CSI 5138 R00 Homework Exercise 2

Vasileios Lioutas

October 12, 2018

Question 1. The back-progation algorithm, in order to compute the gradients $\frac{\partial \mathcal{L}}{\partial A}$ and $\frac{\partial \mathcal{L}}{\partial B}$ would have to follow the gradient using the chain rule. Specifically:

- For computing the $\frac{\partial \mathcal{L}}{\partial A}$ we must break all the derivatives until the very last equations.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial A} &= \frac{\partial \mathcal{L}}{\partial w} \odot \frac{\partial w}{\partial A} \\
\frac{\partial \mathcal{L}}{\partial w} &= 2 * w \\
\frac{\partial w}{\partial A} &= A.\frac{\partial z}{\partial A} + z.\frac{\partial A}{\partial A} = A.\frac{\partial z}{\partial A} + z.1 \\
\frac{\partial z}{\partial A} &= A.\frac{\partial (u \odot v)}{\partial A} + (u \odot v).\frac{\partial A}{\partial A} = A.\frac{\partial (u \odot v)}{\partial A} + (u \odot v).1 \\
\frac{\partial (u \odot v)}{\partial A} &= u \odot \frac{\partial v}{\partial A} + v \odot \frac{\partial u}{\partial A} \\
\frac{\partial v}{\partial A} &= \frac{\partial (B.x)}{\partial A} = 0 \\
\frac{\partial u}{\partial A} &= \frac{\partial \sigma(y)}{\partial y} \odot \frac{\partial y}{\partial A} \\
\frac{\partial \sigma(y)}{\partial y} &= \sigma(y)(1-\sigma(y)) \\
\frac{\partial y}{\partial A} &= A.\frac{\partial x}{\partial A} + x.\frac{\partial A}{\partial A} = x.1
\end{aligned}$$

Note - where . is the dot product operation, $\odot$ is the element-wise multiplication operation and $*$ is the multiplication operation with scalar.

When it's done, we replace each derivative and we end up with:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial A} = 2 * A.A.(\sigma(A.x) \odot B.x) \odot (A.A.(B.x \odot (\sigma(A.x)(1-\sigma(A.x))) \odot x.1) \\
+A.(\sigma(A.x) \odot B.x).1 \\
+A.(\sigma(A.x) \odot B.x).1)
\end{aligned}$$

- Accordingly for computing the $\frac{\partial \mathcal{L}}{\partial B}$:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial B} &= \frac{\partial \mathcal{L}}{\partial w} \odot \frac{\partial w}{\partial B} \\
\frac{\partial \mathcal{L}}{\partial w} &= 2 * w \\
\frac{\partial w}{\partial B} &= A.\frac{\partial z}{\partial B} + z.\frac{\partial A}{\partial B} = A.\frac{\partial z}{\partial B} \\
\frac{\partial z}{\partial B} &= A.\frac{\partial (u \odot v)}{\partial B} + (u \odot v).\frac{\partial A}{\partial B} = A.\frac{\partial (u \odot v)}{\partial B} \\
\frac{\partial (u \odot v)}{\partial B} &= u \odot \frac{\partial v}{\partial B} + v \odot \frac{\partial u}{\partial B} = u \odot \frac{\partial v}{\partial B} \\
\frac{\partial v}{\partial B} &= \frac{\partial (B.x)}{\partial B} = x.1 \\
\frac{\partial u}{\partial B} &= \frac{\partial (\sigma(A.x))}{\partial B} = 0
\end{aligned}$$

If we replace each derivative, we end up with:

$$\frac{\partial \mathcal{L}}{\partial B} = 2 * A.A.(\sigma(A.x) \odot B.x) \odot (A.A.(\sigma(A.x) \odot x.1))$$

The implementation of the back-propagation algorithm would first create the derivative networks $\partial\mathcal{G}[x, A]$ and $\partial\mathcal{G}[x, B]$. Then, compute their dual networks $(\partial\mathcal{G}[x, A])^T$ and $(\partial\mathcal{G}[x, B])^T$. Finally, pass 1 as input to the dual networks and obtain the output which corresponds to $\frac{\partial \mathcal{L}}{\partial A}$ and $\frac{\partial \mathcal{L}}{\partial B}$.

