Artificial Neural Network

1 Introduction

Learning Goals

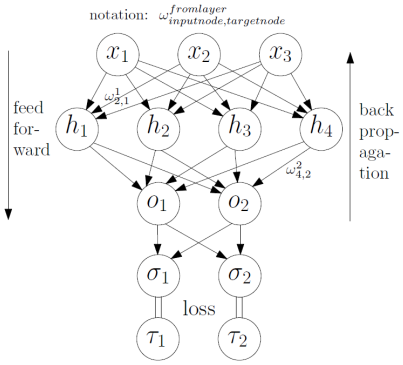
- understand the parameters of an ANN
- configure and train an ANN
- understand effects of different optimization functions and learning rates
- interpreting loss curves

2 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a fully-connected artificial neural network (ANN) consisting of at least one hidden layer.

Loss Functions for ANNs - Softmax & Cross-Entropy

- use linear activation (identity) in output layer
- apply a softmax layer after output ⇒ predict class-probabilities
- compares those to target using cross-entropy
- This is the standard setup for ANNs in (multiclass) classification



loss: cross-entropy $ce(\boldsymbol{x})$ between softmaxed outputs $\sigma(\boldsymbol{o})$ and targets (one-hot encoded) $\boldsymbol{\tau}(\boldsymbol{x})$:
$ce(\boldsymbol{x}) = -\sum_{k \in \mathcal{L}} \tau_k(x) \ln \sigma_c(\boldsymbol{o})$

Recall the principles of artificial neural networks from the lecture. Then, have a look at scikit-learn's `MLPClassifier` documentation to make yourself familiar with its usage.

3 Tasks

Design Decisions

Group work: Discuss the parameter settings of the network

Discuss the following aspects of the network and refer to the network depicted above whenever possible:

- How many hidden layers are used?
- Which activation function should be used in the hidden layer(s)?
- Which activation function should be used in the output layer?
- Which loss function is used?

Next, by using scikit-learn's `MLPClassifier` documentation, identify which parameters allow you to set

- your choice of optimizer
- the strength of L2 regularization
- the learning rate, its decay and momentum
- early stopping and its tolerance

First Experimentation

1. Train-test split the breast cancer data (`test_size=0.1`) using `random_state=42`.
2. Configure an MLP with `1` hidden layer consisting of `64` hidden nodes with `ReLU` as the hidden layer's activation function. Make sure `softmax` is set as the activation function in the output layer, alongside `cross entropy loss` as the loss function. Set a batch size of `100`.
3. Set the optimizer of the `MLPClassifier` to `SGD` and keep L2 regularization default. We want to train the network without momentum, so please make sure to turn off momentum!
4. Fit the network to the training data and evaluate the test set.
5. Print the classification report and confusion matrix for the test set.
6. Plot the training loss curve (it's stored in the attribute `loss_curve_` of a fit instance of the `MLPClassifier`).

7. Experiment with different initial learning rates (in ranges between `0.0001` and `0.1`) and plot the resulting learning curves. Try to interpret the changes in the loss curves.
8. Inspect the attribute `coef_` of the `MLPClassifier` and interpret it. What does it represent? Can you make sense of its shape?

Different Optimizers

Next, experiment with different optimizers and settings. Use an initial learning rate of `0.0001` and set the strength of L2 regularization to `0.001`. Plot the training loss curves of the following four optimizers in a single plot and try to interpret the results:

- `'SGD Baseline'`
- `'SGD with momentum'`: enable momentum and set its value to `0.9`
- `'SGD with decreasing lr'`: set the parameter that decreases learning rate over training time.
- `'Adam'`: Change optimizer from `SGD` to `Adam` with its default settings.

Early Stopping Strategy

Now we want to introduce early stopping to our model's training process, a strategy that is used to prevent overfitting. To do this, we need to make sure to dedicate a portion (scikit-learn uses 10% by default) of the data to validation. During training, this validation data is used to compute the **validation loss**. To simplify, training stops if validation loss increases. Enable early stopping for all your different optimizers, plot the training loss curves again and interpret the results.

Grid Search

Implement a grid search with 5-fold crossvalidation using `GridSearchCV` to find the best set of hyperparameters for your `MLPClassifier`. As we are focused on optimizers in this session, make sure to include both **Adam as well as SGD with momentum** in your grid. Other parameters for your grid may include:

- learning rate
- number of hidden nodes and hidden layers
- activation function in the hidden layer
- batch size

Don't forget to enable early stopping to limit training time while preventing overfitting.

4 Homework and Quiz(zes)

After you are done with this lab session, take the **Quiz 5 | Artificial Neural Networks** in Moodle, which closes **midnight before our next lab session**.
In addition, have a look at the section **Recap and Prereading** in Moodle. Within, you find articles as well as reading assignments, with two being related to **Natural Language Processing (NLP)** and **Image Processing (IP)**. Read through the articles, answer the questions in the reading assignments and, once you are done and confident in your answers, take the associated quizzes in Moodle.
Attention: As we need the knowledge contained in the reading assignments for the next lab sessions, the quiz on **NLP** closes **before our first session in January**. Similarly, the quiz on **IP** closes **before our second session in January**.

Optional: Notebook Feedback

If you want to get (ungraded) feedback on your notebook and the way you complete assignments, you can upload your Notebook as `firstname_lastname_dslab5.html` under **Optional Upload | Notebook Lab 5 (Ungraded)** to Moodle.
Your (voluntary) submission for this lab session should consist of **a jupyter notebook (exported as .html) starting from subsection Different Optimizers**. Make sure to include a **discussion section** at the end of the notebook where you comment your results and findings. This is an individual assignment. Collaboration in discussing the approach or some technical details is highly welcome, the uploads need to be prepared and uploaded individually. Make sure that essential original content (anything else than exchanging comments or texts such as headlines or discussion) distinguishes your upload from that of your colleagues significantly. Anything else is considered plagiarism.

Dos

- Make sure your notebook is organized, with cell outputs (e.g. plots) readable and commented.
- Make sure your plots are self-contained (e.g. add title, axes labels, ...).

Don'ts

- Not exporting your notebook as `.html`.
- Unnecessary/duplicate/inactive cells.
- Unnecessary imports or helper functions.
- Overlong print statement outputs or prints without formatting or context.

See this optional hand-in as a dry run for the two code-related assignments you need to submit for the lab sessions in NLP and IP.