Ludwig-Maximilians-Universitaet Muenchen 13.05.2015
Institute for Informatics
Prof. Dr. Volker Tresp
Gregor Jossé
Johannes Niedermayer

Machine Learning and Data Mining

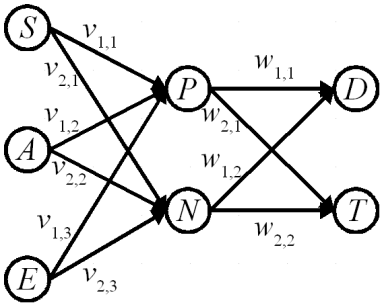
Summer 2015

Exercise Sheet 4

Presentation of Solutions to the Exercise Sheet on the 20.05.2015

Exercise 4-1 Neural Network

Consider the following neural network. It models the game result between two teams D and T, depending on the inputs "self-confidence of team D" (S), "antagonizing power of players of team T" (A) and "efficiency of team D" (E). The hidden neurons model the positive (P) and negative (N) actions of team D.



$\mathbf{x}_i$ $\mathbf{z}$ $\hat{\mathbf{y}}_i$

The output neurons D and T are estimated as follows:

$$\hat{y}_{i,k} = f(\mathbf{x}_i, \mathbf{w}, \mathbf{v})_k = \sum_{h=1}^{M_\phi} w_{k,h}\ \phi_h(\mathbf{x}_i, \mathbf{v}_h)\ ,$$

$$J_N(\mathbf{w}, \mathbf{v}) = \sum_{k=1}^{2}\sum_{i=1}^{N}(y_{i,k} - f(\mathbf{x}_i, \mathbf{w}, \mathbf{v})_k)^2\ .$$

The activation function is:

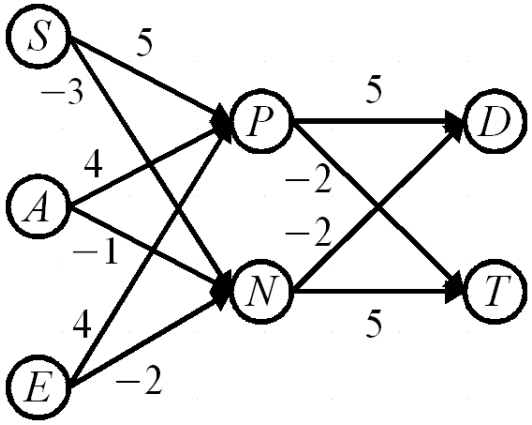
$$z_h(\mathbf{x}_i) = \phi_h(\mathbf{x}_i, \mathbf{v}_h) = \frac{1}{1 + \exp\left(-\sum_{j=1}^{M} v_{h,j} x_{i,j}\right)}\ .$$

The gradient descent for a pattern $\mathbf{x}_i$ is defined as:

$$w_{k,h} \leftarrow w_{k,h} + \eta\ z_h(\mathbf{x}_i)\ (y_{i,k} - f(\mathbf{x}_i, \mathbf{w}, \mathbf{v})_k) \text{ and}$$

$$v_{h,j} \leftarrow v_{h,j} + \eta \sum_{k=1}^{2} w_{k,h}\ z_h(\mathbf{x}_i)\ (1 - z_h(\mathbf{x}_i))\ x_{i,j}\ (y_{i,k} - f(\mathbf{x}_i, \mathbf{w}, \mathbf{v})_k)$$

Consider the already trained neural network:



(a) Compute the prediction $\hat{\mathbf{y}}_i = \begin{pmatrix} D \\ T \end{pmatrix}$ for the input vector $\mathbf{x}_i = \begin{pmatrix} S \\ A \\ E \end{pmatrix} = \begin{pmatrix} -5 \\ 7 \\ 3 \end{pmatrix}$ up to the second decimal.

(b) Use the result from (a) and

$$\mathbf{w}_{k,h} \;=\; \mathbf{w}_{k,h} \;+\; \eta \, \frac{\partial J_N(\mathbf{w}, \mathbf{v})}{\partial w_{k,h}}$$

to conduct one part of the update step of the backpropagation algorithm for the value $\mathbf{y}_i = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$. Use a step size of $\eta = 0.5$.

Exercise 4-2 Building an MLP with Theano

In this exercise we aim at classifying digits using the famous MNIST digits dataset. The dataset consists of 60000 training images and 10000 test images of handwritten digits. Each image has size 28*28, and has assigned a label from zero to nine, denoting the digits value, therefore we can use this dataset for supervised training. Download the MNIST dataset from yann.lecun.com/exdb/mnist. You can download an import script for python from g.sweyla.com/blog/2012/mnist-numpy.

