Otto-Friedrich-Universität Bamberg
International Software Systems Science Program
KogSys-ML-M Winter Term 2019/20

Assignment – 2
Perceptrons and Deep Learning

Team members

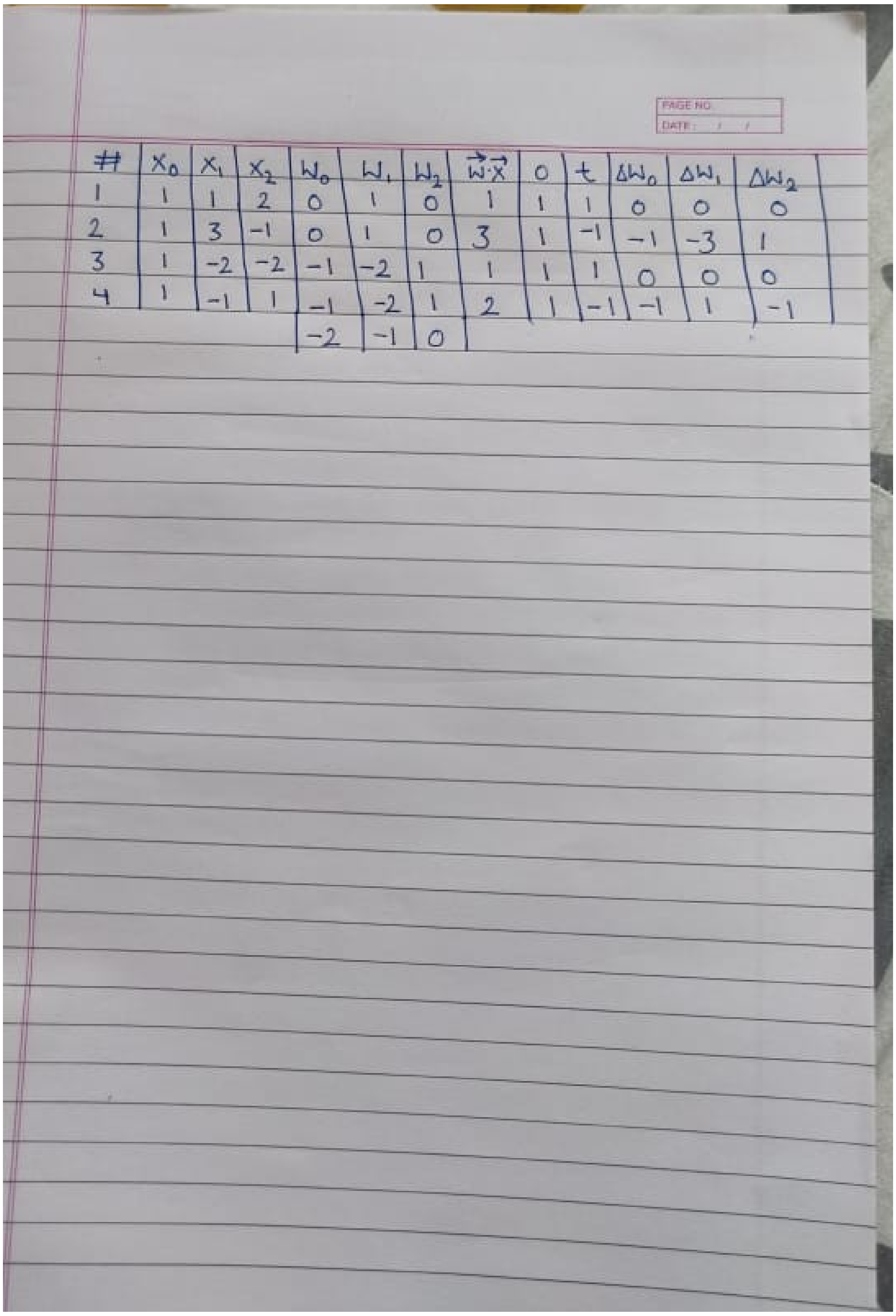
Divyakant Nihalani
(divyakant-jayprakash.nihalani@stud.uni-bamberg.de)

Kiran Akram
(kiran.akram@stud.uni-bamberg.de)

Koshika gaur
(koshika.gaur@stud.uni-bamberg.de)

Siddharth bhatheja
(siddharth.bhatheja@stud.uni-bamberg.de)

