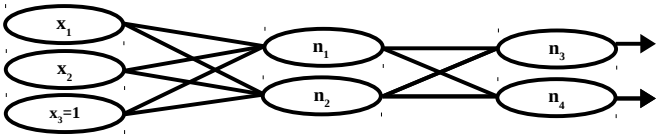
Partiel 8 octobre 2018 : Deep Learning

Jérôme Pasquet

Exercice 1 [8 points]



Nous considérons le réseau ci-dessus constitué de 4 neurones appelés : n_1, n_2, n_3, n_4. Nous notons $w_i^{(j)}$ les différents poids associés au neurone j, avec $i \in \{0..Nb\ d'entrées\ au\ neurone\ j\}$.

1. Nous notons $o^{(j)}$ la sortie du neurone j avant d'appliquer sa fonction d'activation et $s^{(j)}$ la sortie du neurone j après avoir appliqué la fonction d'activation, notée f. Donnez les équations de $o^{(j)}$ et $s^{(j)}$ pour chaque neurone. [1]

2. La fonction de perte est définie comme $J = \frac{1}{2}(s^{(3)} - y_1)^2 + \frac{1}{2}(s^{(4)} - y_2)^2$, avec y_1 et y_2 deux labels. Calculez la dérivée partielle $\frac{\delta J}{\delta w_i^{(3)}}$. [1]

3. En considérant la même fonction de perte, exprimez la dérivée partielle de $\frac{\delta J}{\delta w_1^{(1)}}$ en fonction de l'erreur engendrée par les neurones parents n_3 et n_4.[1]

4. Développez l'équation de $\frac{\delta J}{\delta w_i^{(1)}}$. [1]

5. Si nous considérons une fonction $tanh$, développez l'équation $\frac{\delta J}{\delta w_i^{(1)}}$. [1]

6. Nous considérons désormais la fonction $f(x) = PReLU(x) = \begin{cases} \alpha x & si\ x < 0 \\ x & sinon \end{cases}$.
D'après vous, quel est l'intérêt de cette fonction comparée au $ReLU$? [0.5 bonus]. Sachant que α est un (hyper-)paramètre entrainable, calculez la mise à jour effectuée sur ce paramètre au niveau des neurones n_1 et n_3. Nous notons $\alpha^{(j)}$ l'hyper-paramètre du neurone j. [2]

7. Comment appelle-t-on x_3? Quel est son utilité dans un réseau de neurones? [1]

Exercice 2 [4 points]

1. Décrivez les différents facteurs qui rendent le Deep Learning prédominent dans les problèmes de classification modernes. [1]

2. Expliquez pourquoi la fonction ReLU est plus utilisée que la fonction tanh dans les architectures modernes. [0.5]

3. Soit la fonction $f(x) = x^2 + 0.8x - 0.5$, trouvez le minimum global de cette fonction en expliquant votre méthode [0.5] et en détaillant les itérations effectuées. Vous partirez de la valeur $x_0 = 4$. [0.5]

4. Expliquez la notion de convolution 'étendue' dans les CNN. Utilisez un schéma. [0.5]

5. Appliquez la Conv 1 puis la Conv 2 à l'image en entrée. Donnez l'image finale. [1]

0	1	1	0	0	1	2	0
2	0	1	1	0	1	1	1
1	0	2	1	0	1	1	0
2	4	0	0	1	2	0	0
0	1	0	1	0	0	1	0
1	0	3	2	3	1	0	1
0	1	0	0	0	0	0	0
0	1	1	0	2	3	1	4

Conv 1
Sans Padding
Stride = 1

0	1	2
0	-2	0
-1	1	2

Conv 2
Sans Padding
Stride = 2

0	0	-1
1	0	-1
1	0	-1

Exercice 3 [8 points]

Aide : La fonction tf.split fonctionne de manière similaire à la fonction np.split. Celles-ci divisent un tableau multidimensionnel en un ou plusieurs points selon un axe donné. Par exemple

- np.split(np.array([2,3,4,5]), 2, axis=0) retournera deux tableaux [2, 3] et [4, 5].
- np.split(np.array([[2,3], [4,5], [6,7], [8,9]]), 2, axis=0) retournera deux tableaux [[2,3], [4,5]] et [[6,7], [8,9]].

