Machine Learning Practical

Part 2

MLP Part 2

Steve Renals

Machine Learning Practical — MLP Lecture 10
18 January 2017
http://www.inf.ed.ac.uk/teaching/courses/mlp/

MLP in Semester 2

- **Lectures**
  - One introductory lecture (today)
  - Four guest lectures (weeks 2-5: 25 January / 1 February / 8 February / 15 February)
- **Labs**
  - Weekly labs - six sessions, choose one at `http://doodle.com/poll/dy78erdey4twhk7c`
  - Same place (Forrest Hill 3.D01), some times changed
- **Content**
  - Moving away from MNIST, work on *one* of two datasets in image recognition (CIFAR-10/100) or music (Million Song Database).
  - Moving away from the MLP software framework to TensorFlow (`https://www.tensorflow.org`)

Using Different Data

- Although MNIST is an interesting, standard data set it has limitations
    - Constrained task, results in high accuracy classifiers (quite possible to get 99% with standard approaches)
    - So model/algorithm advances may result in only very small changes
    - (But even in 2016 major conferences like NIPS still had plenty of papers presenting results on MNIST)
- What are we looking for in a dataset for part two of MLP
    - Large enough to be interesting, small enough to be computationally feasible for the course
    - Not necessarily image recogniton.... definitely not digit recognition
    - Clear task, ideally with published results

Choose one!

- CIFAR-10/100 (object recognition in images) – https://www.cs.toronto.edu/~kriz/cifar.html (much harder task than MNIST)
- Million song dataset (audio features and metadata for a million contemporary popular music tracks) – http://labrosa.ee.columbia.edu/millionsong/

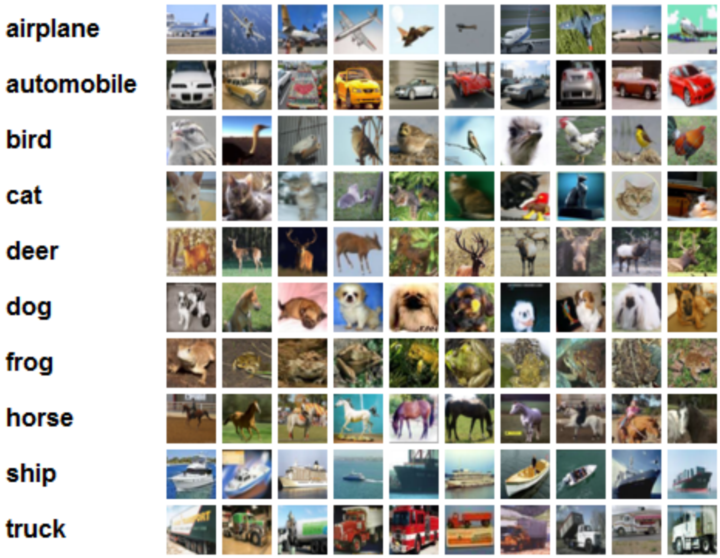
(Note: you don't need to immediately start downloading these datasets, we'll provide versions that you can use, will be explained in lab 9)

CIFAR-10

https://www.cs.toronto.edu/~kriz/cifar.html

- CIFAR-10 is a very well-studied dataset for object recognition in images.
- The data
  - 60,000 colour images (32x32 pixels)
  - 10 object classes – 6,000 examples of each class (5,000 training, 1,000 test)
  - Collected at University of Toronto as a labelled subset of the MIT 80 million tiny images dataset (http://people.csail.mit.edu/torralba/tinyimages/)
- CIFAR-100 – similar to CIFAR-10, but 100 classes and 600 examples per class (500 training, 100 test)
- Many results on CIFAR-10 and CIFAR-100 collected at http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html
  - CIFAR-10 – approx up to 95% correct
  - CIFAR-100 – approax up to 75% correct

CIFAR-10


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

CIFAR-10 vs MNIST

- Similarities:
  - spatial data
  - small images
  - 10-class classification
  - 50k training images
- Differences:
  - **much greater diversity across images of e.g. ’truck’ compared with e.g. ’9’**
  - colour images