Question 2. Given the models:

$$\begin{aligned}
&\mathcal{H}_1 := \{softmax(Wx) : W \in \mathbb{R}^{K \times m}\} \\
&\mathcal{H}_2 := \{softmax((A + B)Cx) : A \in \mathbb{R}^{K \times K}, B \in \mathbb{R}^{K \times K}, C \in \mathbb{R}^{K \times m}\}
\end{aligned}$$

In order to prove that the two models $\mathcal{H}_1$ and $\mathcal{H}_2$ are equal, we need to prove that both models can express the same family of hypotheses. This can be seen by showing that each family of hypotheses from one model can be a subset of all the possible hypotheses of the other model. Specifically:

- For any hypothesis in the model $\mathcal{H}_1$, there is an adjustment of the learnable variables (A, B, C) in the model $\mathcal{H}_2$ giving rise to the same hypothesis from $\mathcal{H}_1$.
- For any hypothesis in the model $\mathcal{H}_2$, there is an adjustment of the learnable variable (W) in the model $\mathcal{H}_1$, which can be decomposed to $W = (W_1 + W_2)W_3$ that can give rise to the hypothesis where $W_1 = A, W_2 = B, W_3 = C$ which is the same with the hypothesis from $\mathcal{H}_2$.

Question 3. Given that we know that in images that belong to second class, the two diagonals have distance that is at least 3 pixels, we can use a 3×3 kernel to highlight all the diagonals in the image. Specifically, we can use the following kernel:

0	0	1
0	1	0
1	0	0

This kernel will highlight, as it passes through the image, any 45° diagonal line with its peak when it matches a line with 3 pixels. Then, we can threshold the feature map to only count the peaks of our line detection. For the kernel we chose, the peak will have the number 3 meaning all three pixels in the diagonal are present in the image at the moment of pass. Finally, we can count the number of peaks and if the result is 1 it means it belongs to Class 1, otherwise it belongs to Class 2. For example:

In the following random Class 2 image we get the following feature map after the convolution operation.

100

100

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$*$

0	0	1
0	1	0
1	0	0

$=$

98

0	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	**2**	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

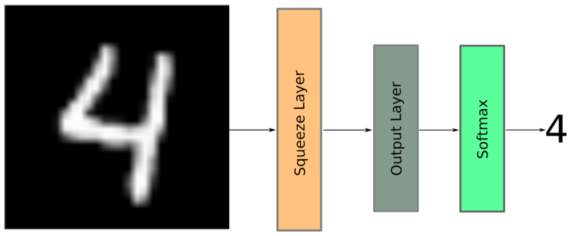
98

Note - I only drew a 20×20 image as an example but the same principle applies to the 100×100 images as well.

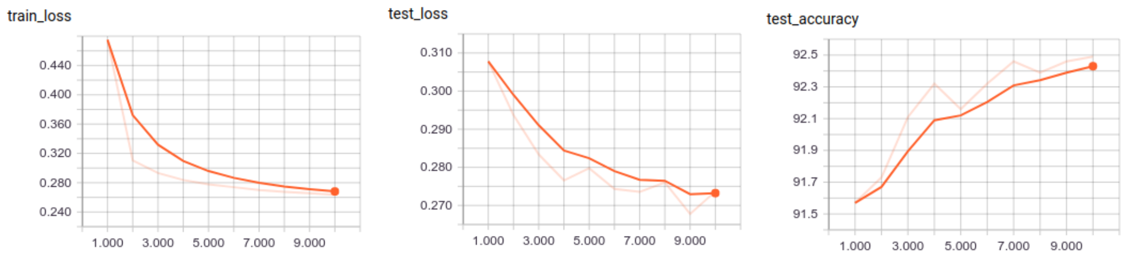
Next, we threshold the feature map to only keep the values of 3 (where the peak is) and the number of peaks defines the class of the image. In our example, the number of peaks is two, thus the classifier correclty predicts that it's a Class 2 image.

Question 4. Below are the different model architectures I trained and tested along with the results I got from each model. All models have been trained for 10 epochs using the default settings for Adadelta optimizer[1]. The number of batch size was set to 128. All models were trained using the Negative Log-Likelihood loss function.

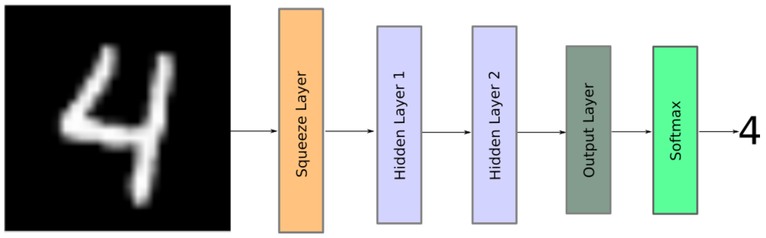
- Softmax Regression: The model first squeezes the image matrices from 28×28 to 784 length vectors. Next, using an output layer it transforms this 784 vector to 10 which corresponds to the number of classes. Finally, it computes the softmax probabilities of this vector.



