MBA em Ciência de Dados

Redes Neurais e Arquiteturas Profundas

Módulo I - Deep Learning e redes do tipo Perceptron

Exercícios com soluções

Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

Recomenda-se fortemente que os exercícios sejam feitos sem consultar as respostas antecipadamente.

Exercício 1)

As redes neurais artificiais são uma família de métodos conhecidos por conexionistas. Assinale a alternativa mais completa que explica o porquê.

(a) Porque são inspirados por neurônios biológicos, que representam a informação por uma rede de unidades de processamento e o aprendizado pela força de suas conexões
(b) Porque apenas as redes neurais possuem uma estrutura em que há nós conectados pelos quais passam as informações
(c) Porque é possível conectar os neurônios artificiais de qualquer maneira, simulando um cérebro humano
(d) Porque são métodos de aprendizado de máquina supervisionados não-paramétricos, muito utilizados em tarefas de classificação e regressão, e que armazenam informações em nós organizados de forma hierárquica, desde sua entrada, conhecida por raiz, até a saída, chamada de folha.

Exercício 2)

As redes neurais artificial passaram por um chamado "inverno" em que recebiam pouca atenção da academia e do mercado quando comparado aos sistemas especialistas. Porque isso mudou em meados de 2010?

(a) Devido a descobertas da inteligência artificial em 2012 que revolucionou a área
(b) Trabalhos científicos mostraram que a utilização de grandes bases de dados anotadas e o uso de placas gráficas permitiam treinar modelos com baixo erro de classificação
(c) Devido a melhoria dos sistemas computacionais, incluindo melhores processadores (CPUs), maior memória principal disponível (RAM) e placas gráficas (GPUs)
(d) Pois foi nessa época em que a área de ciência de dados tomou destaque, devido a grande quantidade de dados disponível (big data)

Exercício 3)

Seja $\mathbf{x} \in X$ um vetor com informações sobre perfis de clientes de um e-commerce, incluindo: data de

nascimento, cidade, quantidade de pedidos realizados nos últimos 12 meses, total em reais das compras, entre outros. Desejamos obter um modelo que seja capaz de receber por entrada uma instância de X e inferir a probabilidade y de um cliente realizar uma compra no próximo mês. Considere duas formulações conforme abaixo:

1. Modelo A: $f(\mathbf{x}) = \hat{y}$
2. Modelo B:

$$g(\mathbf{x}) = \mathbf{r}, \quad (1)$$
$$h(\mathbf{r}) = \mathbf{s}, \quad (2)$$
$$i(\mathbf{s}) = \mathbf{w}, \quad (3)$$
$$f(\mathbf{w}) = \hat{y} \quad (4)$$

O que podemos dizer sobre A e B?

(a) A é um classificador linear e B é um regressor multi-estágio
(b) apenas B é uma formulação de aprendizado de máquina, enquanto A não pode ser considerado aprendizado de máquina
(c) A é um classificador linear como o Perceptron, enquanto B é uma rede neural profunda
(d) A é uma formulação rasa, enquanto que B é uma formulação profunda para aprendizado de máquina

---

Exercício 4)

Seja $\mathbf{x}$ um vetor de entrada e s um valor de saída de um neurônio de uma rede neural baseada em Perceptron. O processamento pode ser escrito na forma:

$$f(\mathbf{x}) = a(\mathbf{w}^t\mathbf{x} + b) = s,$$

sendo que $a()$ é a função de ativação. Sabendo que a entrada tem d dimensões, quais são as dimensões das variáveis w e b?

(a) w é um vetor com d dimensões, enquanto b é escalar possuindo uma dimensão
(b) w é um vetor com 2 dimensões, enquanto b é escalar possuindo uma dimensão
(c) w é escalar com 1 dimensão, enquanto b é o vetor bias cujo número de dimensões é igual ao número de classes do problema
(d) w é um vetor com $d - 1$ dimensões, enquanto b é bias e portanto possui apenas 1 dimensão

---

Exercício 5)

Seja $\mathbf{x}$ um vetor de entrada e $\mathbf{s}$ um vetor de saída de uma camada de rede neural baseada em Perceptron. O processamento pode ser escrito na forma:

$$f(\mathbf{x}) = a(W\mathbf{x} + \mathbf{b}) = \mathbf{s},$$

sendo que $a()$ é a função de ativação. Sabendo que a entrada tem d dimensões e a saída tem z dimensões, quais são as dimensões das variáveis W e b e quantos neurônios possui essa camada?

(a) são d neurônios, sendo que W possui $z \times d$, e b possui d dimensões
(b) são $d \times z$ neurônios, sendo que W possui $z \times d$, e b possui d dimensões
(c) são z neurônios, sendo que W possui $z \times d$, e b possui z dimensões
(d) são z neurônios, sendo que W possui d, e b possui z dimensões

---

Exercício 6)

Qual a diferença entre as funções de ativação *sigmoide* e *softmax* aplicadas a uma camada de uma rede neural?

