EA072 – Turma A (1s2017) – EFC 1
Realizar em grupo de 2 ou 3 alunos(as)

Questão 1) (2,0 pontos) (Data de entrega do relatório: 05/10/2017)
Objetivo: Síntese de modelos lineares e não-lineares para classificação de padrões.
Caso de estudo: Base de dados MNIST (http://yann.lecun.com/exdb/mnist/), a qual contém dígitos manuscritos rotulados em 10 classes (são os dígitos de '0' a '9'), sendo 60.000 amostras para treinamento e 10.000 amostras para teste (os dados de teste não devem ser empregados em nenhuma fase do processo de síntese do classificador). Cada imagem de entrada contém 784 bits, visto que a dimensão de cada imagem é 28 × 28 pixels.

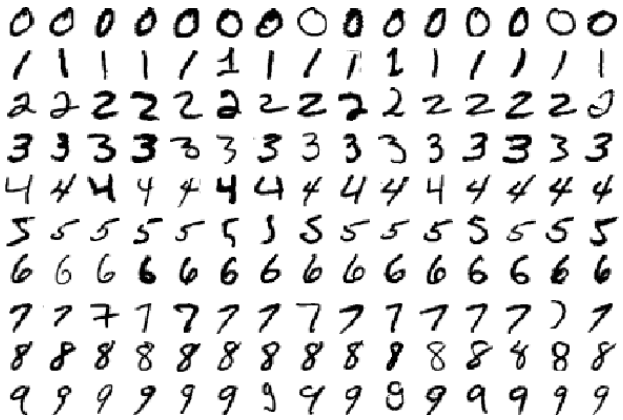

Figura 1 – Exemplos de imagens do conjunto de dados MNIST

Parte 1: Obtenha um modelo de classificação linear, sendo que os parâmetros do modelo linear devem compor uma matriz de dimensão 785 × 10 e devem ser obtidos de forma fechada, a partir de uma única expressão algébrica. Deve-se buscar um bom parâmetro de regularização, o qual tem que ser maior do que zero, pois a matriz de dados de entrada não tem posto completo. Para tanto, tomar parte dos dados de treinamento como validação para poder implementar esta busca pelo melhor parâmetro de regularização. Uma vez encontrado um valor adequado para o parâmetro de regularização, usar todas as 60.000 amostras de treinamento para sintetizar o classificador linear. Apresentar o desempenho junto aos dados de teste, em termos de taxa média de acerto (considerando todas as classes) e taxa média de acerto por classe. Esses mesmos índices de desempenho devem ser empregados nas outras duas partes da atividade.

Parte 2: Faça o mesmo procedimento da Parte 1, mas agora empregando uma máquina de aprendizado extremo (ELM) do tipo MLP, com uma ou mais camadas intermediárias. Procure definir adequadamente o número de neurônios dessa(s) camada(s) intermediária(s) e a estratégia de definição dos pesos sinápticos.

Parte 3: Enquanto as duas partes anteriores envolveram problemas de otimização linear, sendo a Parte 1 para um modelo linear e a Parte 2 para um modelo não-linear, nesta terceira parte o objetivo é treinar uma rede neural MLP com um, duas ou três camadas intermediárias. Sugere-se o uso de um dos três toolboxes em Matlab disponibilizados pelo professor, o qual emprega validação cruzada de *k* pastas,

visando minimizar o erro de classificação junto à pasta de validação. Procure definir adequadamente o número de neurônios dessa(s) camada(s) intermediária(s).

Consolidação dos resultados: Realize uma análise comparativa dos resultados obtidos nas três partes e compare com o estado-da-arte disponível na literatura.

Observação 1: Nas três partes, deve-se considerar classificadores com uma saída por classe, de tal modo que a classe indicada é aquela associada à saída de maior valor numérico. É fundamental o emprego do mesmo critério de desempenho nas três partes. A medida denominada *Classification Error Rate* (CER) contabiliza os erros cometidos em cada classe e divide pelo total de amostras de cada classe, para obter uma taxa de erro por classe. Em seguida, soma esses erros relativos de todas as classes e divide pelo total de classes. Trata-se, portanto, de uma medida restrita ao intervalo [0, 1].

Observação 2: O problema de classificação tratado aqui é de grande porte, de modo que algumas etapas da atividade vão requerer custo computacional elevado, mas ainda passível de realização em equipamentos individuais e de médio desempenho computacional, como um notebook, por exemplo. Também pode haver problemas de limitação de memória para certas configurações.

