IA013 (1s2017) – EFC 1
Realizar em grupos de 2 ou 3 alunos(as)
Data de entrega e pontuação junto a cada questão

Questão 1) (2,0 pontos) (Data de entrega: 28/09/2017)
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.

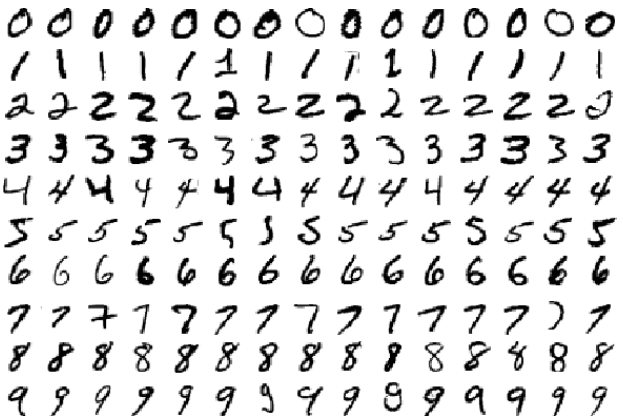

Figura 1 – Exemplos de imagens do conjunto de dados MNIST

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 pelos professores, os quais empregam 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, os professores estão 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 pelos professores, deixando os dados prontos para uso.

Questão 2) (1,0 pontos) (Data de entrega: 10/10/2017)
Implementar a solução de uma instância do problema de caixeiro viajante por Hopfield, por Kohonen e por um algoritmo genético. Nenhuma dessas propostas será competitiva com o estado da arte, por exemplo, dado pelo algoritmo Concorde (http://www.math.uwaterloo.ca/tsp/concorde/index.html), mas o objetivo é trabalhar os conceitos de dinâmica de relaxação, auto-organização e evolução artificial. Devem ser apresentados os pseudo-códigos e resultados intermediários, além do resultado final. Para Hopfield e Kohonen, são disponibilizadas versões funcionais preliminares, que devem ser aperfeiçoadas para ganho de desempenho. Por exemplo, usar o link ⟨ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/theses/tese_Lalinka_Gomes.pdf⟩, capítulo 4, para Kohonen, e o paper Smith et al. "Neural Techniques for Combinatorial Optimization with Applications", IEEE Trans. on Neural Networks, para Hopfiled ⟨ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/theses/tese_Lalinka_Gomes.pdf⟩. No

caso do algoritmo genético, devem ser propostos múltiplos operadores de recombinação e mutação, além de operadores de busca local.

Questão 3) (1,0 pontos) (Data de entrega: 10/10/2017)
Escolha um problema adequado para aplicação de um algoritmo genético, o qual admite codificação binária, proponha operadores de crossover e mutação pertinentes e resolva o problema. Evite aqui problemas que admitem outros tipos de codificação, como ponto flutuante, em que a codificação binária poderia ser utilizada. Apresentar no relatório: (1) Pseudo-código do seu algoritmo evolutivo, mostrando a sequência de operações que ele realiza; (2) Descrição dos operadores propostos; (3) Descrição de como se obteve a população inicial; (4) Descrição da função de *fitness* adotada; (5) Descrição do(s) operador(es) de seleção adotado(s); (6) Valores das taxas e parâmetros do algoritmo e como se chegou a eles; (7) Critério(s) de parada adotado(s). (8) Gráficos apresentando o comportamento do processo evolutivo ao longo das gerações (ao menos a evolução do *fitness* do melhor indivíduo e da média da população). Sugestão: Use crossover uniforme e evite espaços de busca pequenos e com regiões de infactibilidade. Não é necessário incluir operadores de busca local.

Questão 4) (1,0 pontos) (Data de entrega: 10/10/2017)
O mesmo que o anterior, mas agora para problemas que inerentemente admitem codificação em ponto flutuante. Sugestão: Use crossover aritmético. Não é necessário incluir operadores de busca local.

Questão 5) (1,0 pontos) (Data de entrega: 19/10/2017)
Baixe o código do algoritmo NSGA-II (http://www.iitk.ac.in/kangal/codes.shtml) (que é um dos dois mais empregados no tratamento evolutivo de problemas multiobjetivo), defina um problema de caixeiro viajante multiobjetivo à sua escolha, com dois objetivos, e adapte os módulos de avaliação de *fitness* do NSGA-II para o seu problema. Apresente os resultados com a fronteira de Pareto no espaço dos objetivos e com percursos do caixeiro viajante para alguns pontos da Fronteira de Pareto. Inclua comentários pertinentes referentes a outras decisões tomadas durante o processo de uso do NSGA-II. Em seguida, como uma segunda aplicação, considere um dos problemas ZDT, os quais têm fronteira de Pareto conhecida, e apresente os resultados. Observação 1: Existem outras versões do NSGA-II, inclusive para Matlab. Observação 2: Referência original que propôs os problemas ZDT: *E. Zitzler, K. Deb, and L. Thiele "Comparison of multiobjective evolutionary algorithms: Empirical results", Evolutionary Computation, vol. 8, no. 2, pp. 173–195, 2000.*

Questão 6) (1,0 pontos) (Data de entrega: 19/10/2017)
Gere você mesmo um mapeamento $\Re^1 \rightarrow \Re^1$, ou seja, $y = f(x)$, amostre dados para um intervalo restrito de entrada e empregue o software Eureqa, disponível em ⟨http://www.nutonian.com/products/eureqa/⟩, para propor alternativas de funções de aproximação para a sua $f(\cdot)$. Faça o mesmo para um mapeamento $\Re^3 \rightarrow \Re^1$, ou seja, $y = f(x_1, x_2, x_3)$, mas evite propor funções compostas (ou seja, funções que têm como argumento outras funções), empregando mais composições aditivas, multiplicativas e mistas de funções. Controle o intervalo de cada variável de entrada, produza mapeamentos desafiadores, trabalhe com um bom número de amostras e considere a situação de amostras com e sem ruído. Interprete os resultados obtidos, ao longo da fronteira de Pareto.