MSE - T-MachLe

PW-09

{dorian.magnin, noemien.kocher}@master.hes-so.ch

What is the learning algorithm being used to train the neural networks ?

On utilise la méthode `Sequential` de Keras. Selon la doc c'est :

The simplest type of model is the `Sequential` model, a linear stack of layers

On est donc dans une superposition linéaire de couches.

Selons nos recherche, l'algorithme utilisé est " Restricted Boltzman Machines" RBM. Pour ceci, nous utilisons deux couches: une visible et une cachée.

Nous pouvons paramétrer les différentes couches. Voici la couche visible:

```
model.add(Dense(300, input_shape=(784,), activation='relu'))
```

Cette ligne ajoute la première couche et spécifie une dimension de 300 en sortie et 784 en entrée (28x28). Le dernier paramètre spécifie la fonction d'activation.

La couche cachée est ensuite spécifiée:

```
model.add(Dense(n_classes, activation='softmax'))
```

En indiquant une dimension de 10 en sortie (= au nombre de classes).

What are the parameters (arguments) being used by that algorithm ?

Les paramètres (il y en a 3) sont les suivants:

It receives three arguments:

- An optimizer. This could be the string identifier of an existing optimizer (such as `rmsprop` or `adagrad`), or an instance of the `Optimizer` class. See: optimizers.
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as `categorical_crossentropy` or `mse`), or it can be an objective function. See: losses.
- A list of metrics. For any classification problem you will want to set this to `metrics=['accuracy']`. A metric could be the string identifier of an existing metric or a custom metric function.

What cost function is being used ? please, give the equation(s).

Dans notre cas, on utilise comme fonction de coût categorical_crossentropy:

$$\mathcal{L}(X,Y) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \ln a(x^{(i)}) + \left(1 - y^{(i)}\right) \ln\left(1 - a(x^{(i)})\right)$$

X={x(1),...,x(n)} est le set d'entrée dans le training set

Y={y(1),...,y(n)} est le set avec les labels correspondants

a(x) représente le sortie du réseau de neurones.

Model complexity: for each experiment (shallow network learning from raw data, shallow network learning from features and CNN), select a neural network topology and describe the inputs, indicate how many are they, and how many outputs.

Shallow from raw data

Input (what, how many):

```
model.compile(loss='categorical_crossentropy', optimizer=RMSprop(),
metrics=['accuracy'])
history = model.fit(X_train, Y_train,
                    batch_size=batch_size, epochs=n_epoch,
                    verbose=1, validation_data=(X_test, Y_test))
```

Train on 60000 samples, validate on 10000 samples

X_train est une matrice de dimension (60000, 784). **784** = 28*28 pixels

Y_train est une matrice de dimension (6000 x 10)

Total params: 80,610 = 78'600 + 2010

Output (how many): 10 (les différentes classes)

Shallow from features

Input (what, how many):

```
history = model.fit(X_train_hog, Y_train,
                    batch_size=batch_size, nb_epoch=n_epoch,
                    verbose=1, validation_data=(X_test_hog, Y_test))
```

Train on 60000 samples, validate on 10000 samples

X_train_hog est une matrice de dimension (6000 x 392)

Comme nous avons 8 orientations et 4 pixel par cellulle, nous avons donc 28*28*8/(4*4) = **392**

Y_train est une matrice de dimension (6000 x 10)

Output (how many): 10 (les différentes classes)

CNN

Input (what, how many)

```
history = model.fit(X_train, Y_train,
                    batch_size=batch_size, nb_epoch=n_epoch,
                    verbose=1, validation_data=(X_test, Y_test))
```

X_train est une matrice de dimension (60000, 28, 28, 1)

Y_train est une matrice de dimension (60000, 10)

Total params: 7,465 ce qui correspond à 234 + 2034 + 1312 + 3625 + 260 = **7'465**

Output (how many): 10 (les différentes classes)

Compute the number of weights of each model (e.g., how many weights between the input and the hidden layer, how many weights between each pair of layers, biases, etc..) and explain how do you get to those values.

Shallow from raw data

Tous les paramètres sont un poid. Donc (784+1)*300 = 235'500 poids entre l'input et la couche cachée + (300+1) * 10 qui sont les poids entre la couche cachée et la sortie.

Shallow from features

Comme pour le raw data, les paramètres sont des poinds, donc: (394+1)*200 = 78’600 de l’input à la couche cachée et (200+1)*10 = 2’010 poids-

CNN

7’465, qui est le nombre de paramètres.

Are the deep neural networks much more complex than the shallow ones ? explain with an example.

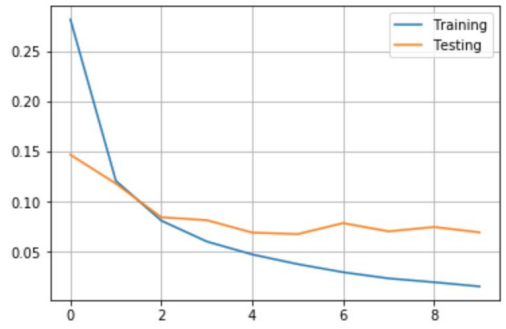
Non, pas forcément. En réduisant la dimension au fil des étapes, le CNN n’est pas beaucoup plus compliqué.

Test every notebook for three different meaningful cases, describe the model and present the performance of the system (e.g., plot of the evolution of the error, final evaluation scores and confusion matrices). Comment the differences in results. Are there particular digits that are frequently confused ?