(a) Import the dataset and, in order to get a vectorial representation, flatten the input images, such that each image is represented by a 784-dimensional vector. You should also flatten the labels to get a vectorial representation from a one-dimensional matrix, as these representations are not equivalent in theano.

(b) Given the data, we want to define an MLP for classification. Construct the following network:

- Input x: 768-dimensional (i.e. 768 visible units representing the flattened 28*28 pixel images)
- 100 hidden units h
- 10 output units y representing the label, with a value close to one in the i-th class representing a high probability of the input representing the digit i
- Use a batch size of 100 and a learning rate of 2
- Train 10 epochs. An epoch corresponds to a training interval where each training image is considered once.
- Initialize all input weights of the hidden layer with a random uniform distribution in the range [-0.007,0.007]
- Initialize all input weights of the top classification layer with a random uniform distribution in the range [-0.05,0.05]
- Initialize all biases with zero
- Use a sigmoid activation function for the first layer and softmax for the second layer
- Optimize with negative log-likelihood as a cost function.

The task should be possible to achieve with the knowledge gained from the exercises last week. If you need additional examples you can borrow some code from the deep learning tutorials on deeplearning.net/tutorial. We recommend however that you construct a minimal version of the network on your own without any class structure to gain better insights into the working of Theano. After defining the necessary variables, the network structure can be defined in three lines of code. Defining the updates and the training function can be written in 8 lines of code, and training can be achieved in an additinal five lines.

Computing the loss in this special case where labels are not given as a non-zero entry in a zero-valued vector is not trivial. If hiddenOut is the output of the hidden layer, then the loss can be computed as `loss=-T.mean(T.log(hiddenOut)[T.arange(y.shape[0]),y])`. The mean of this expression simply takes the mean over a minibatch of 100 training examples. `T.log(hiddenOut)` takes the logarithm of the outputs of the hidden layer, and `[T.arange(y.shape[0]),y]` picks the correct entries from the output matrix of size (minibatch,#labels).

(c) The actual output class of the network can be easily derived by taking the $\arg\max_i y_i$. Evaluate your classifier on the test set and calculate your error rate.

Exercise 4-3 Building a CNN with Theano

In the following we will build a Convolutional Neural Network with these building blocks and Theano. Note that the training can take rather long without GPU support, if necessary train on a small dataset for some preliminary tests and later switch to a larger training set for final results. The theano implementation of this exercise is not mandatory and not relevant for your final exams – understanding convolutions and max-pooling, however, is relevant. Import the dataset, do not flatten the images. You should however flatten the labels. Reshape the images to get a tensor of shape (batchsize,1,28,28).

(a) In this exercise we will extend the previous network to a convolutional architecture with max-pooling. To warm up, first answer the following questions, assuming that we have an input image of size (28,28):

- What are the output dimensions when filtering this image using a convolution in "valid" mode with a filter size of (5,5)?
- When this output is filtered using max-pooling with a filter size of (2,2), what is the dimension of the output?
- What would be the output when convolving the (28,28) pixel image with a filter of size (5,5) in "full" and "same" modes?

(b) Import the dataset, do not flatten the images. You should however flatten the labels. Reshape the images to get a tensor of shape (batchsize,1,28,28).

(c) Given the previously transformed training data, we want to define a Convolutional Neural Network for classification. Construct the following network:

- Input x: (28,28)-dimensional
- First layer: Convolutional layer with
  - One Input channel (black/white image)
  - Weights of the convolutional filter are initialized with a uniform distribution to [-0.06,0.06]
  - 3 output channels (These output channels are called feature maps
  - Filter size of 5x5
  - Question: What are the output dimensions of this layer?
- Second layer: Max-pooling layer with
  - Filter size of 2x2
  - Sigmoidal activation function and biases initialized to zero
  - Flatten the outputs as the next layers will be equivalent to the previously defined MLP. You can use T.flatten(2). The remainder of the network is mostly equivalent to the MLP.
  - Question: What are the output dimensions of this layer?
- Third layer:
  - You have calculated the number of inputs by your own
  - 100 hidden neurons output
  - Weights of the convolutional filter are initialized with a uniform distribution to [-0.01,0.01]
  - Sigmoidal activation function and biases initialized to zero
- Fourth (classification) layer:
  - 100 inputs
  - 10 outputs
  - Softmax activation function
- Use a batch size of 100 and a learning rate of 2

- Train 10 epochs. An epoch corresponds to a training interval where each training image is considered once.
- Optimize with negative log-likelihood as a cost function.

(d) The actual output class of the network can be easily derived by taking the $\arg\max_i y_i$. Evaluate your classifier on the test set and calculate your error rate.