Machine Learning - IV

Methods of Knowledge Based Software Development

S. Nõmm

[1]Department of Software Science, Tallinn University of Technology

22.12.2017

- Part I Back propagation

Some additional references:

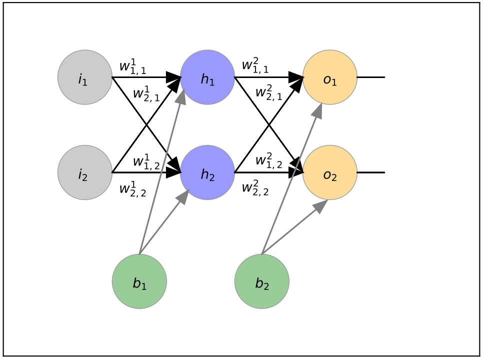
- https://mattmazur.com/2015/03/17/

Back propagation

- Back propagation is the method to train neural networks.
- The goal is to optimize weights so that the network will map correctly arbitrary inputs to outputs.
- Popular and implemented for many programming languages.

Back propagation: example

Let us consider a neural network with two inputs, two outputs and a hidden layer consisting of two neurons.



Additional and hidden neurons will include some bias.

Back propagation: example

Very "artificial" case: Training set is given by:

- inputs: 0.5, 0.1
- desired outputs: 0.01, 0.99
- initial weights are: $w^1_{1,1} = 0.15$, $w^1_{2,1} = 0.2$, $w^1_{1,2} = 0.25$,$w^1_{2,2} = 0.3$ $w^2_{1,1} = 0.4$, $w^2_{2,1} = 0.45$, $w^2_{1,2} = 0.5$,$w^2_{2,2} = 0.55$
- initial biases are $b_1 = 0.35$ and $b_2 = 0.6$

Example of Back propagation: The forward pass

Neural network predicts the outputs given the weights and biases for the inputs 0.05 and 0.1. NB! Keep in mind different notations.

$$
\begin{aligned}
h_1 &= \frac{1}{1+e^{-1(w^1_{1,1}*i_1+w^1_{2,1}*i_2+b_1*1)}} = 0.593\\
h_2 &= \frac{1}{1+e^{-1(w^1_{1,2}*i_1+w^1_{2,2}*i_2+b_1*1)}} = 0.597\\
y_1 &= \frac{1}{1+e^{-1(w^2_{1,1}*h_1+w^2_{2,1}*h_2+b_2*1)}} = 0.751\\
y_2 &= \frac{1}{1+e^{-1(w^2_{1,2}*h_1+w^2_{2,2}*h_2+b_2*1)}} = 0.772
\end{aligned}
$$

Total error is given by $E_t = \sum 0.5(y_i - \hat{y})^2 = 0.298$. Where y_i is desired output or (target) and $\hat{y}$ is the actual output of the neural network.

Example of Back propagation: The backward pass

How much the change of weight $w^m_{i,j}$ affects the total error? Apply the chain rule! (For simplicity let us start with the neuron of the output (second) layer).

$$\frac{\partial E_t}{\partial w^2_{1,1}} = \frac{\partial 0.5((y_1 - \hat{y_1})^2 + (y_2 - \hat{y_2})^2)}{\partial w^2_{1,1}}$$

obviously the second term does not depend on $w^2_{1,1}$ therefore

$$\frac{\partial E_t}{\partial w^2_{1,1}} = (y_1 - \hat{y_1})\frac{\partial (y_1 - \hat{y_1})}{\partial w^2_{1,1}}$$

Note that $\hat{y_1}$ is logistic function and derivative of logistic function $f(x)$ equals $f(x)(1-f(x))$. This lead

$$\frac{\partial E_t}{\partial w^2_{1,1}} = (y_1 - \hat{y_1}) * \hat{y_1} * (1 - \hat{y_1}) * \frac{\partial (w^2_{1,1} * h_1 + w^2_{2,1} * h_2 + b_2 * 1)}{\partial w^2_{1,1}}$$

$$\frac{\partial E_t}{\partial w^2_{1,1}} = (y_1 - \hat{y_1})\hat{y_1}(1 - \hat{y_1})h1 = 0.741 \cdot 0.187 \cdot 0.593 = 0.082$$

Example of Back propagation: The backward pass

In order to decrease the error, one should subtract this value from the current value of the weight $w^2_{1,1}$. As an option the value may be multiplied by a learning constant.

$$w^{2+}_{1,1} = w^2_{1,1} - \eta \frac{\partial E_t}{w^2_{1,1}} = 0.358$$

by repeating this process one may get:

$$\begin{array}{rcl} w^{2+}_{2,1} & = & 0.4 \\ w^{2+}_{1,2} & = & 0.51 \\ w^{2+}_{2,2} & = & 0.56 \end{array}$$

NB! these weights will be used only after weights for the hidden layer are updated!!!

Example of Back propagation: The backward pass (hidden layer)

$$\frac{\partial E_t}{\partial w^2_{1,1}} = \frac{\partial 0.5((y_1 - \hat{y_1})^2 + (y_2 - \hat{y_2})^2)}{\partial w^1_{1,1}}$$

$$\begin{aligned}\frac{\partial E_t}{\partial w^1_{1,1}} &= (y_1 - \hat{y_1})\hat{y_1}(1 - \hat{y_1})w^2_{1,1}\frac{\partial h_1}{\partial w^1_{1,1}} + \text{similar} \quad \text{term} = \\ &= (y_1 - \hat{y_1})\hat{y_1}(1 - \hat{y_1})w^2_{1,1}h_1(1 - h_1) * 0.05 + \text{similar} \quad \text{term}.\end{aligned}$$

Example of Back propagation: The backward pass (hidden layer)

This procedure will lead to the updated values for the hidden layer.

$$\begin{aligned} w_{1,1}^{1+} &= 0.15 \\ w_{2,1}^{1+} &= 0.2 \\ w_{1,2}^{1+} &= 0.25 \\ w_{2,2}^{1+} &= 0.3 \end{aligned}$$

Now perform forward pass again and compute the error. After certain number of iterations the process should converge.

