Rdfia - Neural Network

Missoum Youcef, Bourzag Mohamed Chakib

October 2025

Solution

1 Introduction

This document contains exercises Solution of Pdf-tp1

Exercise 1. *solutions*

1. *What are the train, val, and test sets used for?*

   Solution.

   - ***Train set:*** *The subset of data used to fit the model. The model learns the relationship between features and targets from this portion.*
   - ***Validation set:*** *A separate subset used during training to evaluate the model on unseen data. It helps tune hyperparameters and make decisions such as early stopping to prevent overfitting.*
   - ***Test set:*** *A completely unseen labeled subset used only after training is finished. It measures the model's final generalization performance on new, real-world data.*

   □

2. *What is the influence of the number of examples N?*

   Solution.

   - ***Larger*** N***:*** *A larger dataset exposes the model to a wider variety of examples, reducing the risk of overfitting. This generally improves performance on validation and test sets.*
   - ***Smaller*** N***:*** *With too few examples, a model can easily overfit, achieving high training accuracy but poor generalization to unseen data.*

   □

3. *Why is it important to add activation functions between linear transformations?*

   Solution. *to add no linearity to learn complex pattern* □

4. *What are the sizes n_x, n_h, n_y in Figure 1? In practice, how are these sizes chosen?*

   Solution.

   - n_x ***(input size):*** n_x*=2, chosen by the dimensionality of the input features (e.g., two input nodes x_1 and x_2 in Figure 1).*
   - n_h ***(hidden size):*** n_h*=4, A user-chosen hyperparameter that depends on the problem's complexity and is typically tuned through experimentation.*
   - n_y ***(output size):*** n_y*=2, Determined by the number of classes in the classification task (e.g., two output nodes $\hat{y}_1$ and $\hat{y}_2$ in Figure 1).*

   □

5. *What do the vectors $\hat{y}$ and y represent? What is the difference between these two quantities?*

   Solution.

   - $\hat{\boldsymbol{y}}$*: The predicted labels produced by the model.*
   - $\boldsymbol{y}$*: The true (ground-truth) labels from the dataset.*
   - *the diffrence between them is to mesure how the predicted is close to the true label.*

□

6. *Why use a SoftMax function as the output activation function?*

   Solution.

   - *It provides a probability distribution across classes, indicating how confident the model is about each class.*
   - *It converts raw output scores (logits) into probabilities where each value is between 0 and 1 and all values sum to 1.*
   - *This makes it ideal for multi-class classification, as each output can be directly interpreted as the model's confidence for that class.*

□

7. *Write the mathematical equations allowing to perform the forward pass of the neural network, i.e. allowing to successively produce $\tilde{h}$, h, $\tilde{y}$ and $\hat{y}$ starting at x.*

   Solution.

$$\begin{aligned}\tilde{h} &= xW_h^\top + b_h \\ h &= \tanh(\tilde{h}) \\ \tilde{y} &= hW_y^\top + b_y \\ \hat{y} &= SoftMax(\tilde{y})\end{aligned}$$

□

8. *During training, we try to minimize the loss function. For cross-entropy and squared error, how must $\hat{y}_i$ vary to decrease the global loss function L?*

   Solution.

   - ***Cross-entropy:*** *For the correct class, $\hat{y}_i$ should be as close to* 1 *as possible, and for all incorrect classes, it should be close to* 0.
   - ***Squared error:*** *The prediction $\hat{y}_i$ should be as close as possible to the true label y_i.*

□

9. *How are these functions better suited to classification or regression tasks?*

   Solution.

   - ***Cross-entropy (for classification):*** *Measures the difference between the predicted probability distribution $\hat{y}$ and the true labels y, making it well-suited for classification tasks.*
   - ***Squared error (for regression):*** *The loss $\sum_i (\hat{y}_i - y_i)^2$ ensures that the predicted values $\hat{y}_i$ are close to the true labels y_i, making it appropriate for regression tasks.*

□

10. *What seem to be the advantages and disadvantages of the various variants of gradient descent: classic, mini-batch stochastic, and online stochastic? Which one seems the most reasonable to use in the general case?*

    Solution.

    - ***Classic (Batch) Gradient Descent***
      - ***Advantage:*** *Provides a stable, direct convergence path towards the minimum.*
      - ***Disadvantage:*** *Highly demanding in terms of computation and memory, as it processes the entire dataset for each update. This results in significantly slow update times*
    - ***Online Stochastic Gradient Descent (SGD)***
      - ***Advantage:*** *Very fast updates (one per sample) with low memory usage. Noisy updates can help escape poor local minima.*
      - ***Disadvantage:*** *High variance in updates creates a noisy and unstable convergence path, sometimes preventing convergence to the exact minimum.*
    - ***Mini-Batch Stochastic Gradient Descent (SGD)***
      - ***Advantage:*** *The most widely used approach, balancing stability and computational efficiency. More efficient than classic GD and more stable than online SGD.*

- ***Disadvantage:*** *Introduces an extra hyperparameter: the batch size.*

- *Which is most reasonable?*
  Mini-batch SGD is the most reasonable and widely used method in practice because it offers the best compromise between computational efficiency and convergence stability.

□

11. *What is the influence of the learning rate η on learning?*

Solution.

- ***Too high*** η***:*** *The optimization may overshoot the minimum and fail to converge.*
- ***Too low*** η***:*** *Very slow Convergence, and the model may get stuck in a poor local minimum.*

□

12. *Compare the complexity (depending on the number of layers) of calculating the gradients of the loss with respect to the parameters, using the naive approach and the backpropagation algorithm.*

Solution.