Shallow from raw data

On utilise un “linear stack of layer”. Dans ce cas on en a deux et on cherche à minimiser la fonction de coût categorical_crossentropy. On utilise RMSprop comme optimizer et on a a deux couches avec *relu* comme fonction d’activation pour la première et *softmax* pour la deuxième.

```
Test score: 0.0696496090101
Test accuracy: 0.9807
```

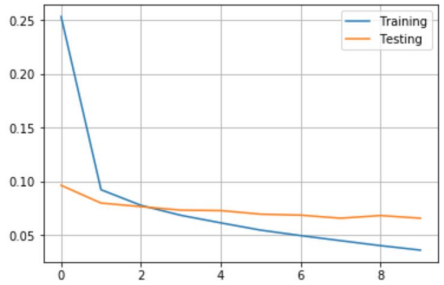



```
array([[ 972,    0,    1,    0,    0,    0,    2,    1,    4,    0],
       [   0, 1122,    3,    0,    0,    0,    2,    1,    7,    0],
       [   4,    1, 1009,    3,    1,    0,    2,    1,   11,    0],
       [   1,    0,    2,  989,    0,    6,    0,    4,    6,    2],
       [   2,    0,    4,    0,  958,    0,    4,    3,    2,    9],
       [   2,    0,    0,    4,    1,  878,    2,    0,    3,    2],
       [   5,    1,    0,    1,    1,    5,  943,    0,    2,    0],
       [   0,    5,   10,    1,    0,    0,    0,  992,   12,    8],
       [   4,    0,    3,    1,    0,    2,    1,    2,  957,    4],
       [   4,    2,    0,    2,    3,    3,    1,    2,    5,  987]])
```

Shallow from features

C’est comme pour le *raw data,* seulement que les données d’entrées sont différentes (HOG). On a aussi deux couches avec la première qui a *relu* comme fonction d’activation et la deuxième couche *softmax*.

```
Test score: 0.0657455540907
Test accuracy: 0.9805
```




```
array([[ 973,    0,    1,    0,    0,    2,    1,    0,    2,    1],
       [   0, 1127,    1,    1,    0,    0,    1,    2,    3,    0],
       [   3,    1, 1011,    1,    1,    0,    1,    8,    6,    0],
       [   0,    1,    0,  991,    0,    5,    0,    4,    7,    2],
       [   3,    2,    1,    0,  960,    0,    1,    2,    1,   12],
       [   2,    1,    0,    7,    0,  871,    2,    1,    6,    2],
       [   5,    4,    0,    0,    4,    3,  940,    0,    2,    0],
       [   1,    7,    3,    1,    5,    0,    0, 1003,    2,    6],
       [   7,    0,    2,    6,    2,    1,    0,    3,  947,    6],
       [   0,    5,    1,    5,    5,    3,    0,    7,    1,  982]])
```

CNN

On est dans le cas d'un CNN avec 10 couches qui sont les suivantes:

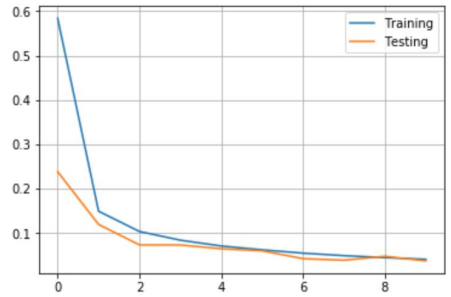
Layer (type)	Output Shape	Param #
l0 (InputLayer)	(None, 28, 28, 1)	0
l1 (Conv2D)	(None, 28, 28, 9)	234
l1_mp (MaxPooling2D)	(None, 14, 14, 9)	0
l2 (Conv2D)	(None, 14, 14, 9)	2034
l2_mp (MaxPooling2D)	(None, 7, 7, 9)	0
l3 (Conv2D)	(None, 7, 7, 16)	1312
l3_mp (MaxPooling2D)	(None, 3, 3, 16)	0
flat (Flatten)	(None, 144)	0
l4 (Dense)	(None, 25)	3625
l5 (Dense)	(None, 10)	260

MaxPooling2D permet de réduire la dimension.
Conv2D permet de créer une matrice de convolution.
Dense effectue l'opération avec la matrice de convolution (on un kernel):

```
activation(dot(input, kernel) + bias)
```

Le premier dense utilise *relu* et le dernier *softmax*. *Conv2D* utilise à chaque fois *relu*.

```
('Test score:', 0.037023998715693597)
('Test accuracy:', 0.98829999999999996)
```




```
array([[ 977,    0,    1,    1,    0,    0,    1,    0,    0,    0],
       [   0, 1132,    1,    0,    2,    0,    0,    0,    0,    0],
       [   3,    8, 1014,    0,    0,    0,    0,    5,    1,    1],
       [   0,    0,    0, 1000,    0,    3,    0,    4,    3,    0],
       [   1,    0,    0,    0,  978,    0,    1,    0,    0,    2],
       [   1,    0,    0,    7,    0,  878,    1,    1,    1,    3],
       [   5,    5,    0,    0,    2,    3,  942,    0,    0,    1],
       [   0,    2,    4,    0,    3,    0,    0, 1016,    1,    2],
       [   3,    0,    1,    2,    0,    0,    1,    3,  960,    4],
       [   3,    2,    0,    1,    9,    1,    0,    6,    1,  986]])
```

Conclusion

On remarque que c'est le CNN qui a la meilleure performance et le *test score* le plus bas. Le *4* est souvent confondu avec le *9* et le *7* est souvent confondu avec le *2*. Mais de manière générale, les digits sont bien identifiés.