Observação 3: Por se tratar de reconhecimento de padrões em imagens, modelos lineares, redes ELM e redes MLP não representam o estado-da-arte, sendo superadas em desempenho por redes convolucionais com arquiteturas profundas (*deep learning*) e ensemble dessas redes convolucionais, incluindo também técnicas de *data augmentation*. No entanto, mesmo sem esses recursos, o desempenho será próximo do estado-da-arte, ao menos no caso dos modelos não-lineares.

Observação 4: Especificamente para a Parte 3 desta atividade, o professor está disponibilizando 3 toolboxes em Matlab (https://www.mathworks.com/), os quais também rodam no Octave (https://www.gnu.org/software/octave/). Eles são idênticos no fluxo de informação, diferindo apenas no número de camadas intermediárias da rede MLP: um toolbox considera uma camada intermediária para a rede neural MLP, o segundo duas e o terceiro adota três. A codificação poderia ter sido mais sofisticada e aceitar o número de camadas como argumento do programa, mas não se atingiu esta funcionalidade, que levaria a um único toolbox.

Observação 5: Não é preciso baixar o conjunto de dados MNIST do *link* mencionado acima, pois isso já foi feito pelo professor, deixando os dados prontos para uso.

Questão 2) (1,0 pontos) (Data de entrega do relatório: 05/10/2017)
Emprego da metodologia *wrapper* com *backward elimination* para seleção de variáveis em uma tarefa de regressão. Comparação com a abordagem baseada em filtro.

(1) ~~Recorra ao paper [Guyon, I.; Elisseeff, A. "An introduction to variable and feature selection", Journal of Machine Learning Research, vol. 3, pp. 1157-1182, 2003] para explicar a diferença entre filtro e *wrapper*.~~

(2) ~~Explique como funcionam as abordagens *forward selection* e *backward elimination* e apresente a razão pela qual elas não garantem encontrar a melhor~~

combinação de entradas para a tarefa. Aponte vantagens e desvantagens da abordagem *backward elimination*.

(3) Caso de estudo 1: Série temporal SUNSPOT, do ano de 1749 ao ano de 2014 [http://www.esrl.noaa.gov/psd/gcos_wgsp/Timeseries/Data/sunspot.long.data]. São considerados 20 atrasos candidatos (a entrada 1 corresponde ao atraso 20 e a entrada 20 corresponde ao atraso 1), são incluídas 5 entradas aleatórias e deve ser empregado 10-folds cross-validation. Todas as variáveis excursionam no intervalo [0;+0,2]. Operação com preditor linear e preditor ELM, ambos com regularização.

a. Use os programas do diretório [wrapper_sunspot]. Comente acerca da série temporal sunspot: a que seus valores se referem, qual é o período estimado desta série, qual é a periodicidade dos valores medidos, etc.
b. Use o programa [pre_proc.m], que vai carregar os dados do arquivo [sunspot.txt]. Defina o número de atrasos candidatos como sendo 20 e o número de entradas aleatórias como sendo 5. A execução de [pre_proc.m] vai preparar os dados de treinamento, salvos no arquivo [train.mat].
c. Use o programa [lin_filter.m] com argumento 'train' de modo a obter as correlações lineares de Pearson entre as 25 entradas candidatas e a saída (predição a ser realizada). Comente os resultados obtidos.
d. Use o programa [gen_k_folds.m], com arquivo de entrada 'train' e número de pastas para validação cruzada igual a 10. Do único arquivo de treinamento, serão produzidas 10 pastas (arquivos [train1.mat] até [train10.mat]).
e. Use o programa [wrapper_lin_regr.m], fornecendo as mesmas informações dos programas anteriores, de modo a realizar a seleção de variáveis empregando *wrapper* com *backward elimination*. Comente os resultados obtidos e compare com o item (3c). Não deixe de apresentar as entradas que produzem o menor erro médio RMS para os 10 conjuntos de validação.
f. Apresente o diagrama de barras com a frequência de escolha dos 7 valores definidos para a regularização do modelo linear. É necessário regularizar um modelo linear? Justifique.
g. Explique por que nem sempre as entradas de menor correlação de Pearson são as primeiras a serem podadas na abordagem *backward elimination*, sendo algumas dessas entradas de baixa correlação até selecionadas para compor a entrada do modelo, ao final.
h. Use o programa [wrapper_nlin_regr.m], fornecendo as mesmas informações dos programas anteriores, de modo a realizar a seleção de variáveis empregando *wrapper* com *backward elimination*. O modelo não-linear deve ser uma ELM com um número de neurônios em torno de 100. Comente os resultados obtidos e compare com o item (3e). Não deixe de apresentar as entradas que produzem o menor erro médio RMS para os 10 conjuntos de validação.
i. Apresente o diagrama de barras com a frequência de escolha dos 7 valores definidos para a regularização do modelo não-linear. Comente.

