Eindhoven Data Science Meetup

31 okt 2017: Recurrent Neural Networks

Agenda

- 18:00 - Walk in
- 18:15 - presentation on RNN
  - Recap Tensorflow
  - Recap Neural Networks in Tensorflow
  - Recap Convolutional Neural Networks in Tensorflow
  - Theory Recurrent Neural Networks
  - Different types of RNN
  - The Dataset of tonight
- 18:45 - Start hacking on assignments and your project
- 20:45 - Wrapup
- 21:00-21:30 drinks

Past Deep Learning meetups:

May 2017: Introduction to Tensorflow and Neural Networks

June 2017: Convolutional Neural Networks in Tensorflow

…

Oktober 2017: Recurrent Neural Networks in Tensorflow

…

December 2017/Jan 2018: Text Analytics with RNN

Recap Tensorflow (1)

Basic building block of Tensorflow:

- **tf.Variable**; a variable;
  - used for weight matrices, bias vectors, etc
- **tf.Constant** ; a variable whose value cannot be changed
  - test-set, learning rate
- **tf.Placeholder**; a way to allocate memory, without specifying the values
- **Graphs** ; all computational steps for a processes are defined within one graph.
- **Session**; A graph can be executed with a Session.

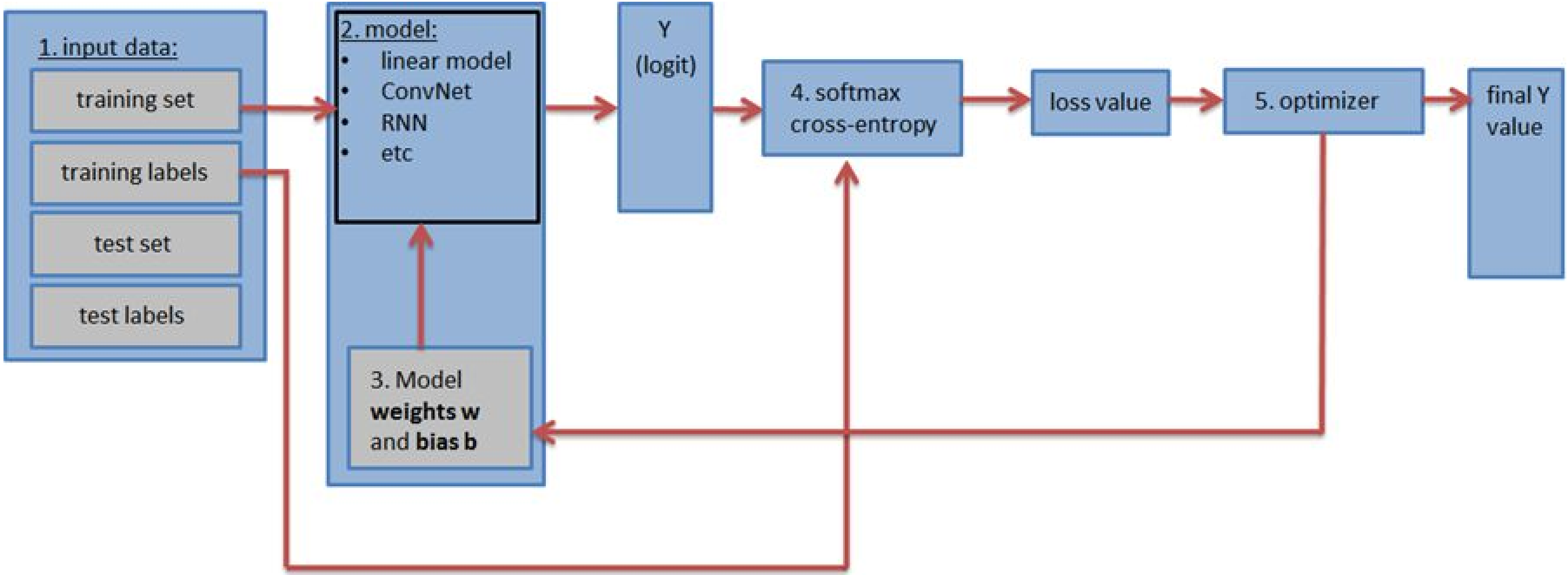
Recap Tensorflow (2)

```
graph = tf.Graph()
with graph.as_default():
        tf_train_dataset = tf.placeholder(tf.float32, shape=(..,..,..,..))
        tf_test_dataset = tf.constant(test_dataset, tf.float32)

        bias = tf.Variable(0, tf.float32)
        weight_matrix = tf.Variable(tf.zeros([2,2], tf.float32))
        (...)


with tf.Session(graph=graph) as session:
        tf.global_variables_initializer().run()
        (…)
```

