IA353 – Redes Neurais (1s2020)
Exercícios de Fixação de Conceitos 3 – EFC 3
Atividade Individual – Peso 3
Data de entrega dos resultados solicitados: 02/07/2020

Questão 5)
Tomando a base de dados MNIST, treinar um *autoencoder* e constatar o sucesso da tarefa de codificação e de decodificação (mesmo que ainda com algumas imperfeições), indicando que duas variáveis latentes podem ser suficientes para codificar a essência de toda a variedade exibida pela escrita manuscrita de 10 dígitos. Por fim, explorar o *manifold* gerado no gargalo do *autoencoder*. Para tanto, como um primeiro passo, execute o notebook fornecido pelo professor (**q5.ipynb**) e procure compreender o que está sendo feito em cada trecho de código. Atividades práticas:

1. Com base no desempenho obtido com a proposta fornecida, procure melhorar a distribuição das classes ao longo do *manifold*, atuando na arquitetura do *autoencoder* e nos parâmetros de treinamento. Forneça o novo notebook gerado, com a denominação **([seu_RA]q5_1.ipynb**).
2. Apresente um notebook que aborde algum outro problema a ser resolvido com *autoencoder* (pode ser a versão variacional), executando e comentando as principais etapas envolvidas na solução. Devem ser apresentados resultados gráficos que comprovem o bom desempenho. Forneça o novo notebook gerado, com a denominação **([seu_RA]q5_2.ipynb**).

Questão 6)
Ainda em elaboração.

Questão 7)
Esta questão aborda conceitos de interpretabilidade em aprendizado de máquina, evidenciando dois paradigmas de grande relevância e focando num subconjunto de técnicas, dentre várias possibilidades que vêm sendo propostas na literatura. A busca por modelos de aprendizado interpretáveis sempre esteve na alça de mira das pesquisas em aprendizado de máquina, mas até recentemente havia uma predominância de foco em acurácia e se defendia a existência de um compromisso entre acurácia e interpretabilidade, de modo que a busca por mais acurácia tenderia a gerar modelos menos interpretáveis, da mesma forma que modelos mais interpretáveis não conseguiriam atingir os mesmos níveis de acurácia que modelos "caixa-preta". As últimas conquistas voltadas para interpretabilidade em aprendizado de máquina, no entanto, demonstram que acurácia e interpretabilidade podem caminhar juntas, trazendo maior confiabilidade para aplicações de inteligência artificial.
Tomando o mesmo problema de classificação de dados da base MNIST, o objetivo desta questão é analisar a interpretabilidade de uma RNA treinada. Para tanto, vamos utilizar o código a seguir para treinar uma RNA.

```
import keras

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

model.get_config()

model.compile(optimizer='adam',
 loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
evaluation = model.evaluate(x_test, y_test)

model.save('mnist_model.h5')
```

7a)

Utilize a biblioteca *innvestigate* [1] para analisar as predições da RNA e comparar diferentes métodos de explicação [3]. Uma explicação é a coleção de características do domínio interpretável, que contribuíram para que uma dada amostra produzisse uma decisão (de classificação ou regressão, por exemplo) [3]. Utilize os seguintes métodos da biblioteca *innvestigate*: *Gradient*, *SmoothGrad*, *DeepTaylor*, *LRPAlphaBeta*, *LRPEpsilon*, *LRPZ*. Um código que pode servir de ponto de partida é fornecido a seguir. Escolha 6 imagens aleatórias de 3 classes distintas para analisar, portanto teremos 2 imagens por classe. Escolha parâmetros adequados para os métodos *LRPAlphaBeta* e *LRPEpsilon* (use os mesmos parâmetros para as 6 imagens escolhidas). A Figura 1 apresenta um modelo de como os resultados devem ser exibidos. Analise o resultado obtido.

```
import keras
import innvestigate
import matplotlib.pyplot as plot

mnist = keras.datasets.mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
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 = keras.models.load_model('mnist_model.h5')
model_wo_sm = innvestigate.utils.keras.graph.model_wo_softmax(model)

imagem = x_test[0:1]
plot.imshow(imagem.squeeze(), cmap='gray', interpolation='nearest')

analyzer = innvestigate.analyzer.LRPEpsilon(model=model_wo_sm,
 epsilon=1)
analysis = analyzer.analyze(imagem)
plot.imshow(analysis.squeeze(), cmap='seismic',
 interpolation='nearest')
```

