MACHINE LEARNING (T-MACHLE)

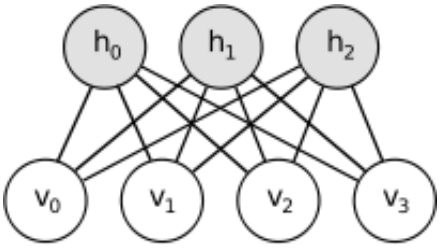
Practical work 10 - Deep Neural Networks

STUDENT: ***ROMAIN CLARET***
PROFESSOR: ***A. PEREZ-URIBE & J. HENNEBERT***
ASSISTANT: ***H. SATIZABAL***
DUE-DATE: ***MONDAY 03 DECEMBER 2018***

DIGIT RECOGNITION FROM RAW DATA

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

I am confused about this question, I can't figure out what you are asking for here. After reading the Keras documentation and the class slides, I couldn't make sense of this question. Indeed, I don't know if you want us to go in detail about what is the name of the algorithm used to make the sequential model possible in Keras, by connecting layers to each other with the Restricted Boltzmann Machines (RBM) algorithm.



Or if you want just to know what is the learning algorithm:

RMSprop()

Because based on the following question, I could make sense for both solutions, and it was a time-consuming reflexion. After discussion with my classmates, I will go for the optimizer solution.

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

  Default arguments

  `lr=0.001, rho=0.9, epsilon=None, decay=0.0`

  - lr: float >= 0. Learning rate.

  - rho: float >= 0.
  - epsilon: float >= 0. Fuzz factor. If None, defaults to K.epsilon().
  - decay: float >= 0. Learning rate decay over each update.
- What cost function is being used?
  - categorical_crossentropy
    - Warning in the documentation: Note: when using the categorical_crossentropy loss, your targets should be in the categorical format (e.g. if you have 10 classes, the target for each sample should be a 10-dimensional vector that is all-zeros except for a 1 at the index corresponding to the class of the sample).

- Please, give the equation(s)

$$CCE = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J} y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)$$

Debunking loss functions in deep learning

> *Cost function works in tandem with softmax activation function. Softmax is a generalized sigmoid activation function for K outputs. It is required because individually outputs would not sum up to 1 so we need to normalize it and that's what softmax does for us.*

MODEL COMPLEXITY

Digit recognition from raw data

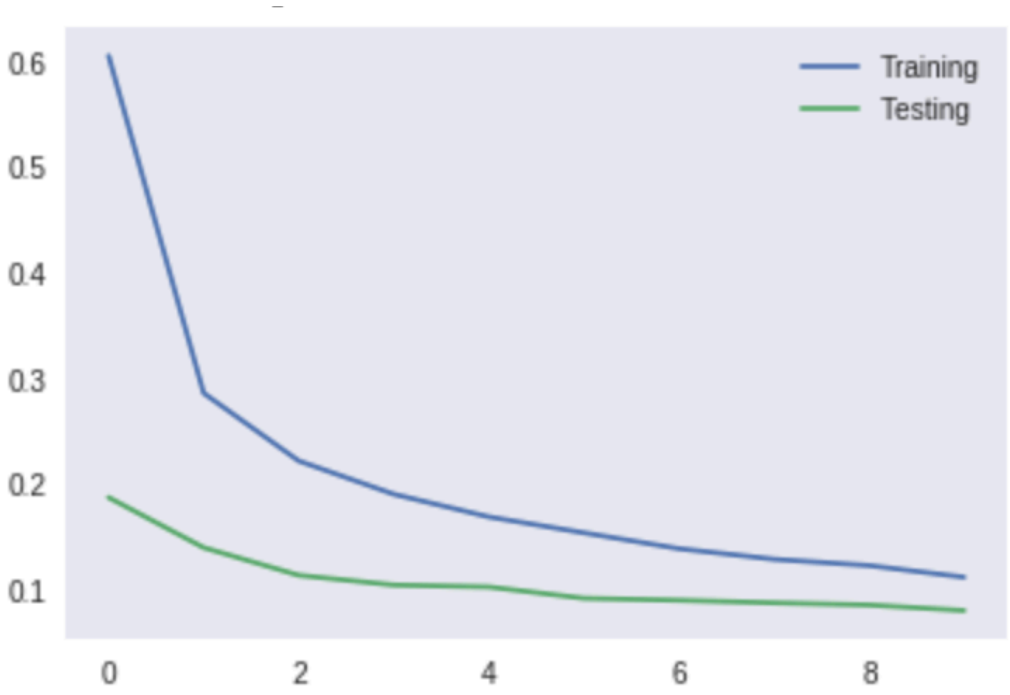
I spent much more time than I should playing with layers and the arguments in hope to find the best outcome.

```
model = Sequential()
model.add(Dense(300, input_shape=(784,), activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(30, activation='relu'))
model.add(Dropout(0.4))
```

