Universität des Saarlandes
Prof. Dr. Dietrich Klakow
Lehrstuhl für Signalverarbeitung
NNTI Winter Term 2024/2025

Exercise Sheet 9

Theory of Neural Networks

Deadline: 08.01.2025 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**. These instructions are mandatory. If you are not following them, tutors can decide not to correct your exercise.

Exercise 9.1 - Dropout (1+0.5+0.5+0.5+0.5+2.5 = 5.5 points)

For this exercise, looking into the following paper, which originally introduced the concept of Dropout, may be helpful.

1. Do ensemble methods like bagging always help, in terms of bias and variance? Suppose you have an ensemble that averages the answers of m ordinary linear regressions models. Assume that, because of bagging, every model "sees" approximately $\frac{2}{3}$ of the original dataset, ignoring duplicates. Compare bias and variance of the ensemble with the ones of a single linear regression model over the whole dataset. Set up a rigorous argument. You may assume a particular underlying data distribution and use the following theorem without proof:

   Theorem: Gauss-Markov: The Ordinary Least Squares (or simple Linear Regression) estimator is the estimator with the lowest variance among the class of linear unbiased estimators, under the assumption that the errors in the linear regression model are uncorrelated, have equal variances and expectation value of zero.

2. How much do you think bagging would be useful, if the ensemble was composed of neural networks? Is it feasible in practice?

3. What's the relation between bagging and the use of dropout for every hidden layer with 50% keep probability? Does dropout always help, in terms of bias and variance? Is it feasible?

4. Dropout helps avoid co-adaptations of hidden units. Discuss what co-adaptation is, why it is bad, and how (on a high level) dropout fixes it. Particularly refer to the paper for this one.

5. In dropout, during training we have a probability $1 - p$ of dropping a hidden or input neuron. During inference, we don't use dropout anymore, but now we need to scale the neuron by p. Why is that?

   There exists a variation of dropout called *inverted dropout*, which scales the neurons by $\frac{1}{p}$ only during training, leaving inference just a forward pass with no dropout. Are these two versions of dropout equivalent?

6. We now study the regularization effects of dropout on a 1-layer NN with linear output. This is linear regression with ordinary least squares! The linear regression model is given by

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

   where the bias term is absorbed into the weight vector $\boldsymbol{w}$. With dropout, the sum-of-squares loss function becomes

$$J(\boldsymbol{w}; \{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{y}) = \sum_{i=1}^{m} \left(y^{(i)} - \sum_{j=1}^{n} w_j R_{ij} x_j^{(i)} \right)^2,$$

   where the elements $R_{ij} \in \{0, 1\}$ of the dropout matrix are chosen randomly from a Bernoulli distribution with parameter p. We now take an expectation over the distribution of random dropout parameters. Show that

$$\begin{aligned} \mathbb{E}[R_{ij}] &= p \\ \mathbb{E}[R_{ij} R_{ik}] &= \delta_{jk} p + (1 - \delta_{jk}) p^2. \end{aligned}$$

   Using these results, show that the expected error for this dropout model is given by

$$\mathbb{E}_{\boldsymbol{R}}[J(\boldsymbol{w}; \{\boldsymbol{x}_i\}_{i=1}^m, \boldsymbol{y})] = \sum_{i=1}^{m} \left(y^{(i)} - p \sum_{j=1}^{n} w_j x_j^{(i)} \right)^2 + p(1-p) \sum_{j=1}^{n} w_j^2 \sum_{i=1}^{m} {x_j^{(i)}}^2.$$

   You have proved that the expected loss is a sum-of-squares loss where weights are shrunk by a factor of p, with modified L_2 norm penalty: weights for a feature whose square is usually larger are punished more.

   Now, write down a closed-form solution for the weight matrix that minimizes this regularized error function.

Exercise 9.2 - SGD (0.25+0.5+0.5+0.25 = 1.5 points)

Read the following paper and answer the following questions:

1. What's the difference between regular Gradient Descent (GD) and Stochastic Gradient Descent(SGD)? Where does the stochasticity come from?

