IA353 – Redes Neurais (1s2020)
Exercícios de Fixação de Conceitos 2 – EFC 2
Atividade Individual – Peso 3
Data de entrega dos resultados solicitados: 07/05/2020

Observação: O peso do EFC1 passa a ser de 1, pois a Q2 compõe agora o EFC2. Há algumas poucas alterações no enunciado da Q2, que foram destacadas em vermelho.

Questão 2)
Objetivo: Síntese de modelos não-lineares para classificação de padrões, mas lineares nos parâmetros ajustáveis. Para tanto, repetir o fluxograma da Q1, mas agora para uma máquina de aprendizado extremo (ELM) com 500 neurônios na camada intermediária e pesos definidos aleatoriamente, com distribuição normal e desvio padrão 0,2. Não variar esses pesos, ou seja, usar a mesma camada intermediária para todos os experimentos solicitados e também adotar a mesma partição para os conjuntos de treinamento e validação, durante a busca pelo coeficiente de regularização.

O que deve ser entregue: 1 arquivo [Q2_RA_do(a)_aluno(a)].pdf com os valores dos 2 coeficientes de regularização encontrados na busca (um para o erro quadrático médio e outro para o erro de classificação) e 2 gráficos semilog do desempenho dos classificadores junto aos dados de validação para os 11 valores de coeficiente de regularização sugeridos (um para o erro quadrático médio e outro para o erro de classificação), seguidos pelos 2 gráficos da fase de refinamento. Se o intervalo sugerido não for adequado, propor um intervalo de busca mais adequado. Análises a serem feitas: (2.1) Considerando os dados de treinamento, apresente a matriz de confusão e alguns exemplos de dígitos classificados de forma equivocada, ao menos de 3 classes distintas. (2.2) Seguindo a sugestão de empregar 500 unidades na camada intermediária da rede neural, apresente argumentos para sustentar o ganho de desempenho verificado e uma execução em menor tempo computacional, quando comparado com o classificador linear da Q1. (2.3) Compare os coeficientes de regularização obtidos nessas duas primeiras atividades (classificador linear da Q1 e ELM da Q2), os apresentando numa única tabela para fácil visualização, e procure justificar a diferença. (2.4) Mantendo a mesma partição entre conjuntos de treinamento e validação, o que você espera que ocorra com o coeficiente de regularização caso os neurônios da camada intermediária sejam inicializados com pesos sinápticos distintos a cada execução? (2.5) Promova algum tipo de alteração nas especificações da ELM e/ou de seu treinamento de modo a produzir resultados superiores àqueles conquistados ao se seguir o roteiro desta questão. Descreva adequadamente as alterações realizadas.

Especificações para Q3 e Q4: Trabalhe no ambiente Anaconda e elabore o seu relatório como um Jupyter notebook. Todas as atividades deverão, assim, compor um único relatório no Jupyter notebook, a ser encaminhado ao Prof. Fernando J. Von Zuben (vonzuben@dca.fee.unicamp.br) até 23h59 do dia especificado para entrega, usando [IA353 - EFC2] como assunto do e-mail.

Questão 3)
Tomando o mesmo problema de classificação de dados da base MNIST, use o framework Keras, tendo o TensorFlow como *backend* e realize o treinamento de uma rede neural MLP. Busque inspiração em resultados já publicados na literatura e/ou adote o procedimento de tentativa-e-erro para definir, da melhor forma que você puder, o número de camadas intermediárias, o número de neurônios por camada, o algoritmo de ajuste de pesos, a taxa de *dropout* (onde for pertinente) e o número de épocas de treinamento. Procure trabalhar com a média de várias execuções (junto a cada configuração candidata) para se chegar a um índice de desempenho mais estável. Um código que pode servir de ponto de partida é fornecido a seguir, considerando 1 camada intermediária, 512 neurônios nesta camada intermediária, ADAM, ocorrência de dropout com taxa de 50%, 5 épocas de treinamento e função perda sendo uma forma de entropia cruzada. A sua proposta deve ser capaz de superar o desempenho desta sugestão abaixo e você deve descrever de forma objetiva o caminho trilhado até a sua configuração final de código para a rede neural MLP, assim como uma comparação de desempenho com a sugestão abaixo.

```
import tensorflow as tf
import os
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.5),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

model_json = model.to_json()
json_file = open("model_MLP.json", "w")
json_file.write(model_json)
json_file.close()
model.save_weights("model_MLP.h5")
print("Model saved to disk")
os.getcwd()
```

Questão 4)
Tomando o mesmo problema de classificação de dados da base MNIST e novamente usando o framework Keras, tendo o TensorFlow como *backend*, realize o treinamento de uma rede neural com camadas convolucionais, usando maxpooling e dropout. Mais uma vez, é apresentada a seguir uma sugestão de código e de configuração de hiperparâmetros que pode ser tomada como ponto de partida. A sua proposta deve superar, em termos de desempenho médio, essa sugestão fornecida abaixo. Compare os resultados (em termos de taxa de acerto na classificação) com aqueles obtidos pelos três tipos de máquinas de aprendizado adotadas nas atividades anteriores (classificador linear, ELM e MLP). Descreva de forma objetiva o caminho trilhado até a sua configuração final de código.

```
import tensorflow as tf
import os
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()

# reshape to be [samples][width][height][pixels]
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                        activation='relu',
                        input_shape=(28, 28, 1)))
model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.Dropout(0.25))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(10, activation='softmax'))

model.compile(optimizer='adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

model_json = model.to_json()
json_file = open("model_CNN.json", "w")
json_file.write(model_json)
json_file.close()
model.save_weights("model_CNN.h5")
print("Model saved to disk")
os.getcwd()
```