Chair for Theoretical Information Technology
RWTH AACHEN UNIVERSITY

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

Exercise 4

Friday, November 17, 2017

Problem 1. *MNIST visualization (tensorflow version)*

We start by loading the libraries we need.

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

        from tensorflow.contrib.tensorboard.plugins import projector
        from tensorflow.examples.tutorials.mnist import input_data
```

We define the directory where the visualization files are going to be stored, a name for our visualization model, and the number of samples we are going to visualize. For this example we visualize the first 500 samples of the dataset.

```
In [2]: LOG_DIR = 'MNIST_tensorflow/mds_sample'
        NAME_TO_VISUALISE_VARIABLE = "mnistembedding"
        TO_EMBED_COUNT = 500
```

The tensorflow visualization library (called tensorboard) has the option to display miniature figures (called *sprites*) associated to each data point in the space, instead of the standard bullet point representation. In addition, we can also provide a metadata file that contains the labels associated to each data point. This metadata is later used for coloring the data points according to the class they belong. Futhermore, we now provide tensorflow a path to the sprites file and metadata file.

```
In [3]: path_for_mnist_sprites =  os.path.join(LOG_DIR,'mnistdigits.png')
        path_for_mnist_metadata =  os.path.join(LOG_DIR,'metadata.tsv')
```

Now, we load the MNIST data and store the first 500 samples in a *bacth* of data points. We do the same with their corresponding labels.

```
In [4]: mnist = input_data.read_data_sets("MNIST_tensorflow/MNIST_data/",
                                          one_hot=False)
        batch_xs = mnist.train.images[:TO_EMBED_COUNT, :]
        batch_ys = mnist.train.labels[:TO_EMBED_COUNT]

        print(batch_ys[:5])
        print(np.array(batch_xs).shape)
```

```
Extracting MNIST_tensorflow/MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_tensorflow/MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_tensorflow/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_tensorflow/MNIST_data/t10k-labels-idx1-ubyte.gz
[7 3 4 6 1]
(500, 784)
```

Now we apply multidimensional scaling to the 500 samples within the batch.

```
In [5]: X = np.array(batch_xs)
        n, k = X.shape


        # Define distance metric
        dist_metric = lambda x,y: np.power(np.linalg.norm(x-y,ord=2),2)

        # Construct Dissimilarity Matrix
        Delta = np.zeros([n,n])
        for i in range(n):
            for j in range(i):
                Delta[i, j] = dist_metric(X[i,:],X[j,:])
                Delta[j, i] = Delta[i, j]

        # Construct En
        En = np.eye(n) - 1/n * np.ones([n,n])


        # Apply multidimensional scaling
        eigenValues, eigenVectors = np.linalg.eig(-1/2*np.dot(np.dot(En,Delta),En))

        idx = eigenValues.argsort()[::-1]  # for sorting eig-values/vectors
        eigenValues = eigenValues[idx] # for sorting eig-values/vectors
        eigenVectors = eigenVectors[:,idx] # for sorting eig-values/vectors


        Xout = np.dot(eigenVectors[:,:3], np.diag(np.sqrt(eigenValues[:3])))

        print(Delta[:5,:5])
        print(X.shape)
        print(Xout.shape)
```

```
[[   0.          140.62154565  119.40662467  114.5790765   118.79807926]
 [ 140.62154565    0.           73.76719796  118.21443826   93.18825017]
 [ 119.40662467   73.76719796    0.           96.17294996  102.17164071]
 [ 114.5790765   118.21443826   96.17294996    0.           78.64832575]
 [ 118.79807926   93.18825017  102.17164071   78.64832575    0.        ]]
(500, 784)
```

```
(500, 3)
```

Note that, to carry out PCA (instead of MDS) there is no need to apply a PCA script ourselves, since tensorboard has already a built-in option for PCA. Therefore, for the case of PCA visualization we can leave the samples within the batch as they are.

```
In [6]: # Uncomment the following to carry out PCA instead of MDS:
        # Xout = np.array(batch_xs) # No MDS, tensorflow does PCA
```

We provide the variable to visualize and the visualization directory to tensorboard.

```
In [7]: embedding_var = tf.Variable(Xout, name=NAME_TO_VISUALISE_VARIABLE)
        summary_writer = tf.summary.FileWriter(LOG_DIR)
```

Finally, we setup the visualization files.

```
In [8]: config = projector.ProjectorConfig()
        embedding = config.embeddings.add()
        embedding.tensor_name = embedding_var.name

        # Specify where you find the metadata
        embedding.metadata_path = 'metadata.tsv'

        # Specify where you find the sprite (we will create this later)
        embedding.sprite.image_path = 'mnistdigits.png'
        embedding.sprite.single_image_dim.extend([28,28])

        # Say that you want to visualise the embeddings
        projector.visualize_embeddings(summary_writer, config)

        sess = tf.InteractiveSession()
        sess.run(tf.global_variables_initializer())
        saver = tf.train.Saver()
        saver.save(sess, os.path.join(LOG_DIR, "model_mds.ckpt"), 1)

        def create_sprite_image(images):
            """Returns a sprite image with all miniatures images."""
            if isinstance(images, list):
                images = np.array(images)
            img_h = images.shape[1]
            img_w = images.shape[2]
            n_plots = int(np.ceil(np.sqrt(images.shape[0])))


            spriteimage = np.ones((img_h * n_plots ,img_w * n_plots ))
```

```
            for i in range(n_plots):
                for j in range(n_plots):
                    this_filter = i * n_plots + j
                    if this_filter < images.shape[0]:
                        this_img = images[this_filter]
                        spriteimage[i * img_h:(i + 1) * img_h,
                          j * img_w:(j + 1) * img_w] = this_img

            return spriteimage

        def vector_to_matrix_mnist(mnist_digits):
            """Reshapes normal mnist digit (:,28*28) to matrix (:,28,28)"""
            return np.reshape(mnist_digits,(-1,28,28))

        def invert_grayscale(mnist_digits):
            """ Makes black white, and white black """
            return 1-mnist_digits

        to_visualise = batch_xs
        to_visualise = vector_to_matrix_mnist(to_visualise)
        to_visualise = invert_grayscale(to_visualise)

        sprite_image = create_sprite_image(to_visualise)

        plt.imsave(path_for_mnist_sprites,sprite_image,cmap='gray')
        plt.imshow(sprite_image,cmap='gray')


        with open(path_for_mnist_metadata,'w') as f:
            f.write("Index\tLabel\n")
            for index,label in enumerate(batch_ys):
                f.write("%d\t%d\n" % (index,label))
```

Now we can go the console and start tensorboard. This is done by specifying tensorboard the directory where the visualization files are located. In linux/ubuntu systems the following command is used

```
$ tensorboard --logdir=MNIST_tensorflow/mds_sample
```