Notes de cours CADL - session-1

cours Kadenze - session-1

Thibaut Marmey

October 24, 2018

- Learn the basic idea behind machine learning: learning from data and discovering representations
- Learn how to preprocess a dataset using its mean and standard deviation
- Learn the basic components of a Tensorflow Graph

Contents

1 Introduction

1.1 Généralités

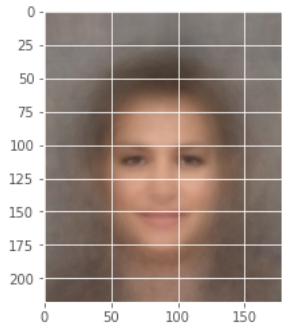
- Deep-learning in a type of Machine Learning
- *Deep* because it is composed of many layers of *Neural Networks*
- Other valuable branches of Machine Learning :
  - Rinforcement Learning
  - Dictionary Learning
  - Probabilistic Graphical Models and Bayesian Methods (Bishop)
  - Genetic and Evolutionary Algorithms
- The differents ways an object can appear in an image is called *invariance*
- The dataset teaches the algorithm how to see the world, but only the world of this dataset
- Existing data :
  - MNIST
  - CalTech
  - CelebNet

- ImageNet
- LFW
- CIFAR10, CIFRA100, MS Coco...

1.2 Preprocessing Data

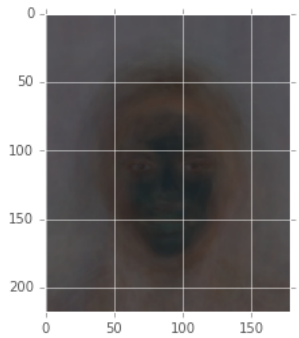
- Collect the images into a batch configuration. With this configuration, it's easier to make some computation over all the data.
  This means, the data is in a single *numpy* variable : *data = np.array(imgs)*
- Compute the Mean and Deviation of Images (of the batch channel)

  ```
  mean_img = np.mean(data, axis=0) #mean of each col
  plt.imshow(mean_img.astype(np.uint8))
  ```

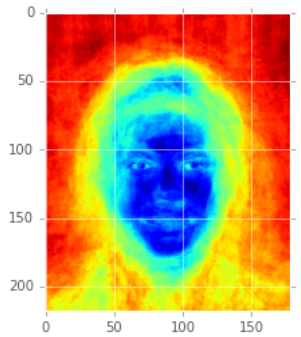
  This describes what most the dataset looks like.

  ```
  std_img = np.std(data, axis=0)
  plt.imshow(std_img.astype(np.uint8))
  ```

  This describes where the changes are the most likely to appear in the dataset of images.

  ```
  plt.imshow(np.mean(std_img, axis=2).astype(np.uint8))
  ```



This describes how every color channel will vary as a heatmap.

* Red part : not the best representation of the image
* Blue part : the less likely that our mean image is far off from any other possible image

1.3 Dataset preprocessing

- We are trying to build a model that understands invariances (different of vision of an object, localization in the image, etc...)
- If we use DL to learn something complex in the data, it starts by modeling both the mean and standard deviation or our dataset.
- Speed up by "preprocessing" the dataset by removing the mean and standard deviation : it's called *normalization*.
  Subsctracting the mean and dividing by the standard deviation.
- Look at the dataset with another way : array into a 1 dimensional array.

```
flattened = data.ravel()
```

- Visualize the **"distribution"**, or range and frequence of possible values. This tell us if **the data is predictable or not**.
  plt.hist(data.ravel(), n takes the min and max values of the *data* array, and divide this interval in n subintervals.

```
plt.hist(flattened.ravel(), 255) #values are grouping in 255 bins
```

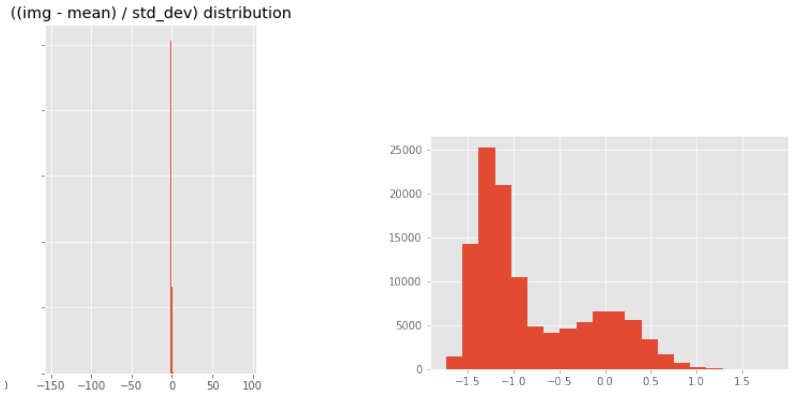
  It tells us if something seems to happen more than anything else. If it does, the neural network will take advantage of that.

- Normalization :

```
plt.hist(((data[0] - mean_img) / std_img).ravel(), bins)
```

  The data has been squished into a peak. Change the scale of the hist.
  The data is concentrated between two values. The effect of normalizing : most of the data will be around 0, where some deviations of it will follow between the two

values.



(a) (b)

- If the normalization doesn't look like this :
  - get more data to calculate our mean/std deviation
  - try another method of normalization
  - not bother with normalization at all
- Other options of normalization :
  - local contrast normalization for images
  - PCA based normalization