Exercise 3

Deadline: 01.06.2016

Note: There will be no lecture & exercise group on the 27.05

In this exercise we want to introduce the theano framework that can be used to create neural networks in python. Its biggest strength is that you can define functional relations between symbolic variables (e.g. Tensors) and calculate the gradient of any nested expression. Theanos tutorials and documentation can be found at `http://deeplearning.net/tutorial/`.

1 Introduction(5 Point)

First you need to make yourself familiar with theano. The following code (also provided on the course website) defines a simple neural network with 2 hidden layers

```
1  import theano
2  from theano import tensor as T
3  from theano.sandbox.rng_mrg import MRG_RandomStreams as RandomStreams
4  import numpy as np
5  from load import mnist
6
7  srng = RandomStreams()
8
9  def floatX(X):
10     return np.asarray(X, dtype=theano.config.floatX)
11
12 def init_weights(shape):
13     return theano.shared(floatX(np.random.randn(*shape) * 0.01))
14
15 def rectify(X):
16     return T.maximum(X, 0.)
17
18 def init_slope(shape):
19     return theano.shared(floatX(0.25  + np.zeros(shape)))
20
21 def PRelu(X, slope):
22     return T.maximum(X, 0.) + a * T.minimum(X, 0)
23
24 def softmax(X):
25     e_x = T.exp(X - X.max(axis=1).dimshuffle(0, 'x'))
26     return e_x / e_x.sum(axis=1).dimshuffle(0, 'x')
27
28 def RMSprop(cost, params, lr=0.001, rho=0.9, epsilon=1e-6):
29     grads = T.grad(cost=cost, wrt=params)
30     updates = []
31     for p, g in zip(params, grads):
32         acc = theano.shared(p.get_value() * 0.)
33         acc_new = rho * acc + (1 - rho) * g ** 2
34         gradient_scaling = T.sqrt(acc_new + epsilon)
35         g = g / gradient_scaling
36         updates.append((acc, acc_new))
37         updates.append((p, p - lr * g))
38     return updates
39
40 def model(X, w_h, w_h2, w_o, p_use_input, p_use_hidden):
41     #X = dropout(X, p_drop_input)
42     h = rectify(T.dot(X, w_h))
43     #h = dropout(h, p_drop_hidden)
44     h2 = rectify(T.dot(h, w_h2))
45     #h2 = dropout(h2, p_drop_hidden)
46     py_x = softmax(T.dot(h2, w_o))
47     return h, h2, py_x
48
```

```
49 trX, teX, trY, teY = mnist(onehot=True)
50
51 X = T.fmatrix()
52 Y = T.fmatrix()
53
54 w_h = init_weights((784, 625))
55 w_h2 = init_weights((625, 625))
56 w_o = init_weights((625, 10))
57
58
59 noise_h, noise_h2, noise_py_x = model(X, w_h, w_h2, w_o, 0.8, 0.5)
60 h, h2, py_x = model(X, w_h, w_h2, w_o, 1., 1.)
61 y_x = T.argmax(py_x, axis=1)
62
63 cost = T.mean(T.nnet.categorical_crossentropy(noise_py_x, Y))
64 params = [w_h, w_h2, w_o]
65 updates = RMSprop(cost, params, lr=0.001)
66
67 train = theano.function(inputs=[X, Y], outputs=cost, updates=updates,
       allow_input_downcast=True)
68 predict = theano.function(inputs=[X], outputs=y_x, allow_input_downcast=True)
69
70 for i in range(100): #you can adjust this if training takes too long
71     for start, end in zip(range(0, len(trX), 128), range(128, len(trX), 128)):
72         cost = train(trX[start:end], trY[start:end])
73     print np.mean(np.argmax(teY, axis=1) == predict(teX))
```

Task: Use the download script(provided on the course website) to download the MNIST dataset for theano. Run the code and save the final test error. This will be the starting point for the rest of the rest of the exercise.

2 Dropout(5 Points)

We want to use dropout learning for out network. Therefore, implement the function

```
def dropout(X, p_use=1.)
```

that sets random elements of X to zero.

Dropout:

- **If** $p_{use} < 1$:
  For every element $x_i \in X$ draw Φ_i randomly from a binomial distribution with $p = p_{use}$. Then reassign
  $$x_i \to \begin{cases} \frac{x_i}{p_{use}} & \text{if } \Phi = 1 \\ 0 & \text{if } \Phi = 0 \end{cases}$$
  Note: Use the binomial function from
  ```
  from theano.sandbox.rng_mrg import MRG_RandomStreams
  ```
- **Else:**
  Return the unchanged X.

You can now enable the dropout functionality. To this end, remove the comments from the lines

```
X = dropout(X, p_use_input)
h = dropout(h, p_use_hidden)
h2 = dropout(h2, p_use_hidden)
```

and check that you code still runs. **Question:** Explain in a few sentences how the dropout method works and how it reduces overfitting. Why do we need to initialize two models in line 59 and 60? Compare the test error with the test error from Section 1

3 Parametric Relu(10 Points)

Instead of a simple rectify mapping (aka rectified linear unit(Relu)) we want to add a parametric Relu that maps every element x_i of the input X to

$$x_i \to \begin{cases} x_i & x_i > 0 \\ a_i x_i & x_i \leq 0 \end{cases}.$$

A detailed description can be found in the paper **Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification** (see `http://arxiv.org/pdf/1502.01852.pdf`). The crux of this method is the learnable weightvector a that needs to be adjusted during training. Define the function

```
def PRelu(X,a)
```

that creates a PRelu layer by mapping $X \to \text{PRelu}(X)$.

