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 1: Fundamentals of Machine Learning and Word Embeddings

1. Introduction

In this lab, we will work with data-driven classification problems as well as data-driven models for learning word representations.

In classification problems, the task is to assign one label from a finite set of categories to each input sample. The challenge is that no predefined model describing each category is available in advance—meaning we cannot hard-code it into an algorithm but must instead learn the model from data. In this lab, you will explore two instances of this problem: image classification and flower type classification. Specifically, we will classify hand-written digits from the publicly available MNIST dataset and flower types using feature descriptions from the IRIS dataset.

We will also address the task of forming vector representations for words in a vocabulary given a text corpus. The challenge here is to generate word vectors such that the relationships between vectors reflect the linguistic relationships between words. In this lab, you will work with two methods – one from each class of models. **Word2Vec** is a popular connectionist approach for building word embeddings, while **Random Indexing** is a distributional model based on the principles of hyperdimensional computing.

For this lab you must submit a report showing the answers to all the questions and screenshots showing the intermediate steps of the algorithm.

Part 1. Warming up for the lab

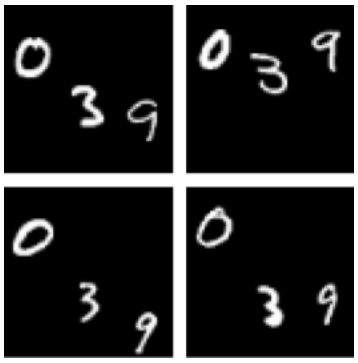
To master basic data pre-processing skills collect your own dataset of images (take at least 5 photos of 5 categories of objects)

Task 1.1 Import your own dataset into Jupyter environment:

1. Make an (n,x,y,3) array containing all the samples of your dataset. Here n is the number of samples, x and y is the resolution of your photos and 3 is the rgb components.
2. Write a function for plotting a sample PlotSample, which takes the index of a sample as an argument and plots it using the functionality of the **matplotlib** library.
3. Follow the tutorial "Preprocessing for deep learning" (https://hadrienj.github.io/posts/Preprocessing-for-deep-learning/), which is alsoi linked from Canvas and perform all the pre-processing steps for your dataset.
    - You need to demonstrate the Jupyter notebook with all the steps.

Task 1.2 Import somebody's else dataset into Jupyter environment:

1. Two months ago I got an e-mail from my collaborator working with multidigit MNIST dataset (see an example below):



"you asked me to share with you data from my experiments. Here it goes 11 thousand samples. The data are 1024-dimensional embeddings of handwritten digits placed on 9 different positions on a scene. The data is formed as a dictionary within a dictionary. The key to the first dictionary is the number of the position and the key to the second dictionary is the digit itself." The file with data ("vecs.npy") is part of the code distribution for Lab1. It took approx. 1 hour

for me to be able to load the data as Numpy arrays suitable for input to an AI algorithm. How long time does it take for you?
2. You need to demonstrate a code that automatically forms **two** Numpy arrays for an arbitrary chosen position on the scene – one containing the embeddings and the second one containing the labels in the order presented in the data file.
3. You need to demonstrate a python code for randomly permuting the order of data and the labels.

Part 2. Nearest Neighbour classifier: skills of working with datasets and hyperparameters

The first approach you will test has little to do with learning, however we will use it for mastering the skills of working with datasets as well as a baseline for benchmarking performances of more advanced approaches.

MNIST data set:

One commonly used toy image classification dataset is the MNIST dataset. This dataset consists of 60,000 tiny images that are 28 pixels high and wide. Each image is labeled with one of 10 classes (“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”). These 60,000 images are partitioned into a training set of 50,000 images and a test set of 10,000 images.

Task 2.1

You are provided with an implementation of a special case of the K-Nearest Neighbors classifier (i.e. 1-NN) using L1 norm. The 1-NN classifier takes a test image, compares it to every single one of the training images, and predicts the label of the closest training image.

1. Run the provided code for 1-NN classifier. What is the accuracy of the method?
2. Modify the code of the NearestNeighbor example so that it uses L2 norm for classification. How the classification accuracy has changed now?
3. The accuracy of the NN classifier that you have observed in the previous two tasks is VERY low. This is due to the bug in the code – find it and fix it. HINT:

check the distances computed by Python and those computed by hand for several pairs of images.

4. Extend the code of the NearestNeighbor example with necessary methods, which implement k-NN klassifier. That is instead of finding the single closest image in the training set, you will find the top **k** closest images, and have them vote on the label of the test image.

Task 2.2. Hyperparameters, cross-validation

The k-NN classifier requires a setting for k as well as there is an option of choosing one or another distance function (e.g. L1 or L2 norm). These choices are examples of so called **hyperparameters**, which needs to be optimized for best classification accuracy. **In machine learning one should avoid adjusting these parameters based on the test data!** This is rather obvious since in this case your algorithm will be optimized for the particular data set. A typical approach for choosing test parameters is to have yet other data set for validation purposes. In some cases, when the dataset is small, a technique called cross-validation is used. It is illustrated in the figure below.