1. Perceptron Training Rule



PAGE NO.
DATE: / /

#	x_0	x_1	x_2	w_0	w_1	w_2	$\vec{w}\cdot\vec{x}$	o	t	Δw_0	Δw_1	Δw_2
1	1	1	2	0	1	0	1	1	1	0	0	0
2	1	3	-1	0	1	0	3	1	-1	-1	-3	1
3	1	-2	-2	-1	-2	1	1	1	1	0	0	0
4	1	-1	1	-1	-2	1	2	1	-1	-1	1	-1
				-2	-1	0						

2 Multi layer Perceptrons and Backpropagation

1. (1 Point) Considering the dimensions of the input and output values of the perceptron, provide a real world concept that could be used for the input. What could the resulting output vector represent?

The input used could be a value composed of three binary numbers and the resulting output could be invert of the input value.

0 0 0 - - - > 1 1 1
0 0 1 - - - > 1 1 0
0 1 0 - - - > 1 0 1
0 1 1 - - - > 1 0 0
1 0 0 - - - > 0 1 1
1 0 1 - - - > 0 1 0
1 1 0 - - - > 0 0 1
1 1 1 - - - > 0 0 0

2. (1 Point) Consider the propagation from first hidden layer to second hidden layer. What peculiarities can you observe? What conclusions do you draw from your findings?

It depicts that these layers have same weight for each and every connection. All in all, it can be said that model share the same weight from layer 1 to layer 2, it means the same feature map would be generated for this 2 layers.

3. (3 Points) Execute one step of the backpropagation algorithm (all the way back to the input) and update the weights accordingly. Use (0 - 0 - 1) as the expected output and 0.5 for the learning rate. Manual calculation is very time consuming, consider a programming approach or the use of tools.

We were unable to solve the task but we know the approach which is mention in next question.

4. (2 Points) How did you solve the previous sub-task? What were your reasons to choose this approach above others? If you didn't solve the previous sub-task, instead, name the technique you would have used and explain briefly why you would choose this specific option in favor of others.

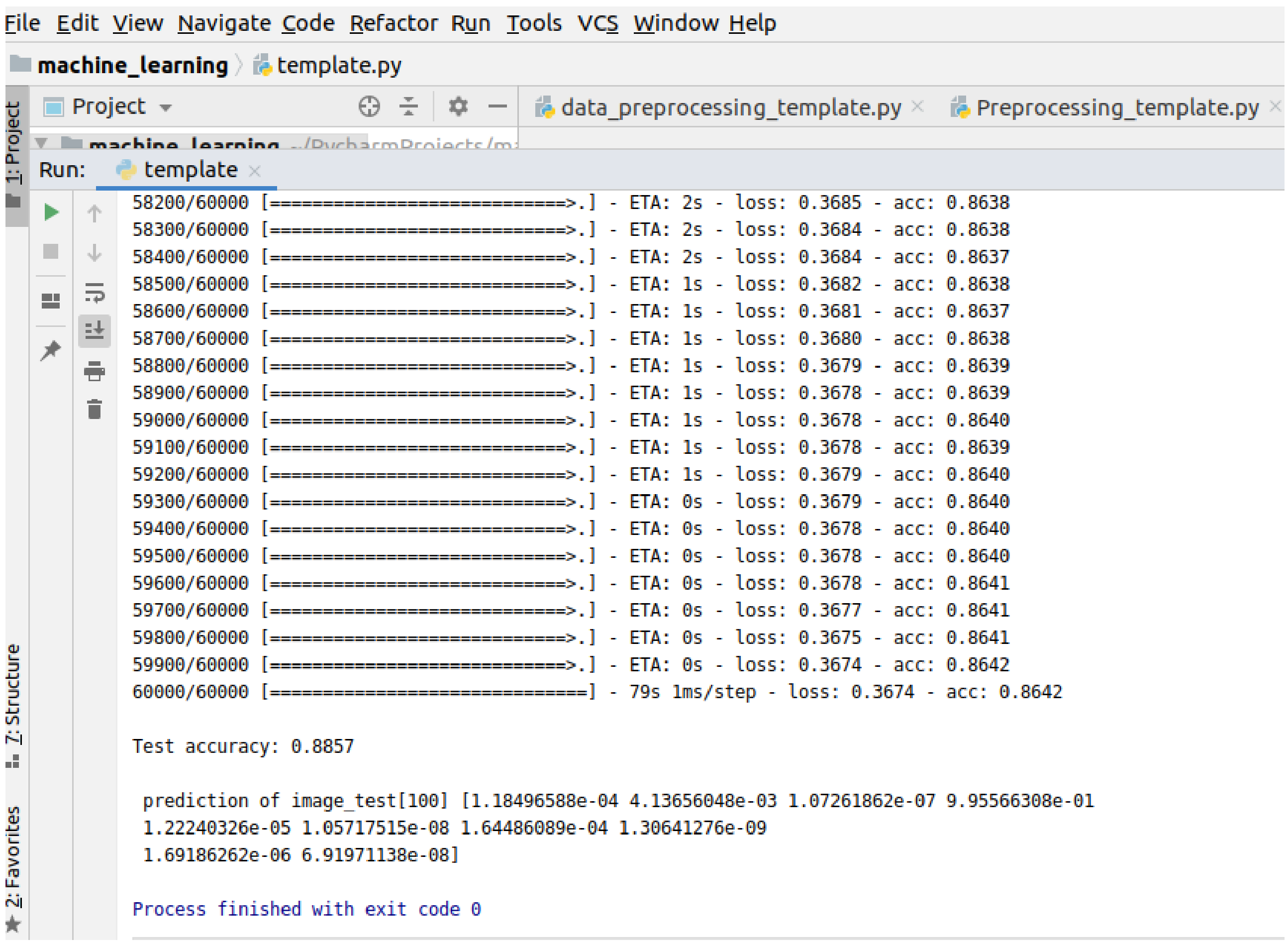
Backpropagation works by using a loss function to calculate how far the network was from the target output. We can calculate the error by using mean sum squared loss function.

