o número de neurônios na camada intermediária se mantém o mesmo. A opção 3 serve para reiniciar o treinamento de onde ele parou numa execução anterior, tomando o estado em que o treinamento se encontrava na última iteração da execução anterior. Para qualquer uma das 3 opções, é solicitado o número máximo de iterações (número adicional de iterações na opção 3), lembrando que o critério de parada é número máximo de iterações ou norma do vetor gradiente abaixo de um valor mínimo pré-definido.

O programa ⟨**nn1h_k_folds.m**⟩ emprega um algoritmo de otimização de 2a. ordem, denominado **gradiente conjugado escalonado estendido**, vai criar um conjunto inicial de pesos diferente para cada um dos *k* treinamentos e vai treinar *k* MLPs distintas, uma para cada conjunto de $k-1$ pastas, tomando a pasta restante como conjunto de dados de validação. Para a *j*-ésima, pasta, serão salvos, após o treinamento, os pesos {w10j.mat, w20j.mat} (pesos iniciais), {w1j.mat, w2j.mat} (pesos ao término do treinamento, ou seja, o estado das conexões da *j*-ésima rede MLP ao final da última iteração de treinamento) e {w1vj.mat, w2vj.mat} (pesos que minimizam o erro junto ao conjunto de validação). O programa ⟨**nn1h_k_folds.m**⟩ realiza em sequência o treinamento das *k* redes neurais MLP, apresentando na tela algumas informações pertinentes.

O programa ⟨**nn1h_k_folds.m**⟩ chama internamente os programas ⟨**init_k_folds.m**⟩, ⟨**process.m**⟩, ⟨**qmean.m**⟩, ⟨**qmean2.m**⟩ e ⟨**hprocess.m**⟩.

Encerrada a fase de treinamento supervisionado, inicia-se a análise de desempenho médio das *k* redes neurais MLP e a síntese do ensemble. O programa ⟨**analysis.m**⟩ faz isso para o usuário, mas restrito apenas à alternativa de média (para combinar a saída das *k* redes neurais MLP, que representa um dos três itens solicitados na Questão 7). Para tanto, ele vai utilizar os dados de teste, já gerados por ocasião da execução do programa ⟨**gen_k_folds.m**⟩ e sempre denominados 'test.mat'.

O programa ⟨**analysis.m**⟩ pode ser executado tanto com os pesos ao final do treinamento (última iteração) como com os pesos que minimizam o erro de validação.

Observação 1: Dada a aplicação pretendida, quanto maior o risco de sobretreinamento das redes neurais MLP individuais, maior tende a ser o ganho promovido pela abordagem ensemble.

Observação 2: Qualquer problema com o toolbox ou possibilidade de melhoria do mesmo, favor comunicar o professor.

Observação 3: Este toolbox é uma sugestão de uso. Nenhum aluno está obrigado a utilizá-lo para resolver questões do EFC 1.

Observação 4: Está sendo disponibilizado um toolbox adicional, dedicado à implementação do método holdout, em que os dados são divididos em conjunto de treinamento (X e S, arquivo 'train.mat'), validação (Xv e Sv, arquivo 'valid.mat') e teste (Xt e St, arquivo 'test.mat'). Não é disponibilizado um programa para promover a divisão de um único conjunto de dados nesses três arquivos. O arquivo 'test.mat' só é usado pelo programa ⟨**analysis3.m**⟩.