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

batch_size = 128
n_epoch = 10

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

Digit recognition from raw data

- Train on 60000 samples, validate on 10000 samples
- Test score: 0.0754462222365255
- Test accuracy: 0.98
- Numbers being mainly confused:
  - 2 -> 7
  - 3 -> 5
  - 4 -> 9
  - 6 -> 4, 5 8
  - 7 -> 3
  - 9 -> 7



```
array([[ 969,    1,    0,    1,    0,    0,    3,    2,    2,    2],
       [   0, 1128,    4,    0,    0,    0,    1,    0,    2,    0],
       [   1,    2, 1013,    2,    3,    0,    3,    6,    2,    0],
       [   0,    0,    4,  991,    0,    4,    0,    6,    3,    2],
       [   0,    0,    4,    0,  963,    0,    5,    1,    1,    8],
       [   2,    1,    0,    6,    1,  864,    8,    1,    6,    3],
       [   4,    3,    0,    1,    4,    2,  941,    0,    3,    0],
       [   1,    4,   12,    1,    0,    0,    0,  999,    3,    8],
       [   2,    1,    1,    3,    4,    1,    6,    3,  947,    6],
       [   2,    5,    0,    3,    9,    0,    1,    3,    1,  985]])
```

Topology

LAYER (TYPE)	OUTPUT SHAPE	PARAM #
dense_264 (Dense)	(None, 300)	235500
dropout_127 (Dropout)	(None, 300)	0
dense_265 (Dense)	(None, 30)	9030
dropout_128 (Dropout)	(None, 30)	0
dense_266 (Dense)	(None, 10)	310

- Total params: 244,840
- Trainable params: 244,840
- Non-trainable params: 0

Input

- X_train: (60000, 28*28)
- Y_train: (60000, 10)

Output

- Classes: 10

Weights

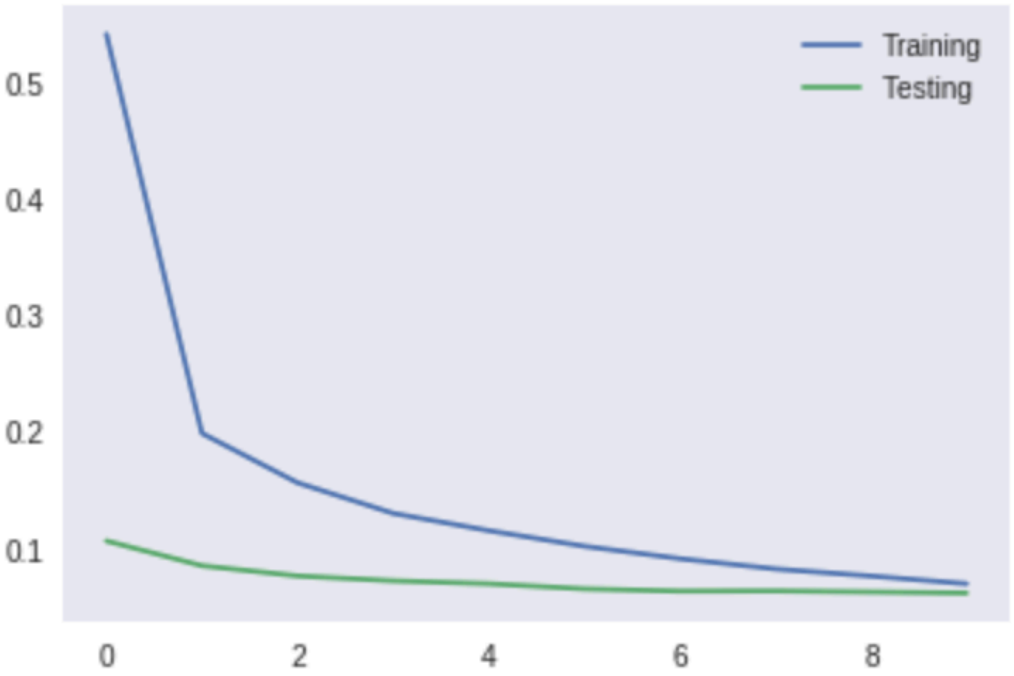
- All parameters are weights
  - INPUT -> HIDDEN_1: 235500
  - HIDDEN_1 -> HIDDEN_2: 9030
  - HIDDEN_2 -> OUTPUT: 310
- Total weights: 244,840