7b)

Utilize a biblioteca keras-vis [2] para interpretar as predições da RNA para cada uma das 10 classes. Neste caso, procuramos por um padrão de entrada que produza uma resposta máxima do modelo para uma métrica de interesse [3]. Um código que pode servir de ponto de partida é fornecido a seguir. Escolha valores adequados para os parâmetros que determinam os pesos da *Total variation* e *L-p norm*. Veja dicas em:

https://github.com/raghakot/keras-vis/blob/master/examples/mnist/activation_maximization.ipynb

Plote os gráficos das imagens obtidas para cada uma das 10 classes e analise o resultado.

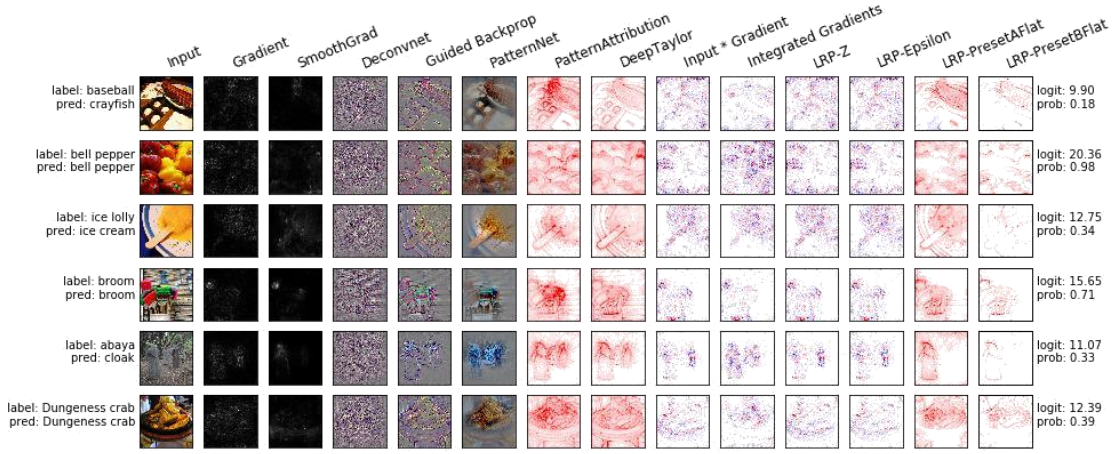

Figura 1 – Exemplo de como exibir os resultados da Questão 7a. Fonte: https://github.com/ albermax/innvestigate

```
import keras
from vis.visualization import visualize_activation
from vis.utils import utils
import matplotlib.pyplot as plot

model = keras.models.load_model('mnist_model.h5')
layer_idx = utils.find_layer_idx(model, 'dense_2')
model.layers[layer_idx].activation = keras.activations.linear
model = utils.apply_modifications(model)

filter_idx = 9
img = visualize_activation(model, layer_idx,
 filter_indices=filter_idx, input_range=(0., 1.), verbose=True,
 max_iter=1000, tv_weight=1., lp_norm_weight=0.)
plot.imshow(img.squeeze(), cmap='seismic', interpolation='nearest')
```

Referências bibliográficas

[1] M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, P.-J. Kindermans, **innvestigate neural networks!**, preprint arXiv:1808.04260, 2018.

[2] R. Kotikalapudi and contributors, **keras-vis**. https://github.com/raghakot/keras-vis, 2017.

[3] G. Montavon, W. Samek, K.-R. Müller, **Methods for interpreting and understanding deep neural networks**, Digital Signal Processing, vol. 73, pp. 1-15, 2018.