Universität des Saarlandes
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNTI Winter Term 2023/2024

Exercise Sheet 8

Neural Networks Basics

Deadline: 11.01.2024 23:59

Guidelines: You are expected to work in a group of 2-3 students. While submitting the assignments, please make sure to include the following information for all our teammates in your PDF/python script:

Name:

Student ID (matriculation number):

Email:

Your submissions should be zipped as **Name1_id1_Name2_id2_Name3_id3.zip** when you have multiple files. For assignments where you are submitting a single file, use the **same naming convention** without creating a zip. For any clarification, please reach out to us on the **CMS Forum**.
Note that the above instructions are mandatory. If you are not following them, tutors can decide not to correct your exercise.

Exercise 8.1 - Universal Approximation Theorem (0.5 + 0.25 + 0.25 + 0.25 points)

Read section 6.4.1 in the Deep Learning book and answer the following questions:

a) What does the universal approximation theorem state? (1 sentence)

b) Does the result hold for any arbitrary activation function? (1 sentence)

c) Given the universal approximation properties of feed forward neural networks, why do we still:

   (i) care about designing neural network architectures different from fully connected feed forward networks? (2 reasons suffices)

   (ii) prefer to use (often many) more than two layers in practice? (2 reasons suffice)

Exercise 8.2 - Regularization (1.0+0.5+1.0+0.5+0.25+0.25 points)

a) Consider a linear regression model

$$\hat{y}(\boldsymbol{x}, \boldsymbol{w}) = w_0 + \sum_{i=1}^{m} w_i x_i$$

and the sum-of-squares loss function

$$J(\boldsymbol{w}; \{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{y}) = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}(\boldsymbol{x}^{(i)}, \boldsymbol{w}))^2$$

Consider adding a noise term $\boldsymbol{\epsilon}_i \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ to each data sample $\boldsymbol{x}_i$ in the training set. Note that, because the noise terms are *i.i.d.*, $\mathbb{E}[\boldsymbol{\epsilon}_i] = \mathbf{0}$ and $\mathbb{E}[\boldsymbol{\epsilon}_i\boldsymbol{\epsilon}_i^T] = \sigma^2\mathbf{I}$. Show that minimizing $\mathbb{E}_\epsilon[J(\boldsymbol{w}; \{\boldsymbol{x}_i+\boldsymbol{\epsilon}_i\}_{i=1}^m, \boldsymbol{y})]$ is equivalent to minimizing the sum-of-squares loss function for noise-free input variables with the addition of L_2-regularization.

b) Consider the regularized objective

$$\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}.$$

For SGD, the weight update for the error $J(\boldsymbol{w})$ is

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\nabla_{\boldsymbol{w}_t}J(\boldsymbol{w}_t)$$

where η is the learning rate. Derive the weight update rule for the regularized loss $\tilde{J}(\boldsymbol{w})$. How is it related to weight decay?

c) For a linear model with L_2 regularization (ridge regression), the optimal weights are given by

$$\boldsymbol{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{X}$ is the data matrix and $\boldsymbol{y}$ is the vector of targets. Show that for predictions made using this model, the following holds:

$$\mathbf{X}\boldsymbol{w}^* = \sum_{i=1}^{n}\boldsymbol{u}_i\frac{\sigma_i^2}{\sigma_i^2+\lambda}\boldsymbol{u}_i^T\boldsymbol{y},$$

where $\mathbf{X} \in \mathbb{R}^{m\times n}$ is the data matrix, $\boldsymbol{y} \in \mathbb{R}^n$ is the vector of targets, $\boldsymbol{u}_i \in \mathbb{R}^n$. Here $\boldsymbol{u}_i$ and σ_i^2 are the eigenvectors and eigenvalues of $\mathbf{X}^T\mathbf{X}$. This shows that for a linear model (and centered data), L_2-regularization shrinks the predictions along the directions of lower variance.

d) Read section *7.1 Parameter Norm Penalties* in the Deep Learning Book. Then answer the following question in 3-4 sentences: Why do we only regularize weights but not the bias/intercept?

e) In a linear model, what problems can arise when the number of features n is larger than the number of samples m? Can regularization help in this case and if so why? Briefly answer in 2-3 sentences.

f) Regularization is defined as any modification that we make to a learning algorithm that is intended to reduce its generalization error as well as its training error. Is this true? Motivate your answer.

Exercise 8.3 - Data Augmentation (0.5+0.25 points)

a) A naive way of implementing data augmentation for an image classification task is shown in the following code snippet.

```python
def augment_data(input_images, labels, num_augmentations):
  augmented_dataset = []

  for image, label in zip(input_images, labels):
      for i in range(num_augmentations):
          # Apply augmentation technique
          augmented_image = apply_augmentation(image)
          augmented_dataset.append((augmented_image, label))

  # Ensure augmented and original images are mixed
  shuffle(augmented_dataset)
  return augmented_dataset

```

It creates a new dataset by creating a fixed number of augmented images from each original image. These comprise the new training set that would be used to train the model. What are some disadvantages of this approach? Can you find a better way to apply data augmentation? (4-5 sentences)
Hint: Think about why PyTorch's `Dataset` class offers the `transform` parameter.

b) When using data augmentation, we clearly apply them to the training set. However, when it comes to the validation- and test set, the answer is not as clear. What are some arguments for and against applying data augmentation to the validation- and test set? (3-4 sentences) Hint: Consider both benchmark test sets as well as real-world test sets.

Exercise 8.4 - Regularization & Agumentation (3.5 points)

See notebook.

Exercise 8.5 - Hyperparameter search (1 points)

In exercise 8.4 you have seen one hyperparameter configuration of the model. To find out if there are better values for the learning rate α and the weight decay λ, you will implement a *grid search* over possible combinations. In a *grid search* we test all combinations, i.e. if $\alpha \in \mathcal{A}$ and $\lambda \in \mathcal{B}$, the model is trained for all parameter combinations $(\alpha, \lambda) \in \mathcal{A} \times \mathcal{B}$. The best model is selected according to the validation loss.
Complete the `hyperparameter_search.py` file by implementing a grid search over $\alpha \in \{0.001, 0.002, 0.005, 0.01\}$ and the weight decay parameters $\lambda \in \{0.0, 0.01, 0.1, 1\}$.
Run each combination for at most 100 epochs, with a batch size of 32 and an early stopping patience of 1.
To run the hyperparameter search, copy your `assignment8.py` and `hyperparameter_search.py` to the cluster and change your submit file accordingly. **As in the previous cluster exercise, your submission is only eligible for grading if your submission can be run on the cluster.**
If you have not yet set up the cluster checkout the tutorial [1].

[1] There also have been some slight improvements to the `run.sub` file, since the last exercise sheet.