Digit recognition from features of the input data

```
model = Sequential()
model.add(Dense(300, input_shape=(hog_size,), activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(30, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(n_classes, activation='softmax'))

batch_size = 128
n_epoch = 10

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

Performance

- Train on 60000 samples, validate on 10000 samples
- Test score: 0.0607265819466993
- Test accuracy: 0.9834
- Numbers being mainly confused:
  - 0 -> 8 and 9
  - 1 -> 7
  - 3 -> 5 and 8
  - 4 -> 9
  - 7 -> 9




```
array([[ 974,    0,    0,    0,    0,    1,    3,    0,    2,    0],
       [   1, 1125,    1,    1,    0,    0,    2,    1,    4,    0],
       [   1,    1, 1020,    0,    1,    0,    2,    5,    2,    0],
       [   0,    0,    1,  993,    0,    8,    0,    3,    4,    1],
       [   2,    0,    0,    0,  965,    0,    2,    2,    2,    9],
       [   3,    1,    0,    7,    0,  870,    4,    1,    5,    1],
       [   4,    2,    0,    0,    2,    3,  947,    0,    0,    0],
       [   2,    5,    3,    0,    3,    0,    0, 1008,    2,    5],
       [   7,    0,    1,    5,    0,    2,    0,    3,  955,    1],
       [   6,    3,    0,    4,    6,    1,    0,    8,    4,  977]])
```

Topology

LAYER (TYPE)	OUTPUT SHAPE	PARAM #
dense_14 (Dense)	(None, 300)	117900
dropout_15 (Dropout)	(None, 300)	0
dense_15 (Dense)	(None, 30)	9030
dropout_8 (Dropout)	(None, 30)	0
dense_16 (Dense)	(None, 10)	310

- Total params: 127,240
- Trainable params: 127,240
- Non-trainable params: 0

Input

- X_train: (60000, 392)
- Y_train: (60000, 10)

Output

- Classes: 10

Weights

- All parameters are weights
  - INPUT -> HIDDEN_1: 117900
  - HIDDEN_1 -> HIDDEN_2: 9030
  - HIDDEN_2 -> OUTPUT: 310
- Total weights: 127,240

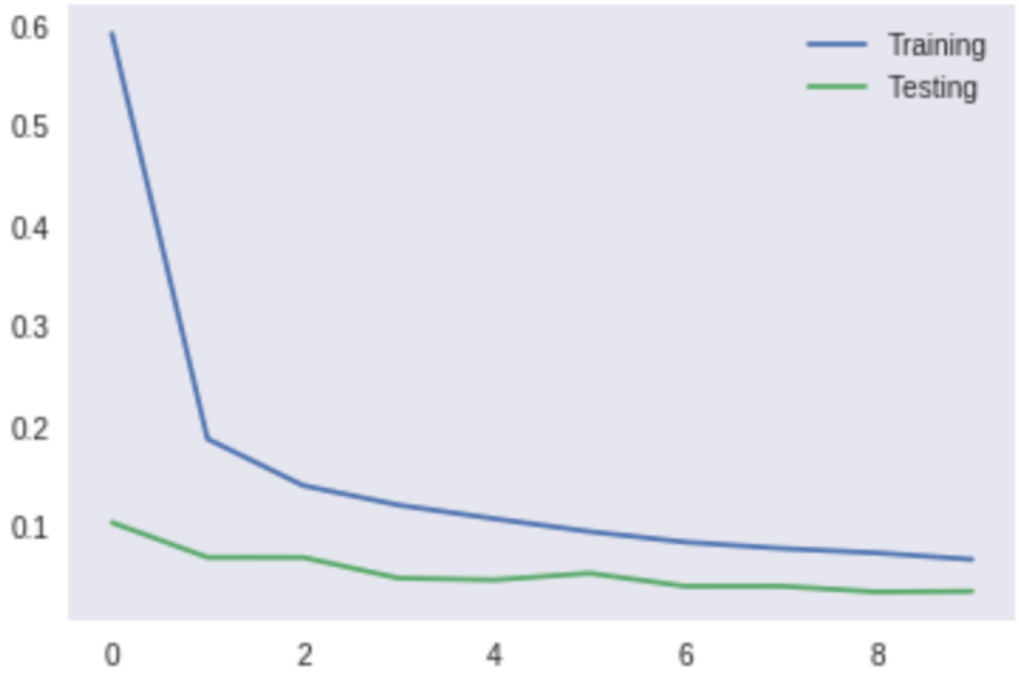
Convolutional neural network digit recognition