Recap Tensorflow (3)

```
list_of_points1 = [[1,2], [3,4], [5,6], [7,8]]
list_of_points2 = [[15,16], [13,14], [11,12], [9,10]]

graph = tf.Graph()
with graph.as_default():
        point1 = tf.placeholder(tf.float32, shape=(1, 2))
        point2 = tf.placeholder(tf.float32, shape=(1, 2))

        def calculate_eucledian_distance(point1, point2):
             (…)
             return eucledian_distance

        dist = calculate_eucledian_distance(point1, point2)

with tf.Session(graph=graph) as session:
        tf.global_variables_initializer().run()
        for ii in range(len(list_of_points1)):
            point1_current = list_of_points1[ii]
            point2_current = list_of_points2[ii]

            feed_dict = {point1 : point1_current, point2 : point2_current}
            distance = session.run([dist], feed_dict=feed_dict)
            print("the distance between {} and {} -> {}".format(point1_, point2_, distance))

>>> the distance between [[1 2]] and [[15 16]] -> [19.79899]
```

Neural Networks in Tensorflow

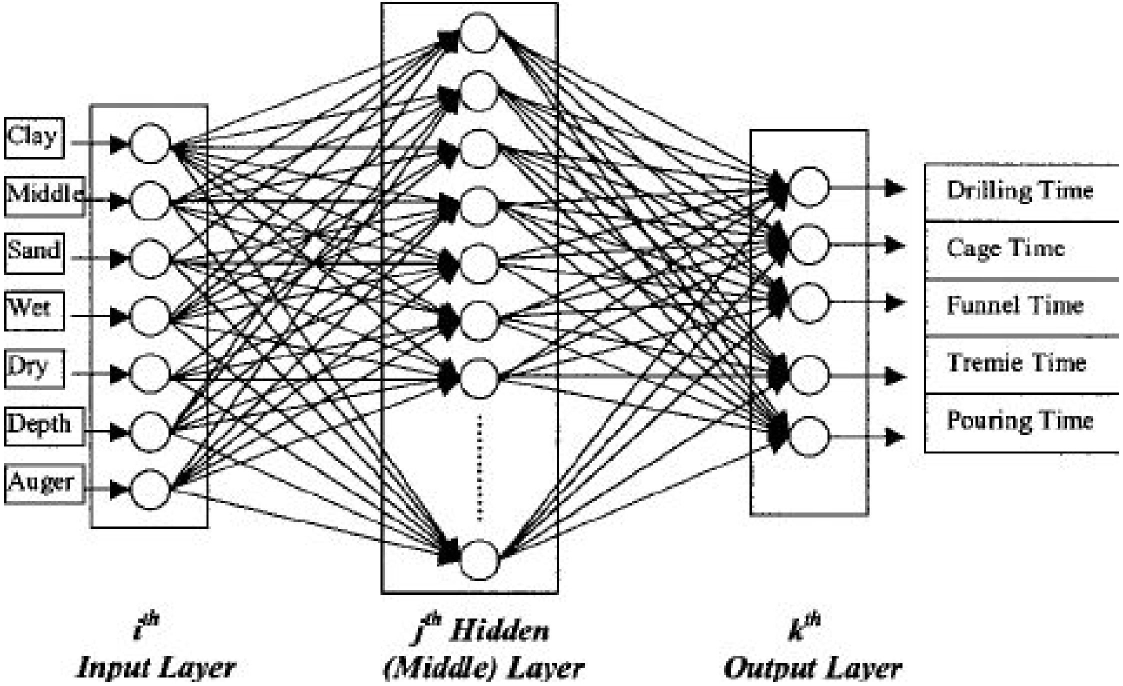


Neural Networks in Tensorflow (3)

The ‘model’ can be a:

- Fully Connected Neural Network
- Convolutional Neural Network
  - Deep ConvNet: Alexet, VGGNet-16, ResNet, etc
- Recurrent Neural Network;
  - static, dynamic, bidirectional, multilayer RNN

Fully Connected Neural Network



See ‘3Blue1Brown’ youtube channel: https://www.youtube.com/watch?v=aircAruvnKk&t=1025s

Convolutional Neural Networks

Neurons correspond with pixels in image.

We can therefore take into account the essential characteristics of images:

- There is a spatial relationship between the pixels
- Local connectivity between the pixels
  - A neuron is only connected to the neurons corresponding to pixels in the immediate surrounding and not to all other neurons. This greatly reduces the number of connections.

A convolutional layer contains filters/kernels and produces a smaller but deeper feature map. These filters can be for edge detection, sharpening, blurring, some color palette etc.

