Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology

D7041E “Applied artificial intelligence”

IT IS STRICTLY FORBIDDEN TO USE AI GENERATED CODE AND COPY EXISTING CODE FROM THE INTERNET. ALL CASES OF VIOLATION WILL BE REPORTED!

LAB 2: Random high-dimensional representation of structured data, Self-Organizing Maps and Back Propagation

1. Introduction

In this lab, we will work with a data representation technique called **distributed representation**, and two types of **Artificial Neural Networks (ANNs)**: **Self-Organizing Maps (SOMs)** and strained using the **Backpropagation** learning algorithm.

In a distributed representation, each feature is encoded as a vector in a high-dimensional random space $H=\{+1,-1\}^d$, where d is typically several hundred or even thousands. We will use this representation to encode feature vectors for data-driven classification problems. In this lab, you will see how **n-grams** can be implemented using high-dimensional (HD) representations, and how to build a simple **centroid-based HD classifier** for identifying the language of input texts.

A **Self-Organizing Map (SOM)** belongs to the domain of unsupervised learning in neural networks. It consists of a grid of neurons used to project multidimensional data onto a lower-dimensional space with meaningful coordinates, thereby revealing the underlying class structure in the data.

The **Multi-Layer Perceptron (MLP)** was one of the first successful neuro-inspired computational models. However, in its original form, it was difficult to parameterize and optimize, and its performance was often suboptimal in practice. The introduction of the **Backpropagation algorithm** was a breakthrough that enabled efficient training of multi-layer networks and paved the way for the modern development of artificial neural networks and deep learning.

Part 1. The language classification problem with distributed representations

In this lab you will classify the language of an input text. The language recognition will be done for 21 European languages. The list of languages is as follows: Bulgarian, Czech, Danish, German, Greek, English, Estonian, Finnish, French, Hungarian, Italian, Latvian, Lithuanian, Dutch, Polish, Portuguese, Romanian, Slovak, Slovene, Spanish, Swedish.

Task 1.1 Import datasets into Jupyter environment:

1. The training data is based on the News Wortschatz Corpora (Any Year):
   - https://wortschatz.uni-leipzig.de/en/download/
2. Test your classifier on the News Wortschatz Corpora (do proper splitting of the dataset for training/testing/validation)
3. Do final testing based on the Euro Parliament Parallel Corpus: http://www.statmt.org/europarl/
4. Import the data into your Jupyter environment, understand its structure
5. Do necessary pre-processing (removing punctuation, etc.)

Task 1.2 Constructing high-dimensional centroids

Use encoding procedure for n-grams in {+1;-1} coordinates as described in file: "ArticleEncodingNgramStatistics.pdf" linked in Canvas (see Labs module).

- Use n=3 (tri-grams). Use length of HD vectors d_1=100 and d_2=1000.

An HD vector for a particular input text of certain language is computed by adding all the n-gram vectors. Since we consider 21 European languages at the end of the training phase we will have 21 d-dimensional language HD vectors (centroids) stored in an array.

Question: what will be the size of the n-gram input vector in conventional (local) representation?

Question: Identify difficulties of working with conventional representations of n-grams in the machine learning context.

Task 1.3. Classification using hyperdimensional centroids

In the test phase for an unknown text sample first a query vector in the same fashion as you constructed language vectors in the training phase. To determine the language of this text sample compare its query vector to all the (21) language vectors w.r.t **cosine similarity measure**. Present confusion matrix, compute accuracy and F1-score.

Part 2. Unsupervised learning with Self Organizing Maps

Following the example of SOM implementation in the Jupyter notebook from the class, extend it to learn the SOM topology of the MNIST dataset. Study the effect of SOM hyperparameters.

Task 2.1 Unsupervised learning of hand-written digits with SOM:

1. Load the MNIST dataset
2. Use the flattened (that is 1D) array of pixels of each image as a feature vector.
3. Initializing the weights in the SOM network randomly, train SOM with grid 20x20, 40x40, 80x80.
4. Display the initial, intermediate (at 50% of iterations) and the final learned weights of SOM neurons as a grid of 28x28 images.
5. Now by passing the TRAINING examples through the trained SOM and recording the statistics of matching in each node assign labels to the neurons.
    - Display the confusion matrix of classification of the TRAINING SET
    - Display the confusion matrix of classification of the TEST SET

6. For a fixed number of iterations increase and decrease the learning rate of SOM:
    - What is the resulting effect?
7. For a fixed number of iterations and the best learning rate increase and decrease the exponential decay of the neighbourhood parameter.
8. What is the effect?
9. What is a biological neuron? How does it relate to the concept of neurons in SOM?

Part 3. fundamentals of the Artificial Neural Networks and Backpropagation algorithm I

In this part we will continue to classify hand-written digits from the publicly available dataset MNIST. In this lab you will use a Jupyter notebook Lab3_ANN_backprop, which is available in Canvas.

Task 3.1. Multi-layer perceptron and backpropagation

The goal of this task is to understand the fundamentals of the backpropagation algorithm and study classification performance of the multi-layer perceptron.

1. Understand the implementation structure of the multilayer perceptron in Jupyter notebook Lab5_ANN_backprop.
    a. Be able to explain the principle of backpropagation algorithm;
    b. Be able to explain the meaning and the role of the Softmax function;
    c. Be able to name typically used non-linear output functions and implications of choosing one or another for implementation.
    d. The code in the provided Jupyter notebook will stop execution at several points. Find the places in the code, where the execution breaks, answer the questions, comment out the exit line and run the code again.
2. Run the code with the suggested configuration of the hyperparameters: number of epochs = 70 and learning rate =0.05. What is the classification accuracy?
3. Run the code with Learning rate =0.005 and Learning rate =0.5. Explain the observed differences in the functionality of the multi-layer perceptron.
4. Extend the code implementing the ReLU output function. Run the perceptron with the suggested by default configuration of hyperparameters: number of epochs = 70 and learning rate =0.05. What is the classification accuracy? Find the values of the learning rate which results in comparable to Sigmoid case accuracy.

Congrats, you are now done with Lab 2!

You have just become familiar with fundamentals of the cutting edge data representation technique called distributed random representations , a very powerful tool for unsupervised learning - Self-Organizing Maps as well as fundamentals of the Artificial Neural Networks and Backpropagation algorithm!

Well done!