Universidade Estadual de Maringá – Centro de Tecnologia
Depto. de Engenharia Química – Curso de Engenharia Elétrica
6694 – Sistemas Inteligentes - 1º Sem. 2020

Avaliação 1 – Parte 2

Data final de entrega: 30/11/2020

Grupos de até 4 pessoas (mas pode fazer individual)

Regras importantes:

- Não será tolerado plágio. Trabalhos identificados como copiados serão penalizados.
- A data final de entrega é inadiável. Como o trabalho pode ser entregue antes da data final, não há motivos para tolerar a não entrega na data final.

Material a entregar

- Códigos em Python;
- Por email (para rkrummenauer2@uem.br):
  - O assunto da mensagem deve ser 1s2020:6694:<nome1>:<nome2>:<nome3>:<nome4>
    - Exemplo: 1s2020:6694:bonovox:tinaturner:fredmercury:johnlennon
  - Relatório (em formato PDF).

Exercício de implementação:

1. (Peso 7,0) **Predição.** Seja o conjunto de dados de um acelerômetro triaxial montado no peito de voluntários disponível no repositório online da Universidade da Califórnia - Irvine no endereço https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer. Para este projeto, a parte selecionada do conjunto de dados já foi separada e pré-processada (é o arquivo `dados_Ax.mat`), formando uma série temporal de aceleração no eixo X de um dos participantes, realizando a atividade de número 5 (subindo ou descendo escadas). Utilize o programa principal de predição desta série (exatamente os mesmos dados) e os códigos auxiliares já vistos em aula (códigos aqui) e implemente as seguintes características no treinamento da rede neural tipo MLP: i) Implemente o modo *minibatch* (em blocos de tamanho ajustável pelo usuário) para que o bloco/lote usado para ajuste dos pesos da rede possa ter tamanho que inclua todos os padrões a cada atualização dos pesos da rede (equivalente a fazer em batelada) até blocos de tamanho unitário (treinamento padrão-a-padrão, um padrão por vez na atualização dos pesos). O tamanho do bloco deve ser escolha do usuário; ii) Faça a melhora (no exemplo dado em aula utilizamos o método do gradiente) da direção de atualização dos pesos utilizando um método de otimização de ordem 2 (quase-Newton como o BFGS ou o Gradiente Conjugado, por exemplo) utilizando o módulo de otimização que criamos na disciplina (`optimize_6694.py`) ou, melhor ainda, utilizando a biblioteca

`scipy.optimize` (veja exemplo de aula aqui). Compare o desempenho de predição de uma rede neural do tipo MLP com uma camada intermediária usando funções de ativação do tipo tangente hiperbólica (escolha o número de neurônios que considerar suficiente), utilizando treinamento *minibatch* com lote de tamanho 20 e utilizando treinamento em batelada (*batch*). Faça a predição para os seguintes passos: 5 amostras, 10 amostras e 52 amostras (esta última equivalente a 1s de dados); e analise os resultados com argumentos fundamentados. Apresente os resultados da evolução do erro quadrático médio em função da época (iteração) no treinamento e validação das duas redes MLP (*minibatch* e *batch*). Dicas: Um parâmetro de projeto importante é a dimensão de entrada, portanto, considere um tamanho de sequência temporal razoável (número de amostras sequenciais) para formar os padrões de entrada. A amostragem é realizada a 52 Hz. Recomenda-se utilizar ao menos 52 amostras para cada padrão (janela de tempo). Para separação dos padrões de entrada da rede, considere usar o método da janela deslizante (*sliding window*) com uma amostra de avanço para cada nova janela. Separe 60% destes padrões para treinamento, 20% para validação e 20% para teste. Tanto a divisão dos padrões de treinamento, validação e teste, como a formação dos padrões com a janela deslizante, já estão no exemplo de código dado.