(a) a sigmóide produz como saída vetores cuja somatória é 1 e que podem ser interpretados como uma distribuição de probabilidade, enquanto a softmax comprime os valores da saída entre 0 e 1
(b) a sigmóide produz como saída vetores cujos valores estão entre -1 e 1, enquanto a softmax comprime os valores da saída entre 0 e 1
(c) a sigmóide comprime os valores da saída entre 0 e 1, enquanto a softmax produz como saída vetores cuja somatória é 1 e que podem ser interpretados como uma distribuição de probabilidade
(d) a sigmóide produz como saída vetores cujos valores individuais estão entre 0 e 1 utilizando uma função suave, enquanto a softmax produz vetores no formato one-hot-encoding compatíveis

---

Exercício 7)

Utilizando a biblioteca Keras, formule um modelo de rede neural sequencial, do tipo MLP, cuja entrada seja preparada para receber um vetor de 100 dimensões e que possua 2 camadas ocultas, cada uma com 50 neurônios, e 1 camada de saída, com 3 neurônios e função de ativação softmax.

Quantos parâmetros, no total, essa rede possui?

(a) 203
(b) 10303
(c) 7651
(d) 7753

In [1]:
```
from tensorflow import keras

model1 = keras.Sequential(
    [
        keras.layers.Dense(50, activation="relu", input_shape=(100,)),
        keras.layers.Dense(50, activation="relu"),
        keras.layers.Dense(3, activation="relu"),
    ]
)
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                5050
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 153
=================================================================
Total params: 7,753
Trainable params: 7,753
Non-trainable params: 0
_________________________________________________________________
```

Exercício 8)

Considerando que temos uma base de dados de treinamento com 8000 exemplos. Definindo o

tamanho do lote (minibatch size) como sendo 50, quantas iterações o algoritmo backpropagation deverá executar, adaptando os parâmetros da rede, para completar 5 épocas?

(a) 800
(a) 160
(b) 801
(c) 250

In [2]:
```
N = 8000
batchsize = 50

iteracoes_epoca = N/batchsize
iteracoes_epoca * 5
```

Out[2]: 800.0

Exercício 9)

Defina as sementes aleatórias do numpy para 1 e do tensorflow para 2, depois carregue a base de dados boston housing da biblioteca Keras, conforme código abaixo.

O objetivo dessa base de dados é obter a regressão do preço das casas com base em 13 características de entrada. Assim, os valores alvo (target) são escalares, tipicamente entre 10 e 50 (representando os preços em milhares de dólares).

Considerando o valor alvo descrito, e que você tem disponíveis as funções de ativação: sigmóide, relu, tangente hiperbólica e softmax, escolha as funções mais adequadas e monte uma arquitetura da rede com 3 camadas ocultas, todas com 32 neurônios.

Como deve ficar a arquitetura da rede neural?

(a) 4 camadas com 32, 32, 32 e 50 neurônios, 4202 parâmetros e função de ativação relu na camada de saída
(b) 4 camadas com 32, 32, 32 e 1 neurônios, 2593 parâmetros e função de ativação relu na camada de saída
(b) 5 camadas com 13, 32, 32, 32 e 1 neurônios, 2414 parâmetros e função de ativação sigmóide na camada de saída
(c) 4 camadas com 32, 32, 32 e 1 neurônios, 2593 parâmetros e função de ativação sigmóide na camada de saída

Justificativa: como a saída precisa ser um número positivo (preços), a função de ativação relu deve ser utilizada. O uso de funções como sigmóide comprime a saída para o valor máximo 1 o que impede obter valores no intervalo exigido pela base de dados (tipicamente entre 10 e 50). Com 3 camadas ocultas cada uma de 32 neurônios e uma camada de saída com 1 neurônio para produzir o valor (preço), veja abaixo o código que confirma o número de parâmetros.

In [3]:
```
from numpy.random import seed
seed(1)
from tensorflow.random import set_seed
set_seed(2)

from tensorflow.keras.datasets import boston_housing
(x_train, y_train), (x_target, y_target) = boston_housing.load_data()
```

In [4]:
```
model2 = keras.Sequential(
    [
        keras.layers.Dense(32, activation="relu", input_shape=(x_train.shape[1],)),
```

```
        keras.layers.Dense(32, activation="relu"),
        keras.layers.Dense(1, activation="relu"),
    ]
)
model2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                448
_________________________________________________________________
dense_4 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,537
Trainable params: 1,537
Non-trainable params: 0
_________________________________________________________________
```

Exercício 10)

Utilizando a base de dados e o modelo de rede neural criado na questão anterior, compile o modelo utilizando uma função de custo de média quadrática (mse), o otimizador SGD e a taxa de aprendizado 0.01. Adicione a métrica `mae` (mean absolute error)

Depois, normalize os dados (x) por meio da normalização z-score (calcule média e desvio no conjunto de treinamento apenas).