Convolutional Neural Networks (2)

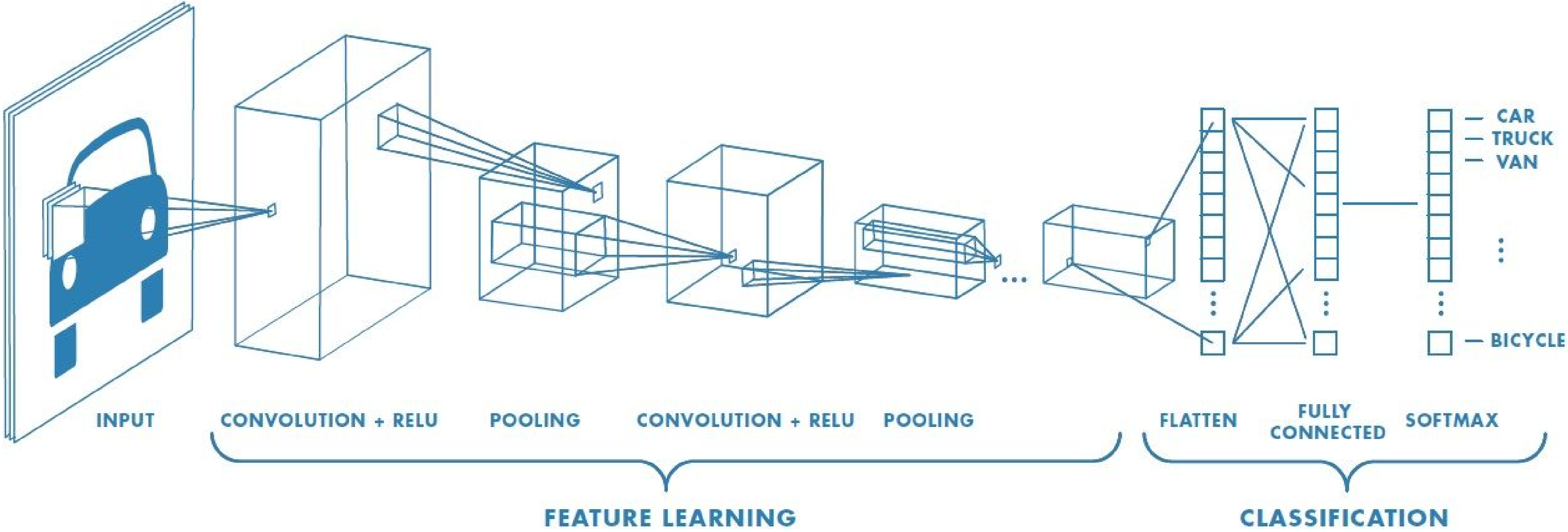


figure : https://www.mathworks.com/discovery/convolutional-neural-network.html

From CNN to RNN

- What are the main differences between CNN and RNN?

From CNN to RNN

- CNN’s work well with data with spatial features.
- RNN’s work well with data with sequential features.
  - Time series
  - Signals
  - Text documents

Main differences:

- Input data can have variable length
- RNN consider temporal dependencies in the data

1 CNN

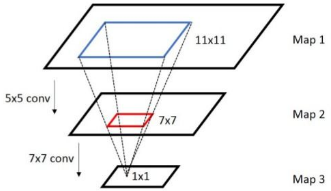


Figure: credits: cvmarcher.com/

2 RNN

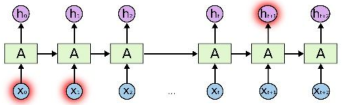
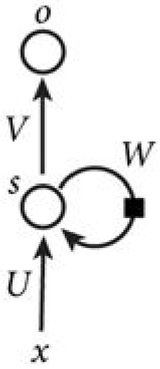


Figure: credits colah.github.io

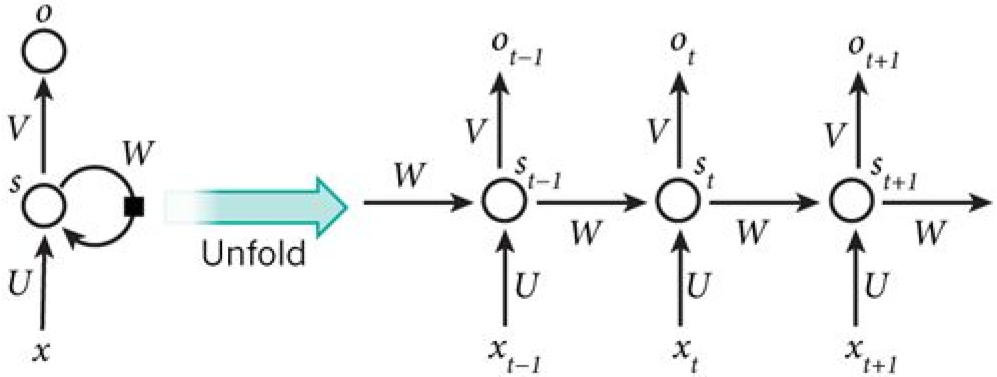
Recurrent Neural Networks

A RNN layer, **recurs** (state) information to itself.