- Part II Deep learning

Some additional references:

- http://cs231n.github.io/convolutional-networks/
- http://www.deeplearningbook.org/

Deep Learning

- Inspired by the recent (relatively) discoveries in neurobiology.
- Different levels of abstractions are represented by different layers.
- Unlike Cybenko theorem based approach propagates the idea of multi-layered networks.
- Unlike traditional algorithms, DL does not suffer from curse of dimensionality.
- Allows to overcome challenges of feature extraction.

Deep Learning Applications

- Computer vision: image and video analysis (object recognition etc.).
- Speech analysis and recognition.
- Financial analysis.
- Machine translation.
- Entertainment, culture etc.
- As a tool for scientific research.

Deep Learning Formal definition

Definition

Deep learning is a class of machine learning algorithms, that satisfy following conditions:

- Uses a cascade of multiply layers of nonlinear processing units to feature extractions and transformations.
- Implements hierarchy concept, each level of abstraction is represented by a layer of cascade.
- Uses gradient descent (like) algorithm via back propagation.

Note! There are many similar definitions, though some variations are possible.

Convolutional neural networks (CNN)

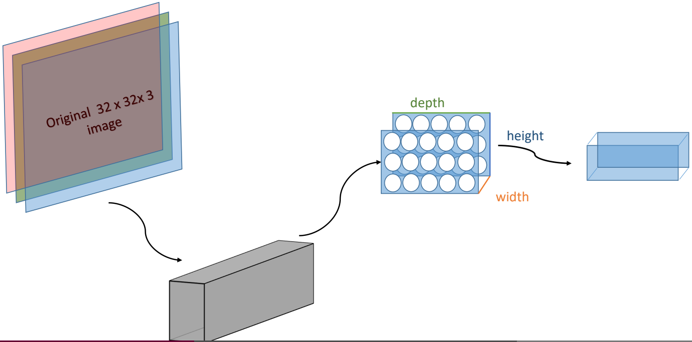
- Convolutional NN architecture is most probably the simplest case among deep learning structures.
- Most common usage is for image analysis (object recognition).
- Note there are other possible application areas like, text data mining, NLP, etc.
- What is the distinctive property of the task, that allows to tackle it with CNN?

Convolutional neural networks (CNN)

- **Assumption:** The data points have image like structure.
- 3D volumes of neurons.
- Specific types of layers; *input*, *convolution*, *relu*, *pool*, *fully connected*.
- Convolution and fully connected layers do have parameters but relu and pool do not.
- Convolution, fully connected and pool layers do have hyperparameters but relu does not.

Convolutional neural networks (CNN)

- **Assumption:** The data points have image like structure.
- 3D volumes of neurons.
- Specific types of layers; *input*, *convolution*, *relu*, *pool*, *fully connected*.
- Convolutional and fully connected layers do have parameters but relu and pool do not.
- Convolutional, fully connected and pool layers do have hyperparameters but relu does not.



CNN layer by layer explanation

- Input Takes row image
- Convolutional layer. Neurons connected to the local **region** of the input. This will result in a volume.
- Relu layer applies activation function.
- Pool layer performs a downsampling operation along the spatial dimensions.
- Fully connected layer computes the class scores.

CNN image processing steps

Let us suppose that RGB image is encoded by $32 \times 32 \times 3$ matrix. 32×32 is the spatial dimensions of data point.

- Input is $32 \times 32 \times 3$ matrix.
- Suppose, that 12 filters is applied. Convolution layer. will produce the volume of $32 \times 32 \times 12$.
- Relu layer applies activation function.
- Pool layer performs a downsampling operation along the spatial dimensions. This will result in $16 \times 16 \times 12$.
- Fully connected layer computes the class scores.

Convolutional layer in detail

- Convolutional layer parameters consists of the set of learnable filters.
- In the case of example above the size of the filter may be $5 \times 5 \times 3$ (some times referred as *receptive field*).
- During the process referred as *convolve* (basically forward pass) of each filter across the spatial dimensions, dot products between weights and inputs are computed and 2 - d activation map will be produced. Network will learn filters, that will be activated by the presence of certain feature like edge. This will produce the volume of the same spatial dimensions as original image but third dimension equal to the number of filters.
- Pool layer performs a downsampling operation along the spatial dimensions. This will result in $16 \times 16 \times 12$.
- Fully connected layer computes the class scores.

Spatial arrangement & parameter sharing

There are three hyperparameters to control the size of the output volume

- Depth is the number of filters. Different neurons along the depth activate in presence of different features. This parameter referred as *depth column or fibre*.
- How many pixels ata time we move the filter. This parameter is referred as *stride*.
- Sometimes input volume is padded with zeros around the border. It allows to control the size of the output volumes. The process is called *zero-padding*.
- Parameter sharing is another way to control the number of parameters. Based on the assumption: If one feature is useful to compute at one spatial point it should be useful at a different point as well.

Notes on practical architecture design

- It is common practice to periodically insert pooling layers.
- In practice architecture may consist of many successive convolution and pooling layers.
- Normalization layer.
- Modern connectivity structures.