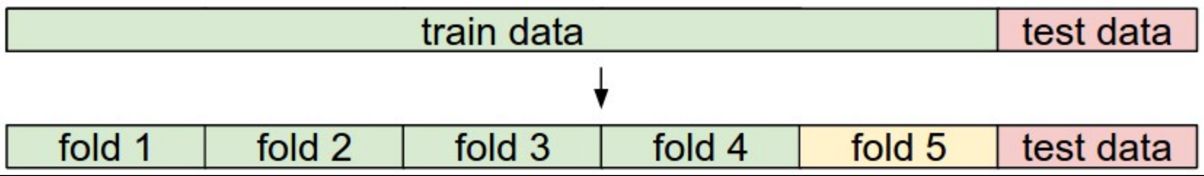


Figure. Data split for cross-validation

The training set is split into folds. In the figure above 5 folds are displayed. One fold for example fifth (shown by yellow color) is chosen for validation while other four for training. In cross-validation one iterates the validation fold when choosing the optimal values for hyper parameters. Typical number of folds you can see in practice would be 3-fold, 5-fold or 10-fold cross-validation.

In the very end once the model is trained and the best values of hyperparameters were determined, the model is evaluated a single time on the test data (red).

1. Extend the code of the NearestNeighbor example with necessary methods to find the best value of k using 3-fold cross-validation method. Use L2 norm as the distance function.

2. What is the classification accuracy on the test data for the best value of **k**?

Part 3. Support Vector Machines

Support Vector Machines is one of the most widely used algorithms for solving classification problems. The algorithm introduces fundamental AI and data science concepts such as model-based learning, highdimensional spaces, separability in highdimensional spapces and others. In this part of the lab you will use implementation of SVM from scikit-learn Python library to understand the fundamentals of using SVM.

Task 3.1 Load Iris dataset, split the dataset into the training and the test parts in proportion 80:20. Train SVM with linear, polynomial and RBF kernels for multiclass classification. In this task your solution could be inspired by the following blog: https://towardsdatascience.com/support-vector-machines-svm-clearly-explained-a-python-tutorial-for-classification-problems-29c539f3ad8

1. Construct confusion matrices for one-vs-one and one-vs-rest training approaches and each kernel type;
2. Using which kernel the best accuracy and F1 score is achieved?
3. Extract the support vectors for each class in one-vs-rest training case.
4. Plot the decision boundary for features 2 vs. 3 and 3 vs. 4.

Part 4. Word embeddings: connectionist and distributional approaches

In this part of the lab you will work with two datasets one for training and one for testing. As the training data you will use TASA corpus of school reading materials from kindergarten through high school. It contains approximately 37,600 text samples. There is also the evaluation task for the learned embeddings: the Test of English as a Foreign Language (TOEFL) synonymy assessments. The dataset (file "new_toefl.txt") contains 80 synonym tasks. Each task includes a query word and 4 alternatives. One of the alternatives is a synonym of the query word. If the chosen word is the correct answer (i.e. the synonym) than the score on the overall dataset increases by 1. The overall score on the TOEFL synonymy assessment is an integer number between 0 and 80. Note that if choosing answers randomly (e.g. by tossing a coin twice) the score averaged after many runs will be close to 20 or 25% (20/80) (i.e. chance of randomly guessing the correct answer out of four alternatives).

You are provided with two Python projects GensimW2V (for connectionist approach to word embedding) and RI (Random indexing method), which are available in Canvas.

Note. In order to execute the provided code you might have to install several packages: the library implementing Word2Vec (e.g, by easy_install -U gensim) and natural language toolkit (ntlk). You may also have to download corpora for ntlk package in order to perform the lemmatization.

Note. Usually before building word embedding models the original text is pre-processed. Common pre-processing steps include disregarding the most frequent words (e.g., articles "a" and "the") and lemmatization (read more about it https://en.wikipedia.org/wiki/Lemmatisation), which changes words to their default forms (e.g., were -> be; goes -> go; tables -> table; etc.). You are already provided with the lemmatized text in the file "lemmatized.text". In order to see the difference take a look on the original TASA corpus (file tasa.text) in the root folder for this Lab.

Note. The lemmatization is still used in the code in order to process the words in the TOEFL dataset.

Note. The TASA corpus used in this lab was made available for use only in academic research, courtesy of Touchstone Applied Sciences Associates.

Task 4.1. Word2Vec

You have got the code to train word embedding using neural networks. The code also estimates the performance on TOEFL task.

- Get the performance of the model for three different dimensionalities. The choice of dimensionalities is on your own but use different values (e.g., 10, 100, 1000).
- If your computational resources allow run simulations several times (e.g. 5) for each dimensionality
- Report the accuracy on TOEFL for all simulations
- Elaborate how accuracy changes with the dimensionality

Task 4.2. Random indexing with permutations

You have got the code to train word embedding using Random indexing. The code also estimates the performance on TOEFL task.

- The current code works adequately only when size of window is 2. What should be changed in order to overcome this issue?
- Get the performance of the model for three different dimensionalities. The choice of dimensionalities is on your own but use different values (e.g., 1000, 4000, 10000).
- If your computational resources allow run simulations several times (e.g. 5) for each dimensionality
- Report the accuracy on TOEFL for all simulations

- Elaborate how accuracy changes with the dimensionality

As a conclusion elaborate of the accuracy of both methods for synonyms tasks.

Also compare how computational demanding are both methods. Remember that RI forms representations only for the subset of the vocabulary, which is needed for TOEFL tasks. You could train all words and check how much time is needed or estimate the training time based on the time needed to train the subset

Final Note. Be patient as training data-driven models take time :-). Word2vec reports its progress while executing while RI does not. You are more than welcome to implement this functionality.

Congrats, you now have completed Lab 1!

you have just become familiar with fundamentals of the cutting-edge learning techniques for data-driven classification as well as with the frontier methods for data-driven machine semantics!

Well done!