Loss = $\Sigma(0.5)(\text{predicted output - actual output})^2$

we have to find that in which direction to alter our weights, we need to find the rate of change of our loss with respect to our weights. Which we can do by gradient descent method.

3 Deep Learning (8 Points)

1. (4 Points) Implement the network architecture as stated above. You may use the deep learning template provided in the VC course in the “Material“ section. This template takes away the work of loading the fashion mnist data set as well as prepossessing it to be used with a Keras architecture. It also provides optional features that might help you gaining a better understanding of the dataset and possible ways to visualize it. Train the network with the training dataset and test it with the test dataset. Provide a screenshot of your implemented code, the accuracy and the loss results of your validation data.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
machine_learning
template.py
Project
data_preprocessing_template.py
Preprocessing_template.py
Run: template
1: Project
7: Structure
2: Favorites

```
58200/60000 [============================>.] - ETA: 2s - loss: 0.3685 - acc: 0.8638
58300/60000 [============================>.] - ETA: 2s - loss: 0.3684 - acc: 0.8638
58400/60000 [============================>.] - ETA: 2s - loss: 0.3684 - acc: 0.8637
58500/60000 [============================>.] - ETA: 1s - loss: 0.3682 - acc: 0.8638
58600/60000 [============================>.] - ETA: 1s - loss: 0.3681 - acc: 0.8637
58700/60000 [============================>.] - ETA: 1s - loss: 0.3680 - acc: 0.8638
58800/60000 [============================>.] - ETA: 1s - loss: 0.3679 - acc: 0.8639
58900/60000 [============================>.] - ETA: 1s - loss: 0.3678 - acc: 0.8639
59000/60000 [============================>.] - ETA: 1s - loss: 0.3678 - acc: 0.8640
59100/60000 [============================>.] - ETA: 1s - loss: 0.3678 - acc: 0.8639
59200/60000 [============================>.] - ETA: 1s - loss: 0.3679 - acc: 0.8640
59300/60000 [============================>.] - ETA: 0s - loss: 0.3679 - acc: 0.8640
59400/60000 [============================>.] - ETA: 0s - loss: 0.3678 - acc: 0.8640
59500/60000 [============================>.] - ETA: 0s - loss: 0.3678 - acc: 0.8640
59600/60000 [============================>.] - ETA: 0s - loss: 0.3678 - acc: 0.8641
59700/60000 [============================>.] - ETA: 0s - loss: 0.3677 - acc: 0.8641
59800/60000 [============================>.] - ETA: 0s - loss: 0.3675 - acc: 0.8641
59900/60000 [============================>.] - ETA: 0s - loss: 0.3674 - acc: 0.8642
60000/60000 [==============================] - 79s 1ms/step - loss: 0.3674 - acc: 0.8642

Test accuracy: 0.8857

 prediction of image_test[100] [1.18496588e-04 4.13656048e-03 1.07261862e-07 9.95566308e-01
 1.22240326e-05 1.05717515e-08 1.64486089e-04 1.30641276e-09
 1.69186262e-06 6.91971138e-08]

Process finished with exit code 0
```

