Neuron Network

Aleksander Sas
2015/2016

Introduction:
This paper describes my approach to recognizing images. Database is CIFAR-10, consisting of 60000 images. Images belong to 10 category (cars, lorries, ships, planes, dogs, cats, deers, frogs, horses and birds). Each image has 32x32 pixels with RGB colors. A category object is in the middle of the image. A goal of the problem, is to read pixels and decide, to which category the object of image belongs. The core of my recognition system is neuron network, build using Theano and Lasagne library. Neurons are trained on GPU. A start point for my network was Lasagne MNIST example, very similar to recommended start point presented on lecture.

Network Architecture:
The network consists of six layers: two convolutions layers, two max_pool layers and two fully connected feed forward layers with threshold activation function. Convolution layers use 5x5 filters. In max_pool layer only one value from 2x2 rectangular is taken and passed to the next layer.

- First convolution layer takes 32x32x3 pixel image, uses 60 5x5 dimension filters. This layer reduces the image dimension to 28x28.
- First max-pool layer reduce the image to 14x14 dimension by taking the highest input from each 2x2 rectangle
- Second convolution layer uses 200 5x5 dimension filters. This layer reduces image to 10x10.
- Second max-pool layer, just like the first one, takes the highest input in 2x2 dimension rectangle. Reduces image dimension to 5x5.
- First feed forward layer uses simple threshold function $\varphi(x)=\max(0,x)$. There are 1200 neurons in this layer.
- Second feed forward layer, the last one, uses softmax to produce probabilities on the output of the network. There are 10 neurons in this layer. The neurons correspond to image categories.

Motivation for using convolution layers is to get a bit more complex features then simple pixel. Pixel values are correlated, so convolution layer layer, taking only a few adjacent pixels, tends to discover edges, and simple shapes.
The learning rate is being changed in every batch according to the equation:

$$2.5\cdot 10^{-3}\frac{15000}{max(15000,b)}$$

where b is the batch number. Momentum is also used with a ratio set to 0.9.

Experiments and required improvements:
The network described above reaches up to 70-74.5% depending on regularization ratio. It turns out, that the network tends to overfit, reaching 100% accuracy on learning set. Setting the regularization ratio to $1.8\cdot 10^{-3}$ prevent the network from total overfitting. Finally dropout turns out the best solution. The same network equipped with dropout reach 81% accuracy on the validation set, and 80% on the test set, however it steel reach 100% accuracy on training set. In final network droput is used instead of regularization. Several regularization ratios (in range from $2\cdot 10^{-5}$ to $1.8\cdot 10^{-3}$) has been testes and no imporvents has been achieved. Setting regularization to $1.8\cdot 10^{-3}$ in network with droput leads to decrease accuracy to 78%, such ratio was good in network without droput. First feed forward layer has 0.5 dropout ratio and the last one 0.2. Convolution layers have no dropout because it causes performance deterioration (from 80% to 75%).

The output error is computed using cross entropy instead of squared error. This is recommended way to compute an error in Lasagne, in the case of multi class classification. The network needs about 30 epochs to reach 80% accuracy.

Conclusion:
Start point network reach about 70% accuracy. It was necessary to increase convolution and neuron number to make network more clever. Unfortunately, new network prone to overfit, so I had to add droput.