Deep Learning- Assignment 1

SugarData (up276, vec241)

February 2016

1 Backpropagation

1. Let's write $\frac{\partial E}{\partial x_{in}}$ in terms of $\frac{\partial E}{\partial x_{out}}$.
   We have:

$$\frac{\partial E}{\partial X_{i-1}} = \frac{\partial E}{\partial X_i}\frac{\partial F_i(X_{i-1}, W_i)}{\partial X_{i-1}}$$

   Hence:

$$\frac{\partial E}{\partial x_{in}} = \frac{\partial E}{\partial x_{out}}\frac{\partial x_{out}}{\partial x_{in}} = \frac{\partial E}{\partial x_{out}}\frac{\partial}{\partial x_{in}}(\frac{1}{1+e^{-x_{in}}}) = \frac{\partial E}{\partial x_{out}}\frac{1}{(e^{\frac{x_{in}}{2}}+e^{\frac{-x_{in}}{2}})^2}$$

2. **for** $i \neq j$:

$$\frac{\partial (X_{out})_i}{\partial (X_{in})_j} = \beta\frac{e^{-\beta(X_{in})_i}.e^{-\beta(X_{in})_j}}{(\sum_k e^{-\beta(X_{in})_k})^2} = \beta(X_{out})_i(X_{out})_j$$

   for $i = j$:

$$\frac{\partial (X_{out})_i}{\partial (X_{in})_i} = -\beta(X_{out})_i + \beta((X_{out})_i)^2 = \beta(X_{out})_i((X_{out})_i - 1)$$

2 Torch (MNIST Handwritten Digit Recognition)

In this section, we will present our work regarding MNIST Handwritten Digit Recognition problem. We will use the code located at $https://github.com/yjxiao/ds-ga-1008-a1$ which is based on Clement Farabet's tutorial on supervised learning $http://code.madbits.com/wiki/doku.php?id=tutorial_supervised$.

We will first present the objective of MNIST Handwritten Digit Recognition (1.), the data set (2.) and the original model we are going to work on (we will focus on the ConvNet model) (3.). Then, we will present our experiment setups and how we changed the original model (4.), and then the results we get (5.). Finally, we will analyse our results (6.).

1. **Objective :** In this Assignment we are dealing with the Handwritten digits Multi classification problem. We are using supervised learning technique to tune our ConvNet model where we will first train our model on Training Data and then evaluate its performance on Testing data. For this problem we are using MNIST Database which is specially designed to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

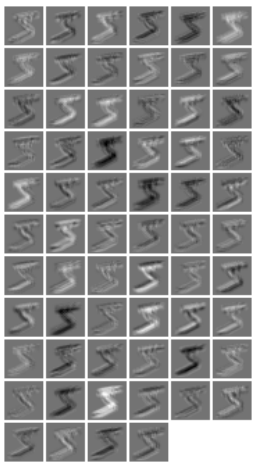


Figure 1: The 64 feature maps after first layer of first block (spatial convolution)

Figure 2: The 64 feature maps after second layer of first block (tanh)

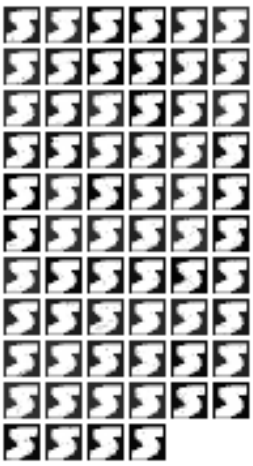


Figure 3: The 64 feature maps after third layer of first block (L2 Pooling)

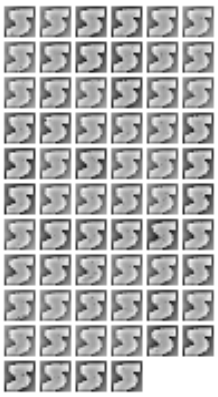


Figure 4: The 64 feature maps after fourth layer of first block (substractive normalization)

2. **MNIST Database :** The MNIST database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. For Our problem we will divide 60000 Training samples into 80:20 ratio, where we will use 80 of the Training Data to train our model and 20 of the Training data as a validation set on which we will evaluate our model performance.[1]

3. **Model architecture** The model we are going to implement here to solve the MNIST Handwritten Digit Recognition task is a ConvNet model made of 3 blocks of multiple layers each.

   The first two blocks are similar and made of the following layers :

   (a) Convolution : 5x5 kernel, 1x1 stride. The convolution layer of the first block takes in input 1 feature (the image of the digit), and outputs 64 feature maps. The convolution layer of the second block takes in input 64 feature maps and outputs 64 feature maps as well.

   (b) Activation function : tanh.