2. What is *implicit regularization* and what is the role of the *implicit regularizer* in the context of SGD? [max 3 sentences]

3. Discuss the modified loss function and explain how does the *implicit regularizer* affects it when the learning rate is small and finite. [max 3 sentences]

4. What's another benefit of using minibatch algorithms like SGD? (Think of parallelizability)

Exercise 9.3 - Obstacles to Optimization (Bonus: 1+2 = 3 points)

1. **Ill-conditioned Hessian** (0.25+0.5+0.25 = 1 points)

   In this exercise we will discuss the challenges in the minimization of a function $f(\theta)$, $\theta \in \mathbb{R}^D$.

   a) Assume we are using Gradient optimization with a fixed learning rate α, such that the weight update is $\theta^{k+1} = \theta^k - \alpha \nabla_\theta f(\theta^k)$. Use Taylor expansion up to the second order to determine under which condition taking a gradient descent step causes an increase in the cost function f.

   b) Now assume we choose a small enough step size α' such that this cannot happen. Let the *condition number* of a matrix A be the ratio of the largest and smallest eigenvalues of A in absolute value, i.e. $\frac{\lambda_{max}}{\lambda_{min}}$ where $\lambda_{min} = \min\{|\lambda| \ \big| \ \lambda \text{ eigenvalue of } A\}$. What's the problem now, with this value of α', if the Hessian matrix H is ill-conditioned, i.e. it has a very large condition number?

   c) What optimization algorithm would you use for the problem of optimizing a strictly convex function with an ill-conditioned Hessian almost everywhere?

2. **Long-term dependencies** (1+0.25+0.25+0.5 = 2 points)

   Recommended reading: Pattern Recognition and Machine Learning, Chapter 5.3.1. It sheds light on the derivation of the fairly general NN backprop formulas we will use in the exercise. We use the same notation as the book.

   Consider having just completed a forward pass in a N-layer fully connected network with linear output and ReLU nonlinearities. We use the following notation:

   - x_j: j-th element of the input of the network
   - y_j: j-th element of the output of the network
   - t_j: j-th element of the target
   - $a_j^{(k)}$: j-th neuron of the pre-activation layer k. $k \geq 2$, $a_j^{(N+1)} = y_j$
   - $z_j^{(k)}$: j-th neuron of the post-activation layer k. $z_j^{(k)} = h(a_j^{(k)})$, where h is the activation function. $z_j^{(1)} = x_j$
   - $w_{ji}^{(k)}$: weight for $z_i^{(k)}$ to produce $a_j^{(k+1)}$: $a_j^{(k+1)} = \sum_i w_{ji}^{(k)} z_i^{(k)}$.
   - E_n: the loss function value for the single data point x. The total loss is simply $\sum_n E_n$.

   Moreover, we use

   $$\delta_j^{(k)} = \frac{\partial E_n}{\partial a_j^{(k)}}.$$

Following the derivation from the book, we arrive to the following backpropagation equations:

$$\delta_j^{(k)} = h'(a_j^{(k)}) \sum_l w_{lj}^{(k)} \delta_j^{(k+1)},$$

$$\frac{\partial E_n}{\partial w_{ji}^{(k)}} = \delta_j^{(k+1)} z_i^{(k)}.$$

(a) Assuming we have the same weights matrix W at every layer, calculate

$$\frac{\partial E_n}{\partial w_{11}^{(1)}}.$$

Hint: use matrix-vector notation and the elementwise multiplication $\odot$ when needed

(b) Now also assume $\nabla_y E_n$ is an eigenvector of W and that for this input all ReLUs fire (have argument > 0). Discuss what happens to the previous weight derivative if the corresponding eigenvalue is smaller or larger than 1, in absolute value. Does this pose any risks for optimization?

(c) Now shift back the focus over activation functions. Suppose we use a sigmoid or tanh activation functions. In a previous assignment you have proved that for all $x \in \mathbb{R}$:

$$0 < \frac{d}{dx}\sigma(x) \leq \frac{1}{4} \qquad 0 < \frac{d}{dx} tanh(x) \leq 1.$$

Now, what happens to $\frac{\partial E_n}{\partial w_{11}^{(1)}}$?

(d) Name (exactly) other two factors making neural network training challenging, briefly explaining why. [1 sentence each]

Exercise 9.4 - Nonconvex optimization (3 points)

See attached notebook.