Prof. Dr. Rudolf Mathar, Dr. Arash Behboodi, Emilio Balda

Exercise 1

Friday, October 20, 2017

Problem 1. *MNIST dataset (tensorflow version)*

The MNIST dataset is commonly used for benchmarking purpose in machine learning research. It contains images of handwritten digits from 0 to 9. Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive. The data files `train.csv` and `test.csv` contain these gray-scale images of hand-drawn digits. The first file contains data that should be used for designing (or training) a model, while the second dataset is used for testing the obtained solution.

In this example we use the *tensorflow* library provided by Google for loading the MNIST dataset, as well as the python libraries *pyplot* and *numpy* (for plotting and computing numerical operations).

```
In [1]: import tensorflow as tf
        import matplotlib.pyplot as plt
        import numpy as np
```

When using tensorflow there is a built-in solution for importing the MNIST dataset within the tensorflow library. This spares us the work of loading and formating the data contained in the aforementioned CSV files. Instead, we may import a data structure from the tensorflow library as

```
In [2]: from tensorflow.examples.tutorials.mnist import input_data
        data = input_data.read_data_sets("data/MNIST/", one_hot=True)
```

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting data/MNIST/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting data/MNIST/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting data/MNIST/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting data/MNIST/t10k-labels-idx1-ubyte.gz
```

Note that, when loading the data, we enabled the `one_hot` option. *One-hot* refers to the formatting style of the label vector(s) provided. For this dataset, there are 10 possible labels corresponding to handwritten digits between 0 and 9. Then, for labeling an image,

it is sufficient to assign an integer number between 0 and 9 corresponding to its label. Nevertheless, when using the aforementioned one-hot representation of the labels, every image is labeled using a 10-dimensional vector with the value 1 in the entry corresponding to its assigned label (i.e., the "hot" entry) and zero elsewhere. Later in this course we will discuss the role of the one-hot format in the context supervised learning.

Up to this point, we have extracted the MNIST datset, which is composed of

- 70.000 images and associated labels.

When loading this dataset from the examples provided by the tensorflow library, these 70.000 images and labels are already portioned into 3 datasets:

```
In [3]: print("Size of:")
        print("- Training-set:\t\t{}".format(len(data.train.labels)))
        print("- Test-set:\t\t{}".format(len(data.test.labels)))
        print("- Validation-set:\t{}".format(len(data.validation.labels)))
```

```
Size of:
- Training-set:                55000
- Test-set:                    10000
- Validation-set:               5000
```

Each element of the dataset is a vector of dimension $784 = 28 \times 28$:

```
In [4]: print("shape of first entry:",data.train.images[0,:].shape)
        print("shape of second entry:",data.train.images[1,:].shape)
        print("shape of third  entry:",data.train.images[2,:].shape)
```

```
shape of first entry: (784,)
shape of second entry: (784,)
shape of third  entry: (784,)
```

As discussed, the label vectors inside `data.train.labels` are in a one-hot format. We can extract the label number (between 0 and 9) by searching for the entry with maximum value within the one-hot vector:

```
In [5]: print("label of first entry:",data.train.labels[0,:])
        print("label of first entry:",data.train.labels[0,:].argmax())
        print("label of second entry:",data.train.labels[1,:])
        print("label of second entry:",data.train.labels[1,:].argmax())
```

```
label of first entry: [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
label of first entry: 7
label of second entry: [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
label of second entry: 3
```

We now show that this images correspond to the handwritten digits '7' and '3' as expected. Note that, in order to display these digits we must reshape the 784-dimensional image vectors back into 28×28 images.

```
In [6]: plt.imshow(data.train.images[0,:].reshape([28,28]),cmap='binary')
        plt.show()
        plt.imshow(data.train.images[1,:].reshape([28,28]),cmap='binary')
        plt.show()
```