Lab Course Machine Learning
Exercise Sheet 8

Mofassir ul Islam Arif
Information Systems and Machine Learning Lab
University of Hildesheim

Submission deadline: Friday Jan 10, 23:59PM (on LearnWeb, course code: 3115)

Instructions

Please following these instructions for solving and submitting the exercise sheet.

1. You should submit two things a) python scripts(zipped) / jupyter notebook and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.
4. Unless explicitly mentioned, you are not allowed to use scikit, sklearn or any other library for solve any part. All implementations must be done yourself.

Plagiarism: Submitting plagiarized work will not be tolerated in this lab by any means.

PyTorch in Research

In this lab, we will be starting with non-toy example implementations of neural networks. This is an extended lab, you have two weeks to implement this lab.

Research Paper Implementation (20 points)

[NOTE: Please note that the authors used AS MANY CONV layers in parallel as the number of channels. Meaning each channel was passed through its own conv layers. The concatenation of the outputs of each channel is then passed to the MLP, as shown in the architecture diagarm. The figure you see is only an example where it is assumed that the data under consideration only has three channels. Your final architecture will have more than three channels.]

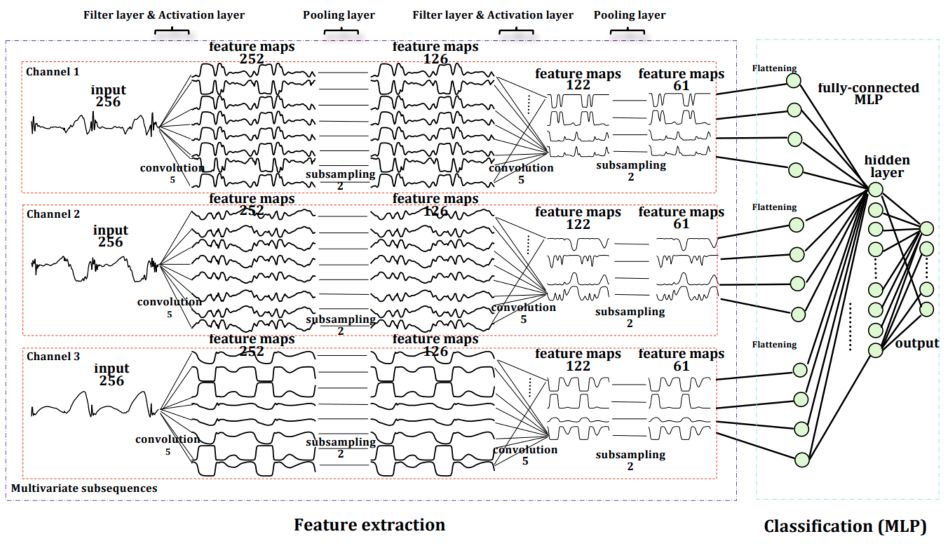
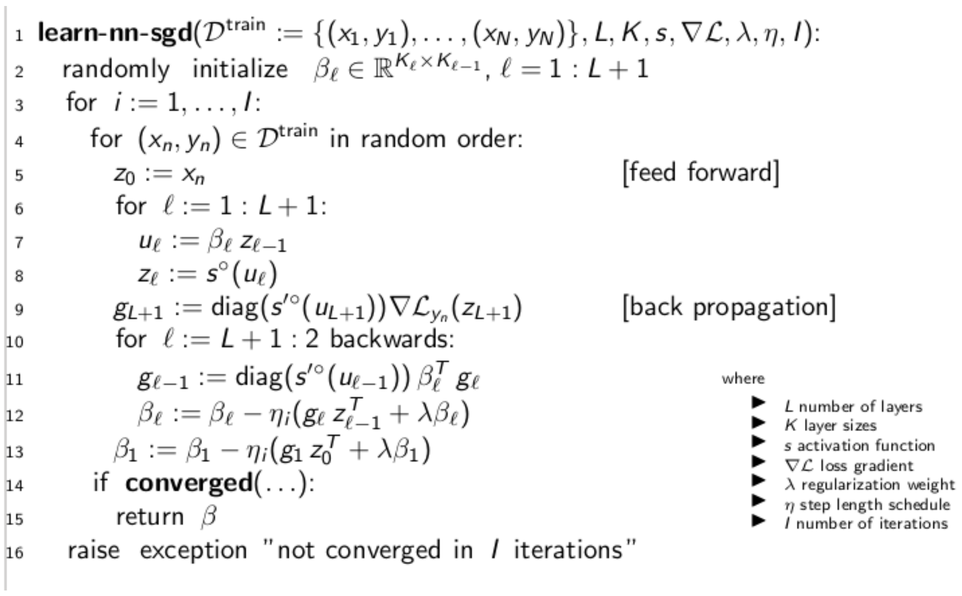


