1 Introduction
In this assignment, we will implement the same linear classifier as in Assignment 2 using TensorFlow, one of the mainstream deep learning libraries. We will reuse a substantial amount of the code of Assignment 2. Similarly, your task is to implement the missing components so that the linear classifier is fully functional on the sentiment classification task.

2 Preparation
Let us start by creating the working directory and download the skeleton code for the assignment via:

```
cd /your/working/directory
cp /local/kurs/ml/Assignment3.zip ./
unzip Assignment3.zip
cd Assignment3
```

Set up the virtual python environment by:

```
source /local/kurs/ml/2018/virtualenv/bin/activate
```

Open `classifier.py` in your favourite python script editor (pyCharm, Vim, Emacs, etc) and read it through. The general structure of the script is very similar to `classifier.py` in Assignment 2. We start by loading the data and split them into three different data sets for training, validation and final testing respectively. In the previous assignment, we use sparse representations for features. In this assignment, we convert them into dense representations so that we can apply numerical computations with TensorFlow. The shape of the dense representations for the features is `[num_instances, num_features]`.

Now, find the computational graph in the script and look through how it is defined. Your main task in this assignment is to complete the missing components of the computational graph.

3 Define Placeholders
In TensorFlow, we feed the data into the computational graph through `tf.placeholder()`. Read the instructions on `tf.placeholder()` on the TensorFlow website (https://www.tensorflow.org/) and find out how it works. For the sentiment classification task, we need two placeholders respectively for the features and corresponding labels to train the classifier. In the skeleton code, the placeholder for features is already defined for you. Note that pre-defining the data type and the shape for placeholder is mandatory while the name is optional. The shape of the feature placeholder is `[num_instances, nfeatures]`. None means that the corresponding dimension is flexible with respect to the input data. The dimension associated with None usually corresponds to the number of data instances.

Now, define the placeholder for labels using `tf.placeholder()` similarly to the feature placeholder. Please note that the label placeholder has a different shape.

4 Define Variables
The model weights are defined and applied as variables in TensorFlow with `tf.get_variable()`. The weight variable is already defined in the code. Similarly, we need to pre-define the variable shape if `tf.get_variable()` is used for creating new variables. In this assignment, we also use `tf.zeros_initializer()` as the initialiser instead of the default one.

Define the bias variable using `tf.get_variable()`. Use `tf.zeros_initializer()` as the initialiser as well.

5 Dot product
The dot product $\tilde{W} \times x$ of the features $x$ and the weights $\tilde{W}$ can be computed using python list as:
prod = 0
mult = [wi * xi for wi, xi in zip(W, x)]
for m in mult:
    prod += m
return prod

This can be computed very efficiently by `tf.matmul(x, W)` using TensorFlow, in which `x` and `W` should have the same rank and be compatible. In our code, we have features in the shape of `[None, nfeatures]` and weights of `[nfeatures]` and therefore we expand weights into `exp_weights` in `[nfeatures, 1]` using `tf.reshape()`. Now, apply `tf.matmul()` in the code with `features` and `exp_weights`. In addition, add bias defined in the previous section to `logits`, the product of `tf.matmul()`.

6 Run the Script

We can see that the loss function and the regulariser are also defined in the computational graph. Similarly to the previous assignment, we compute the gradients and apply gradient decent for training. This can be easily implemented with one line of code in TensorFlow. Find out how to use `tf.train.GradientDescentOptimizer` and add it to the code.

If all the previous steps are done correctly, you should be able to run the script by:

```
python classifier.py
```

which gives you reasonable output that is comparable to Assignment 2. Report the top 10 positive and negative features and validation accuracy.

7 Loss Functions

We use logistic loss `logistic_loss()` as our loss function. There are a number of alternative loss functions that are available in TensorFlow under `tf.losses`. Try to use `hinge loss` and `mean square error` and see how they affect the results. Report your observations.

8 Regularisers

We use `l1_regularizer` by default in our code, which is defined with `tf.contrib.layers.l1_regularizer(reg_lambda)`. `l2_regularizer` is implemented similarly under `tf.contrib.layers`. Figure out how it works and apply it in the code. Does it work differently from `l1_regularizer`? Report your observations.

9 Comparison with the Naive Python Classifier

Change the hyper-parameters of the classifier into the best parameter setting that you had for the naive python classifier in the previous assignment. Does it yield similar results? Test 5 other different settings for the two classifiers respectively and report relevant information. How about their training speeds? Interpret the differences to the best of your knowledge.

10 Mini-Batch Online Learning (for VG)

The default linear classifier is based on batch learning, in which we use all the training instances to compute the gradients for weights updating. Now, implement the perceptron-style mini-batch online learning algorithm with TensorFlow. Instead of using all the training instances for each weight update, feed them into the model by mini-batches. You need to shuffle the training data before each training iteration. Report your result and compare with the default linear classifier.

Submission

You should submit a short written report (3-4 pages) together with your modified `classifier.py`. Try to solve all the problems and answer the questions in your report. Please upload all the required materials to Studentportalen before May 18, 2018.