1. Des erreurs se sont glissées dans le code. Corrigez les directement sur le code. **Attention : aucune faute n'a été mise dans la fonction** $getBatchTrain$ **et dans la variable** $loss$. [1]

2. Expliquez les labels en sortie de la fonction $getBatchTrain$. [1]

3. S'agit-il d'un problème de classification ? [0.25]

4. Quel est la taille des cartes de cartéristiques en sortie des couches $conv5$ et $conv2$. [0.75]

5. Donnez la taille des noyaux de convolution des couches $conv5$ et $conv2$. [0.5]

6. Que retourne les méthodes $conv5\_a.shape$ et $conv5.shape$? [0.5]

7. Expliquez la fonction de perte utilisée. [1] D'après vous quel est son utilité ?[1]

8. Quel est l'objectif des premières couches de convolution? que se passe-t-il lorsqu'on analyse des convolutions à des niveaux plus 'profonds' ?[1]

9. Expliquez la différence entre une Variable TensorFlow, un Placeholder et une variable python. [0.5]

10. Comment fonctionne la méthode 'run' des objets de type 'Session' sous tensorflow? [0.5] Décrivez l'action de cette méthode sur trois types différents d'objets que vous présenterez.[0.5 bonus]

```
import tensorflow as tf
import numpy as np
import random

cifar10 = tf.keras.datasets.cifar10.load_data()

train_data   = cifar10[0][0]
train_labels = cifar10[0][1]

BATCH_SIZE = 16

data  = tf.placeholder(tf.bool, shape=(BATCH_SIZE, 3, 32, 32))
label = tf.placeholder(tf.float32, shape=(BATCH_SIZE/2))


def getBatchTrain(size):
        z = np.zeros((size, 32,32,3))

        label_tmp_1 = np.zeros((size/2))
        label_tmp_2 = np.zeros((size/2))
        for i in range(0, size/2):
                r=random.randint(0, train_data.shape[0]-1)
                tmp = train_data[r]
                f = random.random()
                z[i] = tmp
                label_tmp_1[i] = train_labels[r]
        for i in range(0, size/2):
                r=random.randint(0, train_data.shape[0]-1)
                tmp = train_data[r]
                f = random.random()
                z[i+size/2] = tmp
                label_tmp_2[i] = train_labels[r]
        l = label_tmp_1==label_tmp_2

        return z, l


conv1 = tf.layers.conv2d(
      inputs=data,
      filters=32,
      kernel_size=[3, 3],
      padding="same",
      activation=tf.nn.relu,bias_initializer = tf.constant_initializer(value=-0.1))

pool1 = tf.layers.max_pooling2d(conv1, 2, 2)

conv2 = tf.layers.conv2d(
      inputs=pool1,
      filters=64,
      kernel_size=[3, 3],
```

```
        padding="same",
        activation=tf.nn.relu,  bias_initializer = tf.constant_initializer(value=-0.1))
pool2 = tf.layers.max_pooling2d(conv2, 2, 2)

conv3 = tf.layers.conv2d(
        inputs=pool2,
        filters=64,
        kernel_size=[3, 3],
        strides=1,
        padding="same",
        activation=tf.nn.relu, bias_initializer = tf.constant_initializer(value=-0.1))


conv4 = tf.layers.conv2d(
        inputs=conv3,
        filters=156,
        kernel_size=[3, 3],
        strides=1,
        padding="same",
        activation=tf.nn.relu,bias_initializer = tf.constant_initializer(value=-0.1))

pool4 = tf.layers.max_pooling2d(conv4, 2, 2)
conv5 = tf.layers.conv2d(
        inputs=pool4,
        filters=228,
        kernel_size=[3, 3],
        strides=1,
        padding="same",
        activation=tf.nn.relu, bias_initializer = tf.constant_initializer(value=-0.1))

conv5_a, conv5_b = tf.split(conv5, BATCH_SIZE/2, axis=0)

distance =  tf.sqrt(tf.reduce_sum((conv5_a-conv5_b)**2))
loss = (1.0-train_labels) * tf.maximum(0.0, 1.0 -distance)+train_labels * distance


optimizer = tf.train.AdamOptimizer(0.00001)
train     = optimizer.minimize(loss)


sess = tf.Session()

init = tf.initialize_all_variables()
sess.run(sess)

for i in range(0, 10000):
        z, l = getBatchTrain(size=BATCH_SIZE)
        _, loss_r = sess.run([train, loss], feed_dict={data:z, label:l})
        print i, loss_r
```