Figure 1: Architecture

Data: The paper talks about using the PAMPA2 dataset, you can find the dataset at `http://archive.ics.uci.edu/ml/datasets/pamap2+physical+activity+monitoring`. The authors have talked about their choice of a subset of the data and the preprocessing steps that they have applied. Treat the raw data in a similar manner. **Please Show this both your report and the code.**

- Implement the model as described by the authors using PyTorch.
- I highly recommend using the dataset and dataloader methods offered by PyTorch as they will make your code really streamlined
- If you have made ANY changes to the author's approach, mention it and state why you chose differently.
- Include tensorboards in your report. Include, gradient information, histograms, loss, and accuracies. [Tutorial can be found in Annex]
- You should be able to reproduce the results of the author.
- You are NOT allowed to use Keras for any purpose, using keras will not get your any credits.
- Present your results and compare them against the author's, if you have not been able to reproduce the results, investigate and report what caused this.

Bonus (5 Points)



```
learn-nn-sgd($\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_N, y_N)\}, L, K, s, \nabla\mathcal{L}, \lambda, \eta, I$):
  randomly initialize $\beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$, $\ell = 1 : L+1$
  for $i := 1, \ldots, I$:
    for $(x_n, y_n) \in \mathcal{D}^{\text{train}}$ in random order:
      $z_0 := x_n$                                        [feed forward]
      for $\ell := 1 : L+1$:
        $u_\ell := \beta_\ell \, z_{\ell-1}$
        $z_\ell := s^\circ(u_\ell)$
      $g_{L+1} := \text{diag}(s'^\circ(u_{L+1}))\nabla\mathcal{L}_{y_n}(z_{L+1})$        [back propagation]
      for $\ell := L+1 : 2$ backwards:
        $g_{\ell-1} := \text{diag}(s'^\circ(u_{\ell-1}))\, \beta_\ell^T \, g_\ell$
        $\beta_\ell := \beta_\ell - \eta_i(g_\ell \, z_{\ell-1}^T + \lambda\beta_\ell)$
      $\beta_1 := \beta_1 - \eta_i(g_1 \, z_0^T + \lambda\beta_1)$
    if converged(...):
      return $\beta$
  raise exception "not converged in $I$ iterations"
```

where
- L number of layers
- K layer sizes
- s activation function
- $\nabla\mathcal{L}$ loss gradient
- λ regularization weight
- η step length schedule
- I number of iterations

Figure 2: Algorithm, Neural Network

Algorithm shown here shows the manual implementation of the a Neural network.

- Implement the algorithm for a neural network with 2 Hidden layers and 4 neurons in each hidden layer.
- Generate moons dataset using sklearn and train your NN. The choice of parameters for the dataset are up to you. Use 2 classes.
- Compare your model with the same network implemented in PyTorch and comment of the difference you see. (speed, accuracy, loss)

Annex

1. CNNs: `http://cs231n.github.io/convolutional-networks/`
2. CNNs: `http://cs231n.github.io/convolutional-networks-1/`
3. Data preprocess `http://cs231n.github.io/neural-networks-2/`
4. `http://cs231n.github.io/neural-networks-3/`
5. CNN Lecture 1 `https://www.youtube.com/watch?v=bNb2fEVKeEo&t=833s&list=PL3FW7Lu3i5JvHM8ljYj-zLfQR`
   `index=6`
6. CNN Lecture 2 mini-batch `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.`
   `pdf`
7. Tensorboards in PyTorch `https://pytorch.org/docs/stable/tensorboard.html`