Lab Exercise 5: Deep Learning
Data Science

Due date: 10 December

1 Introduction

In the fifth and final lab session, the aim is to implement more advanced algorithms for deep learning. Concretely, we will apply a convolutional neural network to a classification task, and an autoencoder in order to derive a set of features in an unsupervised manner.

After completing the tasks, you should write a report in PDF that includes the results of all tasks. To submit your solution, create a compressed file that includes your source code and the report, and upload the compressed file to the Aula Global.

2 Convolutional neural network

Download the Flowers dataset: `http://download.tensorflow.org/example_images/flower_photos.tgz`. This dataset contains 3,670 images of 5 different flower species. Uncompress the dataset in a target folder.

Tasks:

1. Train a convolutional neural network with a softmax layer in order to predict the type of flower.
2. Experiment with different numbers and sizes of convolutional layers to reduce the prediction error.

Help with Matlab:

- To load the dataset and convert images to resolution 64×64:

```
imds = imageDatastore('flower_photos','IncludeSubfolders',true,'LabelSource','foldernames');
augmenter = imageDataAugmenter('RandXReflection',true);
augimds = augmentedImageDatastore([64 64],imds,'DataAugmentation',augmenter);
```

- To split the dataset into a training set and a validation set, create random arrays of indices in the range $[1, 3670]$ and use the partitionByIndex function:

```
imdsTrain = partitionByIndex(augimds,trainIdx);
```

- To create a convolutional neural network, manually define an array of network layers:

```
layers = [
    imageInputLayer([64 64 3])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    ...

    fullyConnectedLayer(5)
    softmaxLayer
    classificationLayer
];
```

- Manually define options for training the network in an array `options`:
`https://www.mathworks.com/help/deeplearning/ref/trainingoptions.html`
It is a good idea to include the validation set using the option 'ValidationData'.

- Finally, to train the network:

```
net = trainNetwork(imdsTrain,layers,options);
```

3 Autoencoder

For the following tasks we will use the dataset for digit recognition in images that is built into the neural network toolbox for Matlab. The aim is to combine unsupervised and supervised learning in order to train a classifier for the digit recognition dataset.

Tasks:

1. Train an *autoencoder* on a set of input images in order to automatically extract features that are good predictors of digits in images.
2. Train a second autoencoder whose input is the output of the first autoencoder.
3. Train a softmax layer whose input is the output of the second autoencoder, and whose output is a predictor of the class (i.e. digit).
4. Construct a deep neural network such that the first hidden layer is the first autoencoder, the second hidden layer is the second autoencoder, and the output layer is given by the softmax layer. Report the test performance of the resulting deep network.
5. Retrain the deep network in order to fine-tune the weights, and report the test performance after retraining the network.

Help with Matlab:

- To access the data set simply run the following command:

  ```
  [XTrainImages yTrain] = digittrain_dataset;
  [XTestImages yTest] = digittest_dataset;
  ```

- The following code trains an autoencoder on the training images:

  ```
  autoenc = trainAutoencoder( XTrainImages, hidden, 'L2WeightRegularization', 0.004, ...
      'SparsityRegularization', 4, 'SparsityProportion', 0.15, 'ScaleData', false );
  ```

  The argument `hidden` defines the number of hidden nodes.

- To visualize the reconstructed images as well as the hidden features:

  ```
  xRecon = predict( autoenc, XTrainImages );
  for i = 1:20
      subplot( 4, 5, i );
      imshow( xRecon{i} );
  end
  plotWeights( autoenc )
  ```

- The output of an autoencoder is given by the function `encode`:

  ```
  features = encode( autoenc, XTrainImages );
  ```

- To train a softmax layer we can use the function `trainSoftmaxLayer`:

  ```
  softnet = trainSoftmaxLayer( someFeatures, yTrain, 'MaxEpochs', 400 );
  ```

- To construct a deep network from a series of existing layers we can use the function `stack`:

  ```
  deepnet = stack( autoenc, autoenc2, softnet );
  ```

- To pass inputs to the deep network we first have to preprocess the images:

  ```
  XTest = zeros( 28*28, numel( XTestImages ) );
  for i = 1:numel( XTestImages )
      XTest(:,i) = XTestImages{i}(:);
  end
  yPredict = deepnet( XTest );
  performance = perform( deepnet, yTest, yPredict )
  ```

- To retrain the deep network you can simply use the function `train`:

  ```
  deepnet = train( deepnet, XTrain, yTrain );
  ```

  The matrix `XTrain` has to be constructed in the same way as `XTest`.