   (c) Pooling : L2 Pooling, 2x2 kernel, 2x2 stride.Pooling "can be imagined as giving an increased weight to stronger features and suppressing weaker features"[3]

   (d) Normalization : Spatial normalization is an image processing step of transforming different sets of data into one coordinate system. Data may be multiple photographs, data from different sensors, times, depths, or viewpoints. Normalization is necessary in order to be able to compare or integrate the data obtained from these different measurements.[2]

   Figures 1 to 4 above show the 64 feature maps after each 4 layers of the first block.

   The third block is standard 2-layer neural network made of 2 linear function layers with an activation layer (using tanh function) between them. this block takes in input 64 feature maps and outputs a vector of dimension 10, corresponding to the ten classes corresponding to the 10 possible digits (0-9).

4. **Experiment setups.** Based on the original model, our modifications were of 2 natures, changing parameters of the functions already implemented, and adding new functions. Given the duration of training on the full training data set and the number of parameters we wanteds to try, we used the following methodology in order to choose the best model possible

(a) For each new parameter / function we tried : train on 'small' dataset (600 instances, with a repartition of 80:20 for train and validation sets), for 5 epochs, then report accuracy on train and validation sets

(b) Choose the parameters/functions that give the best results on validation set. Train again on 'full' data set (60.000 instances, with a repartition of 80:20 for train and validation sets) for 11 epochs

5. **Results.**

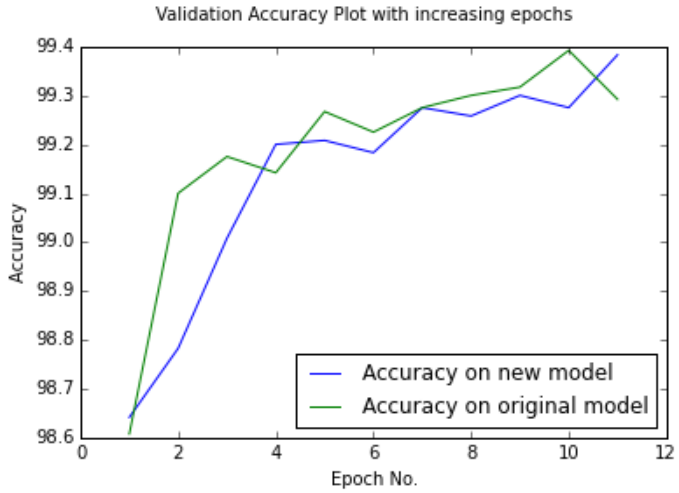
(a) The table below presents the results obtained on 'small' data set, after 5 epochs :

Parameter/function modified	Acc. on train set	Acc. on val. set	#Epoch	Dataset size
Original model	90.83%	86.7%	5	'small'
Pooling : average pooling	67.29%	65.83%	5	'small'
Pooling : max pooling	80.42%	73.33%	5	'small'
nstates : 64,128,128	92.08%	87.5%	5	'small'
nstates : 64,128,258	92.92%	85.83%	5	'small'
Activation function : sigmoid	13.3%	11.67%	5	'small'
Activation function : ReLU	92.29%	85%	5	'small'
Dropout	49.17%	86.7%	5	'small'

(b) Following the previous results we obtained, we choose to implement the following model : original model with nstates = 64,128,128 and Dropout. The table below presents the comparison between the original model and our new one, on full data set, after 11 epochs :

Parameter/function modified	Acc. on train set	Acc. on val. set	#Epoch	Dataset size
Original model	0.99950	0.99292	11	'full'
New model	0.68140	0.99383	11	'full'

Below is the graph comparing the learning rates on the validation set between the original model and our new model :



(c) **Kaggle submission.** We can observe from the previous results that both model are very close and have almost similar learning rate. Hence, we will chose to submit on Kaggle new model. We will also try to run it longer than 11 epochs.

6. **Results analysys.** For the final submission we made on Kaggle, we got an accuracy of 0.99580 on test set (accuracy displayed on Kaggle's Leaderboard). First, we can say that the original was already performing very well. We can suppose that some other different models might perform better of course (such as models using different neural networks). But the objective of this assignment was here to keep the core structure of the original model and try to change various parameters. Hence, since none of our changes would significantly improve the results (most would decrease significantly accuracy, some would give around same accuracy, cf table presenting the results above), we can conclude that the original model was already using the optimal parameters.

References

[1] Yann LeCun, Corinna Cortes, Christopher J.C. Burges *The MNIST database of handwritten digits. http : //yann.lecun.com/exdb/mnist/*

[2] Spatial normalization Wikipedia page *https : //en.wikipedia.org/wiki/Spatial*$_n$*ormalization/*

[3] Pierre Sermanet, Soumith Chintala, Yann LeCun *Convolutional Neural Networks Applied to House Numbers Digit Classification http : //arxiv.org/pdf/*1204.3968*.pdf*