Incorporate the parameter a into the **params** list and make sure that it is adjusted during training. **Note:** You can use an expression like $x * (x > 0)$, which is equivalent to T.maximum(x,0). The gradient functional **T.grad** will be able to differentiate this automatically.

4 Convolutional layers(20 Points)

In this exercise we want to create a similar neural network to LeNet from Yann LeCun. LeNet was designed for handwritten and machine-printed character recognition. It relies on convolutional layers that transform the input image by convolution with multiple learnable filters. LeNet contains convolutional layers paired with sub sampling layers as displayed in Figure 1. The Subsampling is done by max pooling which reduces an area of the image to one pixel with the maximum value of the area. Both functions are already available in theano:

```
convolutional_layer = rectify(conv2d(previous_layer, weightvector,  border_mode=
    'full')) #use border_mode='full' only for the first layer
subsampleing_layer = max_pool_2d(convolutional_layer, (2, 2)) # reduces window 2
    x2 to 1 pixel
out_layer = dropout(subsample_layer, p_drop_input)
```

4.1 Create a Convolutional network

Now we can design our own convolutional neural network that classifies the handwritten numbers from MNIST.

Implementation task:

- Reshape the input image with:

  ```
  trX = trX.reshape(-1, 1, 28, 28) #trainings data
  teX = teX.reshape(-1, 1, 28, 28) #test data
  ```

- Replace the first hidden layer **h** with 3 convolutional layers (including subsampling and dropout)
- connect the convolutional layers to the vectorized layer **h2** by flattening the input with

  ```
  T.flatten(lastConvolutionalLayer_output, outdim=2)
  ```

- The shape of the weight parameter for **conv2d** determines the number of filters f, the number of input images pic_{in}, and the kernel size $k = (k_x, k_y)$. You can initialize the weights with

  ```
  init_weights((f, pic_in, k_x, k_y))
  ```

Make a neural network with

convolutional layer:	first	second	third
f	32	64	128
pic_{in}	1	32	64
k_x	5	5	2
k_y	5	5	2

and add the weightvectors to the **params** list.

- In Section 4.2 you will determine the number of output pixels of the CNN. Use it to adjust the size of the rectifier layer to

```
w_h2 = init_weights((number_of_output_pixel, 625))
```

- Use a softmax output layer with 625 inputs and 10 outputs (as before).

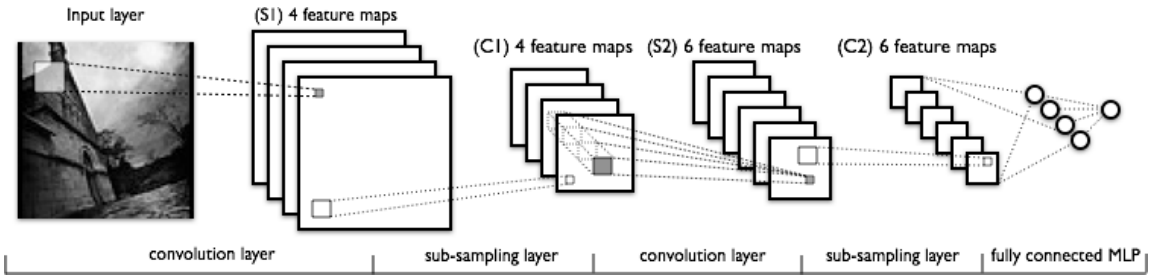

Abbildung 1: Sketch of convolutional neural network similar to LeNet

4.2 Application of Convolutional network

Task:

- draw a sketch of the network(like Figure 1) and note the sizes of the filter images (This will help you to determine how many pixels there are in the last convolution layer).
- after the training plot
  - one image from the test set
  - its convolution with 3 filters of the first convolutional layer
  - the corresponding filter weights(this should be 5 by 5 images).

Finally, choose one of the following tasks:

- add or remove one convolutional layer (you may adjust the number of filters)
- increase the filter size (you may plot some pictures i
- apply a random linear shift to the trainings images. Does this reduce overfitting?
- use unisotropic filters $k_x ! = k_y$
- create a network architecture of your choice and see if you can improve on the previous results

and compare the new test error.

Ideally you should create an overview table that lists the test errors from all sections.

5 Troubleshooting

- If you get the error: "*The image and the kernel must have the same type.inputs(float32), kerns(float64)*", you need to execute *export THEANO_FLAGS=floatX=32* in your command line.

- "*Library not loaded: /usr/local/opt/openssl/lib/libssl.1.0.0.dylib*". This results from a broken openssl installation on mac. It can be fixed by uninstalling and reinstalling openssl:
  sudo brew remove openssl
  brew install openssl

Regulations

Please hand in the python code, figures and explanations (describing clearly which belongs to which). Non-trivial sections of your code should be explained with short comments, and variables should have self-explanatory names. Plots should have informative axis labels, legends and captions. Please enclose all results into a single .pdf document and hand in the .py files that created the results. Please email the solutions to `mlhd1516@gmail.com` before the deadline. You may hand in the exercises in teams of maximally three people, which must be clearly named on the solution sheet (one email is sufficient). Discussions between different teams about the exercises are encouraged, but the code must not be copied verbatim (the same holds for any implementations which may be available on the WWW). Please respect particularly this rule, otherwise we cannot give you a passing grade. If you have 50% or more points in the end of the semester you will be allowed to take take part in the projects.