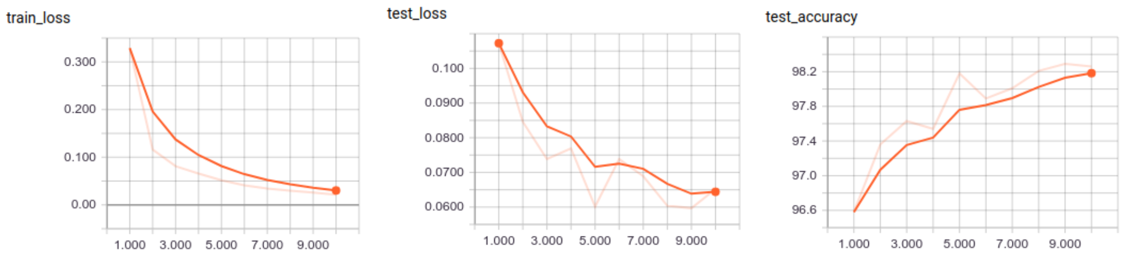
Below it can be seen the train loss, the test loss and the test accuracy of the model.



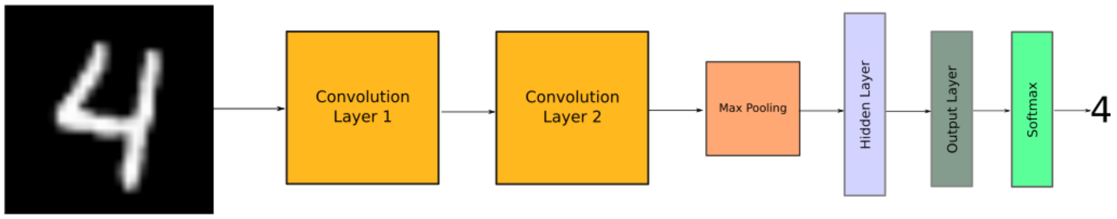
- MLP: The model first squeezes the image matrices from 28×28 to 784 length vectors. Next, using two hidden layers it transforms first the 784 vector to 512 and after the 512 vector to a 512 vector. Next, using an output layer it transforms this 512 vector to 10 which corresponds to the number of classes. Finally, it computes the softmax probabilities of this vector. The model was trained using dropout (0.2) between the layers. The ReLU activation function was used for the hidden layers.



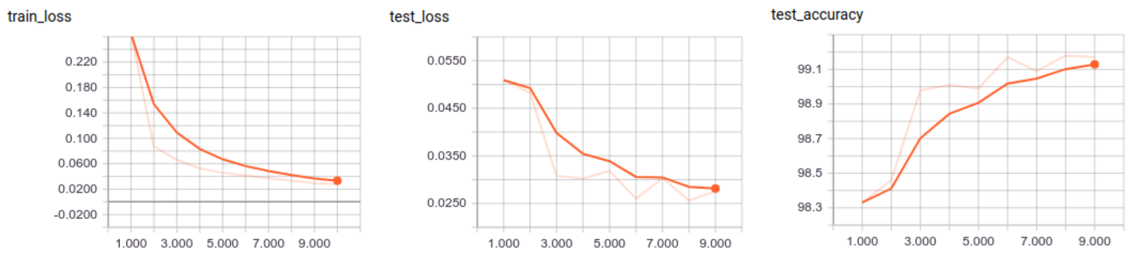
Below it can be seen the train loss, the test loss and the test accuracy of the model.



- ConvNet: The model first passes the image through a convolution layer with kernel size 3 and output number of filters 32 (padding=0 and stride=1). Next, it applies a non-linear function (ReLU) to the features maps. The resulted feature maps are passed through a second convolution layer with kernel size 3 and output number of filters 64 (padding=0 and stride=1) and a ReLU activation function. Afterward, it applies a Max Pooling operation with kernel size 2. These results to features maps with size 12×12×64. Next, it applies a dropout layer (0.25). After, it squeezes these maps to a vector of size 9216 and pass it through a hidden layer of size 128. Next, after a dropout layer (0.5) it uses an output layer which transforms the 128 vector to 10 which corresponds to the number of classes. Finally, it computes the softmax probabilities of this vector.



Below it can be seen the train loss, the test loss and the test accuracy of the model.



It's interesting to note how well the ConvNet is able to learn much better features from the first epoch in order to output accurate predictions. Specifically, on the first epoch the ConvNet achieves 98.33% accuracy which outperforms the previous best results gotten from Softmax Regression and MLP models.

Another interesting remark is that even the simplest model (Softmax Regression), is able to achieve descent results which means that the MMIST dataset is a quite trivial dataset to be learned.

One can also notice that the Softmax Regression model, after 10 epochs was only able to improve 1% compared to epoch 1. This means that the model is very limited and doesn't leave a lot of room for learning more useful transformations.

References

[1] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.