Trabalho 2 Redes Neurais

1. Requisitos de Software:

Anaconda 5.2.0

Instalar anaconda
Abrir anaconda window start anaconda 64

conda create -n py37 python=3.7 anaconda
activate py37
conda install spyder
activate py37
conda install tensorflow
conda install keras

pip install --ignore-installed --upgrade
conda install pandas
conda install scikit-learn
conda install matplotlib

Atualização do SPYDER: https://github.com/spyder-ide/spyder/releases
conda update qt pyqt
conda install -c spyder-ide spyder=4.0.0b1

OBS: Os requisitos abaixo podem precisar ser atualizado na versão mais recente.

2. Base de Dados

A base de base_pacientes_sintomas.xlsx contem 11 sinais e sintomas para covid-19 seguida pela rótulo Resultado: 0- negativo para covid, 1 positivo para covid.

Há duas abas, no arquivo citado acima, já previamente separadas:

- "treino" contendo os dados para busca pela melhor arquitetura e parametrização e
- "teste" para avaliação dos resultados a após a identificação do melhor modelo.

3. OBJETIVOS

O objetivo é maximizar a classificação da base de COVID-19, identificando a melhor parametrização e arquitetura.

No script pode-se alterar:

- Número de neurônios na camada escondida e os parâmetros de treinamento.
- n_batch: número de registros que devem ser apresentados a cada atualização dos pesos

batch_size*int, default='auto'*

Size of minibatches for stochastic optimizers. If the solver is ‘lbfgs’, the classifier will not use minibatch. When set to “auto”, batch_size=min(200, n_samples)

- repeats: número de instâncias da rede neural que deverá ser avaliada para configuração de parâmetros definidos

- activation: funções de ativação, há no script outras opções
  activation*{‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, default=’relu’*
  Activation function for the hidden layer.

  - ‘identity’, no-op activation, useful to implement linear bottleneck, returns f(x) = x
  - ‘logistic’, the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
  - ‘tanh’, the hyperbolic tan function, returns f(x) = tanh(x).
  - ‘relu’, the rectified linear unit function, returns f(x) = max(0, x)

- optimizer: modelo de treinamento SGD é o mais tradicional, os outros possuem parâmetros que devem ser informados e avaliados, como o modelo Adam com sugestões no script. A função ‘adam’ funciona muito bem em conjuntos de dados relativamente grandes (com milhares de amostras de treinamento ou mais) em termos de tempo de treinamento e pontuação de validação. Para pequenos conjuntos de dados, no entanto, 'lbfgs' pode convergir mais rápido e ter um desempenho melhor.

Descomentar no script a função “busca_ajuste_modelos(data_sets, name=name)”para avaliar a melhor arquitetura e parâmetros para escolha do melhor modelo é a indicada abaixo.

Para avaliar as arquiteturas e parâmetros, comente a função busca_ajuste_melhor_resultado(data_sets, name=name)”. Esta função deverá ser usada após a identificação da melhor arquitetura e seus parâmetros, para treinar a rede com a parametrização escolhida e em seguida ter a base de teste avaliada. Propanha algumas parametrizações diferentes, aumentando o número de opções do vetor params (conforme abaixo). O modelo de avaliação está baseado em validação cruzada então as medias indicadas consideram a média obtida com os conjuntos “teste” da validação cruzada, e em seguida a média dos testes da validação para as várias instâncias desse modelo.

Exemplo:

```
params = [

    {'solver': 'sgd', 'learning_rate': 'constant', 'momentum': 0.9,      'learning_rate_init': 0.1},

   {'solver': 'adam', 'learning_rate_init': 0.1, 'epsilon': 0.00001, 'beta_1':0.9, 'beta_2': 0.99}

   ]
```

Indique no vetor “labels” as arquiteturas criadas em “params”, de forma que reflita as opções facilitando a identificação das arquiteturas e parâmetros avaliados.

Exemplo:

```
labels = [
```

"1 sgd constant learning-rate with momentum=0.9 lear_init =0.2"

"2 adam rate_init': 0.05 epsilon': 0.00001",

1. Apresente uma tabela com os erros acurácia/recall/precision/f1 em relação à validação para cada configuração de parâmetros/arquiteturas.

2. Avalie e escolha a melhor arquitetura e suas parametrizações

Descomente a função “busca_ajuste_melhor_resultado(data_sets, name=name)” para treinar a arquitetura/parametrização escolhida e avaliar os resultados de teste após a escolha do melhor modelo é a indicada abaixo: Aqui o modelo escolhido será instanciado um número determinado de vezes e o melhor modelo definido a partir do erro de validação será escolhido para testar a base de teste.

Comente a função ““busca_ajuste_modelos(data_sets, name=name)”. Esta função deverá ser usada antes para a identificação da melhor arquitetura e seus parâmetros.

3. Avalie os resultados obtidos com o conjunto de teste com valores reais para o melhor modelo

4. ENVIAR TRABALHO EM PDF
   O nome do arquivo deve indicar o primeiro e último nome dos membros do grupo.
   Na primeira página do relatório deve indicar o nome completo e matrícula dos membros do grupo.