Recurrent Neural Networks

A RNN layer, **recurs** (state) information to itself.



Input at some timestep x_t also depends on value of previous timesteps x_{t-1}, x_{t-2}, etc

Each node can keep information which is useful and ‘forget’ the rest.

In this way it learns **Long Term and Short Term** temporal dependencies (**LSTM RNN**)

Figure: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Recurrent Neural Networks (2)

Weights are shared across different time-steps. Cost values are different for each time-step.

Backpropagation through time (BPTT)

Since cost values are calculated with the weights, and weights are shared across diff time steps, the chain rule is used to backpropagate the gradient all the way up to t0.

This leads to the vanishing / exploding gradient problem.

- Gradient clipping
- Truncated backpropagation

Recurrent Neural Networks

Different Types of (LSTM) Recurrent Neural Networks:

- (LSTM) Recurrent Neural Network
- Bidirectional (LSTM) Recurrent Neural Network
- Two Layered (LSTM) Recurrent Neural Network
- Multi Layered (LSTM) Recurrent Neural Network

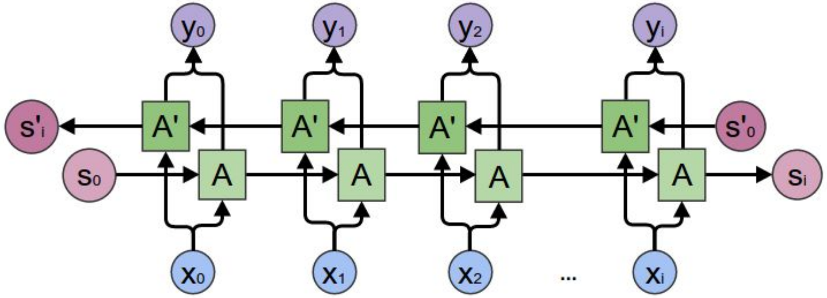


Figure: http://colah.github.io/posts/2015-09-NN-Types-FP/

The Dataset

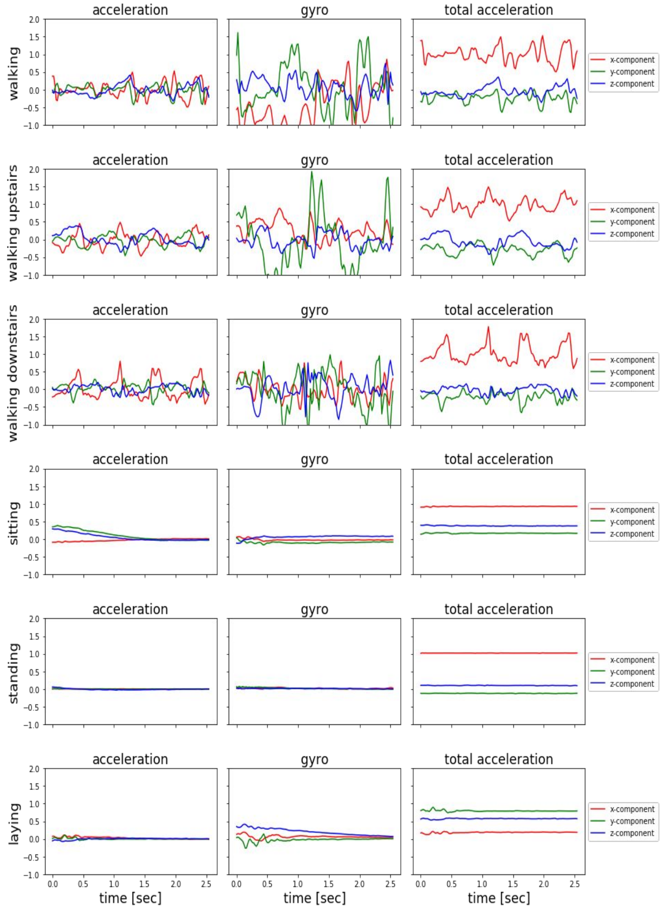
Human Activity Recognition

Each signal has 9 components:

- Triaxial acceleration from the accelerometer: body acceleration.
- Triaxial acceleration from the accelerometer: total acceleration
- Triaxial Angular velocity from the gyroscope.

Each signal can be one of six activities:

1. walking
2. walking upstairs
3. walking downstairs
4. sitting
5. standing
6. laying



The Assignment

The data is already loaded into a training and a test set.

Tf Graph with all computational steps (except for the RNN-model) is provided.

Who can classify the test set with highest accuracy?

Start simple, after you have build a simple but working RNN, you can play around with different learning rates, optimizers, deeper RNN’s, etc.

Start From: https://tinyurl.com/y7zmsx5v