- ***Naive approach:***
  - *Each gradient $\frac{\partial \mathcal{L}}{\partial W^{(l)}}$ is computed independently by applying the chain rule from the output back to layer l.*
  - *Requires recomputing many intermediate derivatives repeatedly.*
  - *Computational complexity: $\mathcal{O}(L^2)$, because for each of the L layers, we traverse back through all previous layers.*
- ***Backpropagation algorithm:***
  - *Reuses intermediate results (forward activations and backward error terms) to compute all gradients efficiently in a single backward pass.*
  - *Computational complexity: $\mathcal{O}(L)$, since each layer's gradients are computed once using the stored activations and errors.*

□

13. *What criteria must the network architecture meet to allow such an optimization procedure?*

Solution.

- *To enable gradient-based optimization methods such as backpropagation, every component of the neural network must be differentiable.*
- *This includes the loss function, activation functions, and transformations such as affine layers—all of which must have well-defined gradients with respect to their inputs and parameters.*
- *Differentiability ensures that the chain rule can be effectively applied to propagate the loss gradient backward through the network, reaching all the way to the parameters of the initial layer.*

□

14. *The function SoftMax and the cross-entropy loss are often used together and their gradient is very simple. Show that the loss can be simplified by:*

$$\mathcal{L} = -\sum_i y_i \tilde{y}_i + \log \sum_i e^{\tilde{y}_i}.$$

Solution.

$$\textit{Start with cross-entropy:} \quad \mathcal{L} = -\sum_i y_i \log(\hat{y}_i)$$

$$\textit{Softmax definition:} \quad \hat{y}_i = \frac{e^{\tilde{y}_i}}{\sum_j e^{\tilde{y}_j}}$$

$$\textit{Substitute into the loss:} \quad \mathcal{L} = -\sum_i y_i \log\left(\frac{e^{\tilde{y}_i}}{\sum_j e^{\tilde{y}_j}}\right)$$

$$\textit{Simplify the log term:} \quad \mathcal{L} = -\sum_i y_i \left(\tilde{y}_i - \log \sum_j e^{\tilde{y}_j}\right)$$

$$\textit{Expand and simplify:} \quad \mathcal{L} = -\sum_i y_i \tilde{y}_i + \log \sum_j e^{\tilde{y}_j}$$

□

15. *Write the gradient of the loss (cross-entropy) relative to the intermediate output $\tilde{y}$:*

Solution.

$$\nabla_{\tilde{y}}\mathcal{L} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \tilde{y}_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \tilde{y}_{n_y}} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 - y_1 \\ \vdots \\ \hat{y}_{n_y} - y_{n_y} \end{bmatrix}$$

□

16. *Using backpropagation, write the gradient of the loss with respect to the weights of the output layer $\nabla W_y\mathcal{L}$. Do the same for $\nabla b_y\mathcal{L}$.*

Solution.

$$\tilde{y} = Wx + b \quad with \quad W \in R^{n_y \times n_h},\ x \in R^{n_h},\ b \in R^{n_y}$$

$$\delta \nabla_{\tilde{y}}\mathcal{L} = \hat{y} - y$$

$$\nabla_W \mathcal{L} \ = \ \delta\, x^\top$$

$$\nabla_b \mathcal{L} \ = \ \delta$$

(For a mini-batch of size m): $$\nabla_W \mathcal{L} \ = \ \frac{1}{m}\sum_{k=1}^{m} \delta^{(k)} x^{(k)\top}, \qquad \nabla_b \mathcal{L} \ = \ \frac{1}{m}\sum_{k=1}^{m} \delta^{(k)}$$

□

17. *Compute other gradients: $\nabla \tilde{h}\mathcal{L}$, $\nabla W_h\mathcal{L}$, $\nabla b_h\mathcal{L}$.*

Solution. *Let $\delta^{\tilde{y}} = \hat{y} - y$ be the error gradient at the output layer's input.*

First, we backpropagate the error to the hidden layer's output h:

$$\nabla_h \mathcal{L} = \frac{\partial \mathcal{L}}{\partial h} = \frac{\partial \mathcal{L}}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial h} = \delta^{\tilde{y}} W_y$$

Next, we backpropagate through the tanh *activation function to find the error at the hidden layer's input, $\tilde{h}$. Let's denote this error as $\delta^{\tilde{h}}$:*

$$\delta^{\tilde{h}} \nabla_{\tilde{h}}\mathcal{L} = \frac{\partial \mathcal{L}}{\partial h} \frac{\partial h}{\partial \tilde{h}} = (\delta^{\tilde{y}} W_y) \odot (1 - h^2)$$

where $\odot$ is the element-wise (Hadamard) product.

Finally, we can compute the gradients for the hidden layer's weights W_h and bias b_h:

$$\nabla_{W_h}\mathcal{L} = (\delta^{\tilde{h}})^\top x$$

$$\nabla_{b_h}\mathcal{L} = \delta^{\tilde{h}}$$

For a mini-batch of size m:

$$\nabla_{W_h}\mathcal{L} = \frac{1}{m}\sum_{k=1}^{m} (\delta^{\tilde{h}(k)})^\top x^{(k)}$$

$$\nabla_{b_h}\mathcal{L} = \frac{1}{m}\sum_{k=1}^{m} \delta^{\tilde{h}(k)}$$

□