```
l0 = Input(shape=(height, width, 1), name='l0')
l1 = Convolution2D(9, 5, 5, border_mode='same', activation='relu',
name='l1')(l0)
l1_mp = MaxPooling2D(pool_size=(2, 2), name='l1_mp')(l1)
l2 = Convolution2D(9, 5, 5, border_mode='same', activation='relu',
name='l2')(l1_mp)
l2_mp = MaxPooling2D(pool_size=(2, 2), name='l2_mp')(l2)
l3 = Convolution2D(16, 3, 3, border_mode='same', activa-
tion='relu', name='l3')(l2_mp)
l3_mp = MaxPooling2D(pool_size=(2, 2), name='l3_mp')(l3)
flat = Flatten(name='flat')(l3_mp)
l4 = Dense(300, activation='relu', name='l4')(flat)
l5 = Dropout(0.4, name='l5')(l4)
l6 = Dense(30, activation='relu', name='l6')(l5)
l7 = Dropout(0.4, name='l7')(l6)
l8 = Dense(n_classes, activation='softmax', name='l8')(l7)

batch_size = 128
n_epoch = 10

model.compile(loss='categorical_crossentropy', optimizer=Adam(),
metrics=['accuracy'])
```

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

Performance

- Train on 60000 samples, validate on 10000 samples
- Test score: 0.03363577147482033
- Test accuracy: 0.9904
- Numbers being mainly confused:
  - 9 -> 7 and 4
  - 7 -> 3
  - 6 -> 0




```
array([[ 972,    0,    1,    0,    0,    0,    5,    1,    1,    0],
       [   0, 1129,    0,    2,    0,    0,    1,    0,    2,    1],
       [   0,    1, 1028,    0,    1,    0,    0,    2,    0,    0],
       [   0,    0,    2, 1001,    0,    5,    0,    0,    1,    1],
       [   0,    0,    0,    0,  977,    0,    1,    0,    2,    2],
       [   1,    0,    0,    1,    0,  887,    1,    0,    0,    2],
       [   4,    3,    0,    0,    1,    0,  950,    0,    0,    0],
       [   0,    3,    8,    1,    1,    0,    0, 1007,    1,    7],
       [   1,    0,    1,    2,    0,    1,    0,    0,  967,    2],
       [   1,    2,    0,    0,   14,    0,    0,    2,    4,  986]])
```

Topology

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, 300)	43500
l5 (Dropout)	(None, 300)	0
16 (Dense)	(None, 30)	9030
l7 (Dropout)	(None, 30)	0
l8 (Dense)	(None, 10)	310

- Total params: 56,420
- Trainable params: 56,420
- Non-trainable params: 0

Input

- X_train: (60000, 28, 28, 1)
- Y_train: (60000, 10)

Output

- Classes: 10

Weights

- All parameters are weights
  - INPUT -> HIDDEN_1: 234
  - HIDDEN_1 -> HIDDEN_2: 2034
  - HIDDEN_2 -> HIDDEN_3: 1312
  - HIDDEN_3 -> HIDDEN_4: 43500
  - HIDDEN_4 -> HIDDEN_5: 9030
  - HIDDEN_5 -> OUTPUT: 310
- Total weights: 56,420

CONVOLUTIONAL NEURAL NETWORK DIGIT RECOGNITION

- Are the deep neural networks much more complex (i.e., do they have more weights?) than the shallow ones?
  - No, in contrary, they are reducing the weight amount drastically