Assignment Two: Image classification, time series prediction, sentiment classification

CS909: 2018-2019 (MSc Only)

Submission: 12 pm (midday) Wednesday 20th March 2019

Notes:

(a) Submit a zip file of the completed iPython notebooks to Tabula. Make sure to put comments in your code explaining what you have done and what you have observed from the results.

(b) This assignment will contribute to 35% of your overall mark.

This assignment consists of 3 parts. You can find each part in the labs of **CNN for Image Classification**, **RNN for Time Series Prediction**, and **Sentiment classification**, respectively**.** Please download the corresponding ipynb files from the module web page https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs909.

1. Image classification (35 marks)

In the first task, you are given a dataset containing 6,000 pictures of cats and dogs (3,000 cats, 3,000 dogs) and asked to train a classifier built upon **Convolutional Neural Networks** (ConvNets) to classify images as "*dogs*" or "*cats*".

You need to perform the following key steps:

a) Split the dataset by selecting 4,800 pictures for training, 600 for validation, and 600 for testing. (2 marks)

b) Train a Convolutional Neural Network (ConvNet) on the training set. The general structure of the ConvNet will be a stack of alternated Conv2D (with relu activation) and MaxPooling2D layers. A Conv2D layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. A MaxPooling2D layer is used to downscale input in both the vertical and horizontal dimensions. (10 Marks)

c) Output the training and validation loss curves and accuracy curves. Also output the classification results (e.g., classification accuracy, confusion matrix, precision-recall curve and/or ROC curve) on the test set. (5 marks)

d) Explore different network architectures (e.g., stacking 4 Conv2D+MaxPooling2D layers) and various ways in tuning the model parameters to see if you can improve the model performance on the validation set. (10 marks)

e) Apply the trained model on the testing set and output the classification results. (3 marks)

f) Plot the saliency map of original image to see which part is important for making classification decisions. You can refer to the following blog article on how to generate visualisation results of the filters in the ConvNets. (5 marks)

https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

2. Time series prediction (35 marks)

This is a time series prediction task. You are given a dataset which reports on the weather and the level of pollution each hour for five years, and asked to train **Recurrent Neural Networks (RNNs)** to predict the hourly pollution level.

You need to perform the following key steps:

a) Load the data from the file. Perform necessary pre-processing (e.g., missing value replacement, uninformative attribute removal, etc.) and visualise the values of various attributes over the five-year period. (5 marks)

b) Frame the task as the supervised learning problem as predicting the pollution at the current hour given the pollution measurement and weather conditions at the previous hour. Using the first 4 years' data as the training set and the remaining 1 year's data as the test set. Prepare the training/test sets accordingly. (10 marks)

c) Train a Recurrent Neural Network (RNN) on the training set. You can split the training set further by using 10% of the data as the validation set and the remaining for training. (5 marks)

d) Output the prediction results such as Root Mean Squared Errors (RMSE) on the test set. Remember that after the forecasts have been made, we need to invert the transforms to return the values back into the original scale. This is needed so that we can calculate error scores. Plot the predicted values vs. the actual values. (5 marks)

e) Explore different network architectures (e.g., stacked LSTM layers) and various ways in tuning the model parameters to see if you can improve the model performance on the test set. (5 marks)

f) Explore alternative prediction setup by predicting the pollution for the next hour based on the weather conditions and pollution over the last 3 days. (5 marks)

3. Sentiment classification (MSc ONLY) (30 marks)

In this task, you will process text data and train neural networks with limited input text data using pre-trained embeddings for sentiment classification (classifying a review document as "*positive*" or "*negative*" based solely on the text content of the review).

You need to perform the following key steps:

a) Download the movie review data as raw text.

First, create a "`data`" directory, then head to http://ai.stanford.edu/~amaas/data/sentiment/ and download the raw movie review dataset (if the URL isn't working anymore, just Google "IMDB dataset"). Save it into the "`data`" directory. Uncompress it. Store the individual reviews into a list of strings, one string per review, and also collect the review labels (positive / negative) into a separate labels list.

b) Pre-process the review documents. (2 marks)

Pre-process review documents by tokenisation and split the data into the training and testing sets. You can restrict the training data to the first 1,000 reviews and only consider the top 5,000 words in the dataset. You can also cut reviews after 100 words (that is, each review contains a maximum of 100 words).

c) Download the GloVe word embeddings and map each word in the dataset into its pre-trained GloVe word embedding. (3 marks)

First go to https://nlp.stanford.edu/projects/glove/ and download the pre-trained embeddings from 2014 English Wikipedia into the "`data`" directory. It's a 822MB zip file named `glove.6B.zip`, containing 100-dimensional embedding vectors for 400,000 words (or non-word tokens). Un-zip it. Parse the un-zipped file (it's a txt file) to build an index mapping words (as strings) to their vector representation (as number vectors).

Build an embedding matrix that will be loaded into an Embedding layer later. It must be a matrix of shape `(max_words, embedding_dim)`, where each entry *i* contains the `embedding_dim`-dimensional vector for the word of index *i* in our reference word index (built during tokenization). Note that the index 0 is not supposed to stand for any word or token -- it's a placeholder.

d) Build and train a simple Sequential model. (10 marks)

Define a model which contains an Embedding Layer with maximum number of tokens to be 10,000 and embedding dimensionality as 100. Initialise the Embedding Layer with the pre-trained GloVe word vectors. Set the maximum length of each review to 100. Flatten the 3D embedding output to 2D and add a Dense Layer which is the classifier. Train the model with a '`rmsprop`' optimiser. You need to freeze the embedding layer by setting its `trainable` attribute to `False` so that its weights will not be updated during training.

e) Plot the training and validation loss and accuracies and evaluate the trained model on the test set. (5 marks)

f) Add an LSTM layer into the simple neural network architecture and re-train the model on the training set, plot the training and validation loss/accuracies, also evaluate the trained model on the test set and report the results. (10 marks)