RDFIA: Pattern recognition for image analysis and interpretation

TP 3 & 4 Report

Kevin Meetooa

Année 2020-2021

Contents

0.1 Introduction

In this report, we present the work done and the results obtained during the TPs 3 and 4.
We first gave a formal definition of a Neural Network, expliciting both its forward and backward passes. We then calculated the gradients with respect to several parameters to get a better understanding of the backpropagation.
After the theorical part, we coded and implemented the neural network.

0.2 Dataset

1.
   - The training set is used to fit the model and its weights.
   - Once the model has been fit on the training set, the validation set is used to tune the model's hyperparameters.
   - Once the model's hyperparameters have been tuned on the validation set, the test set is used to evaluate the performance of the model.
2. A higher value of N means there are more examples in the training set. A higher value of N usually leads to a better chance of getting a model that is able to generalize on larger test sets as long as the training set isn't too imbalanced.

0.3 Neural Network Architecture

3. Activation functions allow us to add some "non-linearity" into the network, this can be better for capturing complex patterns. Without activation functions, our neural network would only be a composition of linear functions, rendering our model unable to treat non-linearly separable problems.
4.
   - $n_x = 2$. This corresponds to the input dimension
   - $n_y = 2$. This corresponds to the number of classes in our problem
   - $n_h = 4$. This is an hyperparameter
5. $\hat{y}$ is a prediction: This is the output of our neural network given an input x. For our classification problem, $\hat{y}$ will be the predicted class for x.

   y is the "ground truth": This is the class that we want our neural network to predict.
6. The softmax function allows us to assign a probability to each of the output classes. We can then use this output to choose the class with the highest probability.
7. $\tilde{h} = W_h x + b_h$

   $h = tanh(\tilde{h})$

   $\tilde{y} = W_y h + b_y$

   $y = Softmax(\tilde{y})$
8. In both cases, $\tilde{y}$ must get closer to y to minimize the loss function.
9. MSE is best suited for regression problems because of its sensitiveness to outliers, giving a high penalty to those extreme values.

   CrossEntropy is best suited for classification because it allows us to compute a "distance" between two probability distributions.

0.4 Learning Method

10. - Classical Gradient Descent: This version converges faster (in terms of number of iterations) than the other versions but it's time of execution may be too long for larger datasets.
    - Stochastic Gradient Descent: This version has a much faster execution time but usually converges slower than classical gradient descent.
    - Mini-batch Gradient Descent: This is a good compromise between classical and stochastic gradient descent

    Mini-batch gradient descent seems to be the best choice in the general case. The convergence is usually faster than stochastic gradient descent and the execution time is faster than classical gradient descent.

11. A too small learning rate may lead to a slow convergence whereas a too high learning rate can either lead to a non-optimal solution or even a divergent behaviour.

12. The naive approach would be applying the chain rule on all of the variables used for computing the loss function's gradient. However, using this solution, the same derivatives would be computed several times.

    The backpropagation approach would be computing all the variables' derivatives first and then storing them. We could then apply the chain rule by looking up for the derivatives in the stored table.

    The latter solution is much more computationally efficient.

13. For the backpropagation to work, the loss function must be differentiable.

14. $l(y_i, \hat{y_i}) = -\sum_i y_i log(\hat{y_i})$

    $= -\sum_i y_i log(\frac{e^{\tilde{y_i}}}{\sum_j e^{\tilde{y_j}}})$ (application of the softmax formula)

    $= -(\sum_i y_i \tilde{y_i} - \sum_i y_i log(\sum_j e^{\tilde{y_j}}))$

    $= -\sum_i y_i \tilde{y_i} + log(\sum_i e^{\tilde{y_i}})$ since $\sum_i y_i = 1$ (y is obtained from a softmax)

15. $\frac{\partial l}{\partial \tilde{y_i}} = \frac{\partial}{\partial \tilde{y_i}}(-\sum_i y_i \tilde{y_i} + log(\sum_i e^{\tilde{y_i}}))$

    $= -y_i + \frac{e^{\tilde{y_i}}}{\sum_i e^{\tilde{y_i}}}$ since $log(f)' = \frac{f'}{f}$

    $= -y_i + \hat{y_i} = (\hat{y_i} - y_i)$

    Therefore, $\begin{bmatrix} \frac{\partial l}{\partial \tilde{y_1}} \\ \frac{\partial l}{\partial \tilde{y_2}} \\ \vdots \\ \frac{\partial l}{\partial \tilde{y_{n_y}}} \end{bmatrix} = \begin{bmatrix} \hat{y_1} - y_1 \\ \hat{y_2} - y_2 \\ \vdots \\ \hat{y_{n_y}} - y_{n_y} \end{bmatrix}$

16. $\frac{\partial l}{\partial W_{y,ij}} = \sum_k \frac{\partial l}{\partial \tilde{y_k}} \frac{\partial \tilde{y_k}}{\partial W_{y,ij}}$

    With $\tilde{y} = W_y h + b_y$

    Therefore, $\frac{\partial \tilde{y_k}}{\partial W_{y,ij}} = \frac{\partial}{\partial W_{y,ij}}(\sum_j W_{y,ij} h_j + b_y) = h_j$

    And since, $\frac{\partial l}{\partial \tilde{y_k}} = \hat{y_k} - y_k$

    Then $\frac{\partial l}{\partial W_{y,ij}} = \sum_k (\hat{y_k} - y_k) h_j = h_j \sum_k (\hat{y_k} - y_k)$

17.

- $(\nabla_{\tilde{h}} l)_i = \frac{\partial l}{\partial \tilde{h_i}} = \sum_k \frac{\partial l}{\partial \tilde{y_k}} \frac{\partial \tilde{y_k}}{\partial h_i} \frac{\partial h_i}{\partial \tilde{h_i}}$
  $= \sum_k (\hat{y_k} - y_k) W_{y,ki} (1 - h_i^2)$ since $\frac{\partial \tilde{y_k}}{\partial h_i} = W_{y,ki}$ and $h_i = tanh(\tilde{h_i})$
  $= (1 - h_i^2) \sum_k (\hat{y_k} - y_k) W_{y,ki}$
- $(\nabla_{W_h} l)_{i,j} = \frac{\partial l}{\partial W_{h,ij}} = \frac{\partial l}{\partial \tilde{h_i}} \frac{\partial \tilde{h_i}}{\partial W_{h,ij}}$
  With $\tilde{h} = W_h x + b_h \Rightarrow \frac{\partial \tilde{h_i}}{\partial W_{h,ij}} = x_j$
  Therefore, $\frac{\partial l}{\partial W_{h,ij}} = ((1 - h_i^2) \sum_k (\hat{y_k} - y_k) W_{y,ki}) x_j$
- $(\nabla_{b_h} l)_i = \frac{\partial l}{\partial b_{h_i}} = \frac{\partial l}{\partial \tilde{h_i}} \frac{\partial \tilde{h_i}}{\partial b_{h_i}}$
  With $\tilde{h} = W_h x + b_h \Rightarrow \frac{\partial \tilde{h_i}}{\partial b_{h_i}} = 1$
  Therefore, $\frac{\partial l}{\partial b_{h_i}} = \frac{\partial l}{\partial \tilde{h_i}} = (1 - h_i^2) \sum_k (\hat{y_k} - y_k) W_{y,ki}$