Labs
Machine Learning Course
Fall 2016

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-136795.html
epfmlcourse@gmail.com

Problem Set 8, May 5-6, 2016
(Neural Networks (see CIL ETH))

Problem 1 (Getting Started with TensorFlow):

Set up TensorFlow on your system, by following

www.tensorflow.org/get_started/index.html.

Get the two provided code templates `ex1.py` and `ex2.py` from

github.com/dalab/lecture_cil_public/tree/master/exercises/ex8.

We implement a multi-valued linear regression model, where we are supposed to find the weights $\boldsymbol{W} \in \mathbb{R}^{d \times k}$ and $\boldsymbol{b} \in \mathbb{R}^k$ in

$$\frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{X}_{i:} \boldsymbol{W} + \boldsymbol{b} - \boldsymbol{Y}_{i:})^2$$

for a set of n data examples given as the rows of the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, and the target observations given as the corresponding rows of $\boldsymbol{Y} \in \mathbb{R}^{n \times k}$.

We would like to find the optimal $\boldsymbol{W}$ and $\boldsymbol{b}$ by means of gradient descent. Tensorflow (as well as Torch, Theano, etc.) makes this very easy because of the built-in *autodifferentiation*. This feature allows differentiating arbitrary computation graphs. For simple (stochastic) gradient descent, TensorFlow even hides this differentiation by means of an *optimizer*. The optimizer will use the autodifferentiation to obtain the gradient of the loss with respect to $\boldsymbol{W}$ and $\boldsymbol{b}$ and then make a descent step each time its computation is triggered.

The workflow when using systems like TensorFlow is thus as follows:

1. Specify your model using variables (parameters) and placeholders (data).
2. Specify your loss (usually some form of distance between the model output and the desired output).
3. Use a built-in optimization algorithm to find parameters that minimize the loss.

For our example of multivariate linear regression, do the following tasks:

1. Fill in the model and loss function in `ex1.py`, and visualize the progress in the loss function during each step of gradient descent for the weight variables $\boldsymbol{W}$ and $\boldsymbol{b}$, by inspecting the tensorboard in the browser. We want to do full gradient descent, so just use the (numpy) dataset variables for $\boldsymbol{X}$ and $\boldsymbol{Y}$.
2. Fill in the model and loss function in `ex2.py`, and visualize the progress in the loss function during each step of stochastic gradient descent for the weight variables $\boldsymbol{W}$ and $\boldsymbol{b}$, by inspecting the tensorboard in the browser. Note that now we want to *stochastic* gradient descent, meaning we don't want to use the whole dataset in each iteration, but just a subset of it. Tensorflow does this by replacing the data variables with *placeholders*, which are then successively fed with a subset of the data by means of the *feed dictionary* parameter.
3. What difference do you see in the plots between full and stochastic gradient descent? What are advantages and disadvantages of each? Play with the step sizes, minibatch sizes and dataset noise settings. Can you find an instance where one method works, but the other one does not?

Problem 2 (Convolutional Neural Networks):

Go through the MNIST digit recognition tutorial on

www.tensorflow.org/tutorials/mnist/beginners/index.html.

Problem 3 (Road Extraction from Satellite Images):

For the third choice of semester project task, we provide a set of satellite/aerial images acquired from GoogleMaps. We also provide ground-truth images where each pixel is labeled as {road, background}. Your goal is to train a classifier to segment roads in these images, i.e. assign a label {road=1, background=0} to each pixel.

1. Download the training data from the competition website

   inclass.kaggle.com/c/cil-road-segmentation.

2. Obtain the python notebook `segment_aerial_images.ipynb` from

   github.com/dalab/lecture_cil_public/tree/master/exercises/ex8

   to see example code on how to extract the images as well as pixel labels.

   The notebook shows how to use `scikit learn` to generate features from each pixel, and finally train a linear classifier to predict whether each pixel is road or background. It also provides helper functions to visualize the images, labels and predictions.

3. As a more advanced approach, try `tf_aerial_images.py`, which demonstrates the use of a convolutional neural network in TensorFlow for the same prediction task.