Deep learning Practical Assignment #2:

Linear Classifier : Fully connected networks

Lab date : October 25th, 2021
Deadline to submit : November 25th, 2021 - 11:59 PM (23:59)

Instructions

Read all the instructions below carefully before you start working on the assignment.

- You must do this assignment in groups of **at most 3** students.
- You must implement all algorithms **from scratch**.
- Please submit a jupyter notebook that contains:
  - A section (multiple cells) for your answer to the theoretical questions.
  - A section (multiple cells) for the installation of the used packaged. please specify the exact package version to avoid possible conflicts.
  - A section (multiple cells) for you implementation.
- Please submit your notebook to your corresponding git branch in our git work-space.
- Late submissions will be graded as follows:
  - On time submission grade SG.
  - $SG = 0.95 * SG$ for each day. Capped at $\frac{2}{3} * SG$
- Early submissions will be graded as follows:
  - On time submission grade SG.
  - $SG = 1.02 * SG$ for each day. Capped at $\frac{5}{4} * SG$
- Code that does not work will be graded accordingly. Please focus on the algorithms you are implementing.

Practical assignment objective

- To create a fully connected network.
- Train different model on different dataset.

Datasets

In this lab we'll use different datasets

- IRIS dataset
- Digit dataset
- Olivetti Faces dataset

1 Implementing fully connected dataset !!

In this segment we implement the perceptron algorithm.

Implement from scratch only the perceptron algorithm, for data splitting and suffling and other operations, you can use 3rd party libraries

1.1 Toy data set

Consider a data set $\mathbb{S} = \{(\mathbf{x_i}, y_i)\}_{i=1}^{250}$ consisting of 250 points $\mathbf{x_i} = (x_1, x_2)$ and their labels y_i.

- The first 125 $\mathbf{x_i}$ have label $y_i = -1$ and are generated according to a Gaussian distribution $\mathbf{x_i} \sim \mathcal{N}(\mu_1, \sigma_1^2\mathbf{I})$, where

$$\mu_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

- The last 125 $\mathbf{x_i}$ have label $y_i = 1$ and are generated according to a Gaussian distribution $\mathbf{x_i} \sim \mathcal{N}(\mu_2, \sigma_2^2\mathbf{I})$, where

$$\mu_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

After shuffling the data-set, split it into train and test set, containing 80% and 20% of the dataset.

1.2 Using keras

```
from keras.models import Sequential
from keras.layers import Dense

...
#loading and understanding the data
...

# Sample code to create a fully connected network
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
```

```
model.add(Dense(5, activation='relu'))
model.add(Dense(3, activation='relu'))
...

# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
...

# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
```

2 Lab description

Using the provided IRIS data-set, train different models (a model is defined by nb of layers, nb of neurons in each layer, error function, ...).

2.1 Selecting best model

Using different metrics: accuracy, recall, precision, ..., present your best architecture you have created.
Please comment the results.

3 Working towards the CNN

The DIGIT data-set provided in the datasets section contains multiple images of handwritten digits.
In this section, we'll show the limits of the fully connected networks.
First use this DIGIT dataset to create your best model using the accuracy as a metric.
Describe what happens when we increase the number of parameters (nb layers and nb neurons).
For your best model please provide the characteristics of its architectures:

- NB Layer
- NB Neurons
- Activation functions (Non linearity functions)
- Error function (Cost function)
- optimization algorithm

Using the face dataset, find a model that can acheive at least 30% accuracy. Describe your model