MBA em Ciência de Dados

Redes Neurais e Arquiteturas Profundas

Módulo I - Deep Learning e redes do tipo Perceptron

Avaliação (com soluções)

Moacir Antonelli Ponti

CeMEAI - ICMC/USP São Carlos

Questão 1)

O que diferencia métodos de aprendizado profundo (*deep learning*) de métodos de aprendizado de máquina considerados rasos (*shallow*)?

(a) Os métodos rasos comumente aprendem um mapeamento direto entre dados de entrada (atributos) e saída (alvo), enquanto os profundos aprendem uma sequência de mapeamentos (ou funções) para múltiplos espaços antes de mapear para o espaço de saída alvo
(b) Os métodos rasos podem ser considerados aprendizado de máquina e permitem tarefas distintas como classificação, regressão, agrupamento, entre outros, enquanto os chamados profundos permitem modelar tarefas de classificação
(c) Os métodos rasos são baseados em métodos estatísticos e árvores de decisão, enquanto os de aprendizado profundo são unicamente baseados em redes neurais
(d) Os métodos rasos trabalham apenas com dados estruturados, enquanto que os profundos funcionam com dados estruturados e não estruturados.

Questão 2)

Seja $\mathbf{z}$ um vetor de entrada e $\mathbf{s}$ um vetor de saída de uma camada de rede neural baseada em Perceptron. Essa camada pode ser formulada como:

$f(\mathbf{z}) = a(W\mathbf{z} + \mathbf{b}) = \mathbf{s}$, sendo que $a()$ é a função de ativação.

Sabendo que a entrada tem 40 dimensões e a saída tem k dimensões, Qual o tamanho da matriz W e do vetor b e quantos parâmetros essa camada possui para serem aprendidos durante o treinamento?

(a) W possui $40 \times k$, e b possui 40 dimensões, totalizando $40k + 40$ parâmetros
(b) W possui $k \times 40$, e b possui k dimensões, totalizando $41k$ parâmetros
(c) W possui $k \times 40$, e b possui 1 dimensão (escalar), totalizando $40k + 1$ parâmetros
(d) W possui $k \times k$, e b possui 40 dimensões, totalizando $2k + 40$ parâmetros

---

Questão 3)

Qual o impacto do tamanho do batch (lote) no treinamento por meio do Stochastic Gradient Descent (SGD)?

(a) O tamanho do batch impacta na quantidade de épocas necessárias para completar o treinamento, se o tamanho do batch for grande, apenas uma época é necessária
(b) Quanto menor o tamanho do batch, mais rápido o treinamento, pois assim o SGD se aproxima do Gradient Descent convencional já que utiliza cada exemplo individualmente para adaptar os pesos
(c) Quanto menor o tamanho do batch melhor será a acurácia do modelo pois as estimativas do gradiente serão mais precisas considerando cada iteração
(d) Quanto menor o tamanho do batch, mais rápido cada iteração do treinamento, porém mais grosseira é a estimativa do gradiente por iteração

---

Questão 4)

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).

Utilizando a biblioteca Keras, formule um modelo de rede neural sequencial, do tipo MLP, com 3 camadas ocultas contendo, respectivamente, 32, 16 e 8 neurônios, todas com função de ativação do tipo `relu` .

Quantos parâmetros, no total, essa rede possui?

(a) 4096
(b) 1121
(c) 53248
(d) 3031

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

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()

model = keras.Sequential(
    [
        keras.layers.Dense(32, activation="relu", input_shape=(x_train.shape[1],)),
        keras.layers.Dense(16, activation="relu"),
        keras.layers.Dense(8, activation="relu"),
        keras.layers.Dense(1, activation="relu"),
    ]
)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                448
_________________________________________________________________
dense_1 (Dense)              (None, 16)                528
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 136
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9         
=================================================================
Total params: 1,121
Trainable params: 1,121
Non-trainable params: 0
_________________________________________________________________
```

Questão 5)

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

Adicione também a métrica `mse` (mean squared error) para permitir avaliá-la adicionalmente.

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 15 épocas, com batch-size 4.

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

(a) MAE = (50,53), MSE = (18,22)
(b) MAE = (6,12), MSE = (35,50)
(c) MAE = (4,8), MSE = (60,80)
(d) MAE = (1,5), MSE = (15,25)

In [2]:
```
mean = x_train.mean(axis=0)
x_train -= mean
std = x_train.std(axis=0)
x_train /= std

x_target -= mean
x_target /= std

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

history = model.fit(
    x_train, y_train,
    batch_size=4,
    epochs=15,
    verbose=1,
    validation_data=(x_target, y_target),
)
```

```
Epoch 1/15
101/101 [==============================] - 0s 2ms/step - loss: 15.0842 - mse: 344.3
475 - val_loss: 4.7393 - val_mse: 36.6931
Epoch 2/15
101/101 [==============================] - 0s 1ms/step - loss: 3.9585 - mse: 32.305
1 - val_loss: 5.3942 - val_mse: 46.1902
Epoch 3/15
101/101 [==============================] - 0s 1ms/step - loss: 3.2637 - mse: 22.248
4 - val_loss: 3.4779 - val_mse: 22.1614
Epoch 4/15
101/101 [==============================] - 0s 1ms/step - loss: 2.9845 - mse: 19.682
5 - val_loss: 3.0816 - val_mse: 20.8855
Epoch 5/15
101/101 [==============================] - 0s 1ms/step - loss: 2.8883 - mse: 18.373
```

```
7 - val_loss: 3.4235 - val_mse: 27.4430
Epoch 6/15
101/101 [==============================] - 0s 1ms/step - loss: 2.8413 - mse: 18.507
4 - val_loss: 4.7524 - val_mse: 38.7638
Epoch 7/15
101/101 [==============================] - 0s 1ms/step - loss: 2.5271 - mse: 14.418
0 - val_loss: 2.7270 - val_mse: 17.5546
Epoch 8/15
101/101 [==============================] - 0s 2ms/step - loss: 2.7714 - mse: 15.987
7 - val_loss: 2.7736 - val_mse: 17.0956
Epoch 9/15
101/101 [==============================] - 0s 1ms/step - loss: 2.6913 - mse: 15.530
4 - val_loss: 3.1288 - val_mse: 22.5440
Epoch 10/15
101/101 [==============================] - 0s 2ms/step - loss: 2.6917 - mse: 16.309
0 - val_loss: 2.6373 - val_mse: 17.3020
Epoch 11/15
101/101 [==============================] - 0s 2ms/step - loss: 2.5552 - mse: 14.527
4 - val_loss: 4.8624 - val_mse: 39.7955
Epoch 12/15
101/101 [==============================] - 0s 2ms/step - loss: 2.4907 - mse: 15.259
8 - val_loss: 3.7555 - val_mse: 24.6467
Epoch 13/15
101/101 [==============================] - 0s 2ms/step - loss: 2.5755 - mse: 14.680
9 - val_loss: 2.8443 - val_mse: 20.9270
Epoch 14/15
101/101 [==============================] - 0s 1ms/step - loss: 2.5284 - mse: 14.651
8 - val_loss: 3.1299 - val_mse: 20.5842
Epoch 15/15
101/101 [==============================] - 0s 1ms/step - loss: 2.6219 - mse: 15.678
```

In [3]:
```
score = model.evaluate(x_target, y_target, verbose=0)
print("MAE: %.1f" % (score[0]))
print("MSE: %.1f" % (score[1]))
```

```
MAE: 2.7
MSE: 18.7
```