Assignment 2

Anonymous & Anonymous

6 April 2017

In this assignment we aim to understand, apply, and master *Deep Neural Networks* and *Deep Learning*. In the first section we will focus on quickly describing 3 types of Deep Neural Networks (Convolutional, Recurrent, and Autoencoders), plus providing a proof of understanding through experimenting with existing code and results. The second section will then introduce our DL Challenge project, followed by a detailed description about our approach, experimentation, results, and a concluding discussion.

1. Deep Neural Networks

Convolutional

A *Convolutional Neural Network (CNN)*, the first (potentially) Deep Neural Network we consider, uses insights from the organisation of the visual cortex in the brain and tries to summarize information by using receptive fields. In simple words, receptive fields consist of a small set of conditions and provide stronger output / are responsive whenever a stimulus matches these conditions. They are locally applied on the input like a moving window and produce a new layer called feature map. One example would be an edge detector in image recognition, which is resembled by a receptive field which responds more strongly whenever one side of the receptive field receives dark (e.g., represented as “-1”) and the other side receives light (“1”) information.

Example: CIFAR-10.

This dataset stems from a typical computer vision problem: to identify what is on the picture. It consists of 60,000 photos of 32 x 32 pixels. There are 10 different classes such as cars, birds, or airplanes, which the neural network is supposed to be able to categorize after learning from the pictures. 50,000 of the images are used for the training set in the present example, while 10,000 images are used to evaluate the overall performance. A CNN is used to catch the underlying features that make up the uniqueness of the image categories.

A simple CNN to learn about the data had the following structure: 1. Convolutional input layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. 2. Dropout set to 20%. 3. Convolutional layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. 4. Max Pool layer with size 2×2. 5. Flatten layer. 6. Fully connected layer with 512 units and a rectifier activation function. 7. Dropout set to 50%. 8. Fully connected output layer with 10 units and a softmax activation function.

Recurrent

Recurrent Neural Networks (RNNs), in contrast to *Feedforward Neural Networks*, extend the network by including cyclic aspects between the units. Thus, instead of units depending statically on the other layers, elements inside the network may be allowed to change dynamically, so that activations over time can alter the expression of a layer.

This enables us to make use of sequential information, i.e. we do not have to assume that all inputs (and outputs) are independent of each other. We can use previous computations to alter the network on how to process new incoming information; in other words the network remembers what has happened just before. Good examples for the application of RNNs lie in text processing or *Neuro-Linguistic Programming (NLP)*.

Example: the IMDB movie review sentiment classification problem.

Autoencoders

The examples above are for supervised learning. Autoencoders instead try to learn "themselves" about the structure of the input given. When given an input vector x, the goal is to return an output vector $\hat{x}$ that resembles x as closely as possible. The identity function that was created in such a way is what the experimenter is interested in, as it might reveal new ways of summarizing the data at hand or reveal hidden structures in the data. This is done by compression through forcing the network to represent a smaller and as-similar-as-possible version of the original by fewer hidden nodes than what would be needed to just push through all the information. For instance, picutres of the size 10 x 10 pixels might be compressed to 7 x 7 pixel versions. If the restructured 10 x 10 version $\hat{x}$ is close enough to $\hat{x}$ the autoencoder successfully compressed the data. A related method is PCA, which goal is to project higher-dimensional objects into a lower-dimensional space, so that most of its information is preserved.

2. DL Challenge