Utilize os dados normalizados para treinar a rede neural por 25 épocas, com batch-size 8.

Avalie o modelo treinado nos dados de teste, e reporte as posições 0 e 1 do score resultante, respectivamente relativas ao MSE e MAE calculados. Escolha a opção para a qual o intervalo se enquadre nos valores computados.

(a) MSE = (60,90), MAE = (4,8)
(b) MSE = (90,110), MAE = (10,20)
(c) MSE = (3,9), MAE = (50,100)
(d) MSE = (10,30), MAE = (6,8)

In [5]:
```
mean = x_train.mean(axis=0)
x_train -= mean
std = x_train.std(axis=0)
x_train /= std

x_target -= mean
x_target /= std

model2.compile(optimizer=keras.optimizers.SGD(0.01),
               loss='mse', metrics=['mae'])

history = model2.fit(
    x_train, y_train,
    batch_size=8,
    epochs=20,
    verbose=1,
    validation_data=(x_target, y_target),
)
```

```
Epoch 1/20
51/51 [==============================] - 0s 3ms/step - loss: 334.0641 - mae: 15.249
```

```
4 - val_loss: 247.3618 - val_mae: 13.2049
Epoch 2/20
51/51 [==============================] - 0s 965us/step - loss: 160.0893 - mae: 8.78
78 - val_loss: 142.0547 - val_mae: 9.3258
Epoch 3/20
51/51 [==============================] - 0s 1ms/step - loss: 200.4650 - mae: 7.5299
- val_loss: 77.0997 - val_mae: 7.0073
Epoch 4/20
51/51 [==============================] - 0s 1ms/step - loss: 93.5560 - mae: 6.5951
- val_loss: 92.3308 - val_mae: 6.8377
Epoch 5/20
51/51 [==============================] - 0s 955us/step - loss: 84.5059 - mae: 6.511
1 - val_loss: 90.4973 - val_mae: 6.7482
Epoch 6/20
51/51 [==============================] - 0s 1ms/step - loss: 80.5228 - mae: 6.2591
- val_loss: 80.5288 - val_mae: 6.2996
Epoch 7/20
51/51 [==============================] - 0s 1ms/step - loss: 83.7971 - mae: 6.5576
- val_loss: 88.2580 - val_mae: 6.7510
Epoch 8/20
51/51 [==============================] - 0s 975us/step - loss: 84.0770 - mae: 6.591
3 - val_loss: 88.0135 - val_mae: 6.7729
Epoch 9/20
51/51 [==============================] - 0s 997us/step - loss: 84.8860 - mae: 6.714
5 - val_loss: 88.2327 - val_mae: 6.7527
Epoch 10/20
51/51 [==============================] - 0s 1ms/step - loss: 84.8026 - mae: 6.6657
- val_loss: 88.2897 - val_mae: 6.7488
Epoch 11/20
51/51 [==============================] - 0s 1ms/step - loss: 75.3431 - mae: 6.0220
- val_loss: 66.8622 - val_mae: 5.7074
Epoch 12/20
51/51 [==============================] - 0s 1ms/step - loss: 81.7483 - mae: 6.5896
- val_loss: 87.8642 - val_mae: 6.8119
Epoch 13/20
51/51 [==============================] - 0s 1ms/step - loss: 84.8955 - mae: 6.7533
- val_loss: 87.4497 - val_mae: 6.6733
Epoch 14/20
51/51 [==============================] - 0s 972us/step - loss: 72.7666 - mae: 5.801
1 - val_loss: 55.9328 - val_mae: 4.6123
Epoch 15/20
51/51 [==============================] - 0s 877us/step - loss: 61.7811 - mae: 5.258
2 - val_loss: 71.7082 - val_mae: 6.1856
Epoch 16/20
51/51 [==============================] - 0s 1ms/step - loss: 52.7741 - mae: 4.8651
- val_loss: 55.1857 - val_mae: 5.0545
Epoch 17/20
51/51 [==============================] - 0s 961us/step - loss: 48.3681 - mae: 4.641
5 - val_loss: 55.2821 - val_mae: 4.8337
Epoch 18/20
51/51 [==============================] - 0s 1ms/step - loss: 47.6488 - mae: 4.5799
- val_loss: 54.3281 - val_mae: 5.1766
Epoch 19/20
51/51 [==============================] - 0s 970us/step - loss: 90.8173 - mae: 7.172
0 - val_loss: 87.5158 - val_mae: 6.8292
Epoch 20/20
51/51 [==============================] - 0s 1ms/step - loss: 84.3498 - mae: 6.6930
```

In [6]:
```
score = model2.evaluate(x_target, y_target, verbose=0)
print("MSE: %.1f" % (score[0]))
print("MAE: %.1f" % (score[1]))
```

```
MSE: 79.0
MAE: 6.2
```