2. (1 Point) Please answer the following questions in one short sentence each:

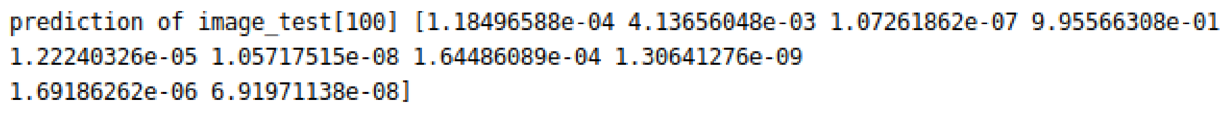
(a) What makes a dropout-layer different from ”normal” convolution layers and why do we use them?

Dropout-layer only drops out a random set of activations in particular layer by setting them to 0. Due to that network becomes more redundant. It also prevent over fitting problem. Moreover, this layer is only used during training, and not during testing.

(b) Describe what a flatten-layer does?

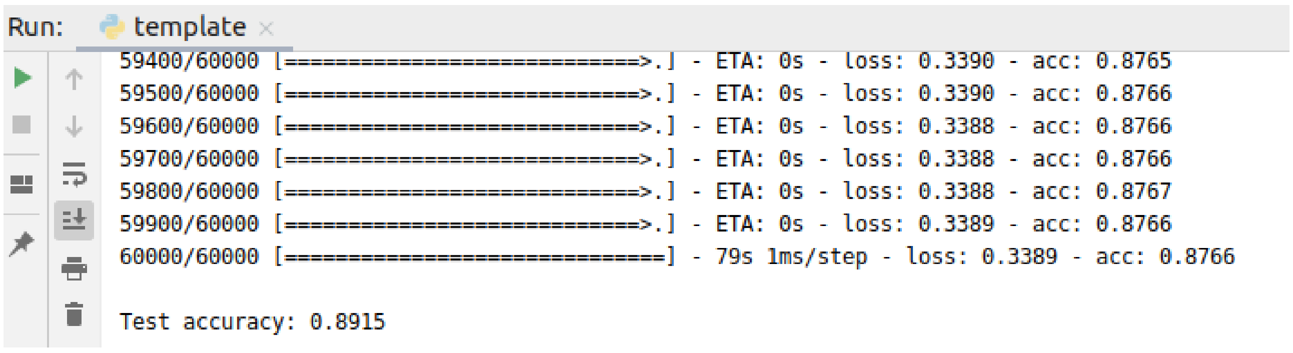
In between the convolution layer and the fully connected layer, there is a 'Flatten' layer. Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classification

3. (1 Point) You have now successfully trained your model. Make a prediction for a single image stored in the test-set-array at position 100 (images test[100]). You may have to reshape the data of this image in order to allow your model to process it correctly. What does the model predict and is this prediction correct? Add your commented code to the script you later submit as solution.



```
prediction of image_test[100] [1.18496588e-04 4.13656048e-03 1.07261862e-07 9.95566308e-01
 1.22240326e-05 1.05717515e-08 1.64486089e-04 1.30641276e-09
 1.69186262e-06 6.91971138e-08]
```

4. (2 Points) The accuracy value with the given instructions should be somewhat close to 0.87. We can actually archive way better results just by tweaking the hyper parameters batch size, epochs and learning rate. Try to adjust these hyper parameters to reach an accuracy of 0.90 or higher. Please state the hyper parameter configurations you used and implement them in script that you submit as solution.



```
59400/60000 [============================>.] - ETA: 0s - loss: 0.3390 - acc: 0.8765
59500/60000 [============================>.] - ETA: 0s - loss: 0.3390 - acc: 0.8766
59600/60000 [============================>.] - ETA: 0s - loss: 0.3388 - acc: 0.8766
59700/60000 [============================>.] - ETA: 0s - loss: 0.3388 - acc: 0.8766
59800/60000 [============================>.] - ETA: 0s - loss: 0.3388 - acc: 0.8767
59900/60000 [============================>.] - ETA: 0s - loss: 0.3389 - acc: 0.8766
60000/60000 [==============================] - 79s 1ms/step - loss: 0.3389 - acc: 0.8766

Test accuracy: 0.8915
```