Million Song Dataset

http://labrosa.ee.columbia.edu/millionsong/

- Contains precomputed features and metadata (labels) for a million songs
- Several possible tasks, using subsets of the data – we will focus on genre classification of a subset of the data
- Acoustic features:
    - Each song represented by a sequence of segments – roughly each segment corresponds to a note, variable number of segments in a song
    - Each segment represented by a fixed size feature vector containing information relating to pitch (chroma), timbre (MFCCs), loudness, duration.
- The specific task and test/training sets are much less standard compared to CIFAR-10, wide range of reported results

Song classification

- Requires temporal modelling – can train a network to label each segment, but the task requires a label per song (not per segment)
- Possible approaches
  - Transform variable length sequence of segments into a fixed-dimension "supervector", and then classify
  - Classify each segment then use some kind of temporal model to generate a song-level classification
  - Train a network which maps an input sequence (variable length) to a single classification (e.g. recurrent network)

TensorFlow

https://www.tensorflow.org

Overview

TensorFlow is an open source software library for numerical computation using data flow graphs.

Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

Flexible, portable, efficient
Automatic differentiation
Python interface

Basics of TensorFlow

https://www.tensorflow.org/get_started/basic_usage

- TensorFlow represents computations as graphs where the nodes are mathematical operations (add, multiply, ...) and the edges are data (multidimension arrays or *tensors*)

Build a graph that multiplies a 1x2 matrix with a 2x1 matrix:

```
import tensorflow as tf
matrix1 = tf.constant([[3., 3.]])
matrix2 = tf.constant([[2.],[2.]])
product = tf.matmul(matrix1, matrix2)
```

Run the graph in a **session**

```
with tf.Session() as sess:
  result = sess.run(product)
```

Other basic concepts

- **Variables** – tensors which maintain state across executions of a graph (e.g. weights and biases of a network)
- **Placeholders** – rather than specifying tensors directly, allow them to *feed* into the graph as an argument to the run call (e.g. input data for a network)

Training a single layer network for MNIST – example in lab 8

Build a single-layer network for MNIST (lab 8)

```
inputs = tf.placeholder(tf.float32, [None, 784], 'inputs')
targets = tf.placeholder(tf.float32, [None, 10], 'targets')

weights = tf.Variable(tf.zeros([784, 10]))
biases = tf.Variable(tf.zeros([10]))

outputs = tf.matmul(inputs, weights) + biases

per_datapoint_errors = tf.nn.softmax_cross_entropy_with_logits(
                            outputs, targets)
error = tf.reduce_mean(per_datapoint_errors)

train_step = tf.train.GradientDescentOptimizer(learning_rate=0.5).
                            minimize(error)

train_data = data_providers.MNISTDataProvider('train', batch_size=50)
valid_data = data_providers.MNISTDataProvider('valid', batch_size=50)
```

Run a single-layer network for MNIST (lab 8)

```
sess = tf.InteractiveSession()
init_op = tf.global_variables_initializer()
sess.run(init_op)

num_epoch = 5
for e in range(num_epoch):
    running_error = 0.
    for input_batch, target_batch in train_data:
        _, batch_error = sess.run(
            [train_step, error],
            feed_dict={inputs: input_batch, targets: target_batch})
        running_error += batch_error
    running_error /= train_data.num_batches
```

Labs and coursework

- After the initial lab on Introduction to TensorFlow, you will focus on a mini-project, either
  - CIFAR-10/100
  - Million Song DataSet
- **Phase 1:** develop baseline systems (typically DNN systems), report results, develop plan for phase 2. Submit a report. (weeks 3–5)
- **Phase 2:** conduct more advanced experiments (e.g. RNN architectures, CNN architectures, pretrainining, data augmentation, ...). Report results and conclusions. Submit a report. (weeks 6–8)

Semester plan

- **Week 2:** Intorduction to TensorFlow (lab 8)
- **Week 3:** Start work on *either* CIFAR-10 *or* Million Song Dataset (there will be a lab to get you started for each released end of next week)
- **Week 5 (16 February):** Handin initial report including baseline results and plan for future work. ("coursework 3")
- **Week 8 (16 March):** Handin final report, with methods, results, discussion, and conclusions ("coursework 4")