Questão 3) (1,0 pontos) (Data de entrega do relatório: 05/10/2017)

No contexto de redução de dimensão e visualização de dados, análise de componentes principais (PCA, do inglês *principal component analysis*) é geralmente adotada. Basicamente, o PCA define um hiperplano de projeção de dimensão menor ou igual à dimensão do espaço original dos dados. O critério de otimização do PCA é minimizar

o somatório da distância dos dados originais ao hiperplano. Como consequência da aplicação deste critério, o posicionamento ótimo do hiperplano de projeção faz com que os dados projetados apresentem o máximo espalhamento possível, de modo que os eixos do hiperplano representam as direções de maior variância dos dados. Abusando um pouco da terminologia, pode-se denominar de análise de componentes principais não-lineares (NLPCA, do inglês *nonlinear principal component analysis*) a toda iniciativa equivalente ao PCA, mas que toma uma hipersuperfície em lugar de um hiperplano de projeção. Como esta hipersuperfície pode ser produzida por uma rede neural MLP, iremos empregar aqui uma técnica de NLPCA para um problema de classificação de padrões, comparando o resultado do PCA com o resultado do NLPCA. A estratégia é ilustrada na Figura 1 a seguir.

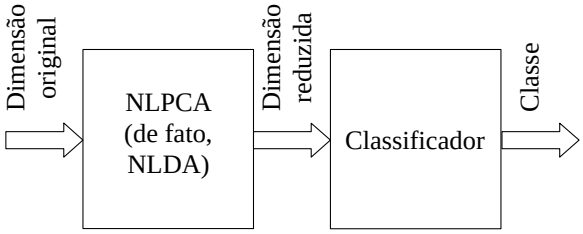

Figura Q7 – Adaptação de análise de componentes principais não-lineares, de fato, produzindo análise discriminante não-linear

A ideia é usar uma MLP para o papel de NLPCA e uma outra MLP para o classificador, e treinar ambas as MLPs simultaneamente. Em termos práticos, resulta uma única MLP com 5 camadas, mas onde terceira e quinta camadas têm neurônios com funções de ativação linear. Sendo o número de saídas da terceira camada menor que o número de entradas, cria-se um gargalo no processamento de informação, permitindo interpretar a saída da terceira camada como os componentes principais não-lineares. Na verdade, o conceito de NLPCA está efetivamente associado ao conceito de codificador de um *autoencoder* (https://en.wikipedia.org/wiki/Autoencoder). A iniciativa da figura acima, que emprega um codificador seguido de um classificador ou regressor é melhor descrita como *nonlinear discriminant analysis* (NLDA). Para implementar este esquema, empregue o *toolbox* de NLDA fornecido pelo professor para obter os 2 componentes principais para os dados de entrada do *Wine data set* (*UCI Machine Learning Repository*) [http://archive.ics.uci.edu/ml/datasets/Wine]. Os dados de entrada correspondem às colunas de 2 a 14 da base de dados original, as quais devem ser independentemente escaladas para o intervalo [–1, +1]. As classes estão indicadas na coluna 1. O programa [pre_proc_wine.m], ao ser executado, realiza esta etapa e ainda prepara a saída desejada para vetores binários de dimensão igual ao número de classes. Com base nas matrizes de pesos [w1.mat] e [w2.mat] produzidas após o treinamento (use 15 neurônios nas duas camadas intermediárias com funções de ativação não-linear), plote os dois componentes principais (eles correspondem à saída do gargalo) num plano, colorindo os 178 pontos no $\Re^2$ com vermelho (se classe 1), preto (se classe 2) e azul (se classe 3). Isso é feito pelo programa [visual_NLDA.m]. Faça o mesmo no caso de PCA. Isso é feito pelo programa [visual_PCA.m]. Um classificador usando PCA conseguiria o mesmo desempenho que um classificador usando NLDA para *Wine data set*? Nota: Evidentemente, continuam válidas as preocupações com *overfitting*, mas nesta atividade em particular não se considera *holdout* ou *cross-validation*.