PROJECT ML

University of Hamburg

Department of Mathematics

Jonas Eckhoff - Timo Greve

Max Lewerenz - Giulia Satiko Maesaka - John-Robert Wrage

Machine Learning Methods
Group 5

Contents

List of Figures

1 Introduction

What is this paper for? What are the main goals?

E.g. this is for ourself. We want to build an overview about the common algorithms, how they work and their advantages/disadvatages.

2 What is ML?

First some words about ML. Maybe from Lotz, first chapter? Next we could use this graphic to visualize the topic.

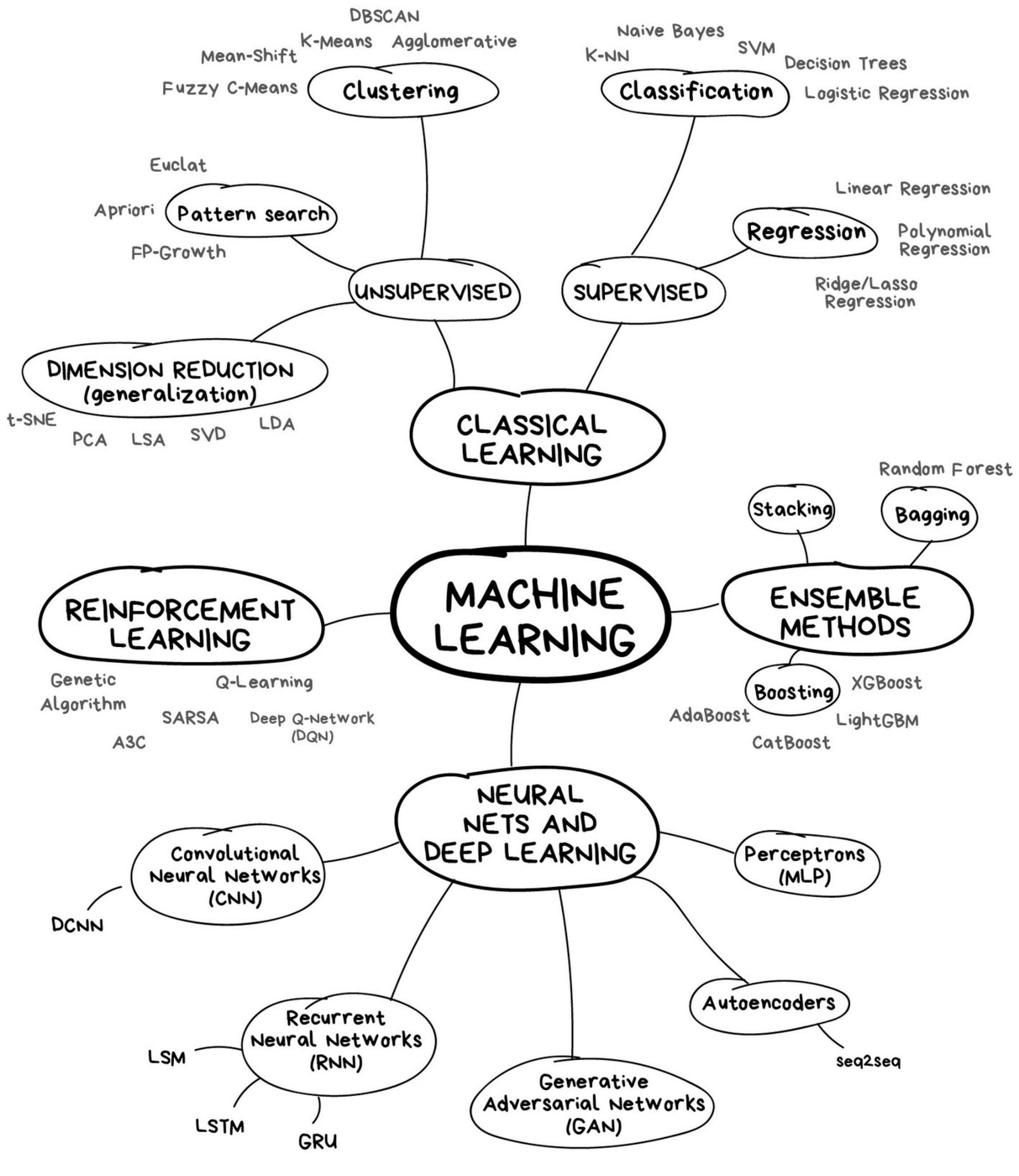


Figure 1: ML-Overview

Finally i would suggest to finish with a Table.

I would structure it like: Algorithm, Main Idea(in like 3-5 sentences), possible application(image processing, clustering, etc.) advantages, disadvantages

2.1 Basis ML 1(maybe Data)

We could speak a lil bit more about data in this topic?

2.2 Basis ML 2

TBA

3 Classical learning

Difference between Supervised unsupervised.

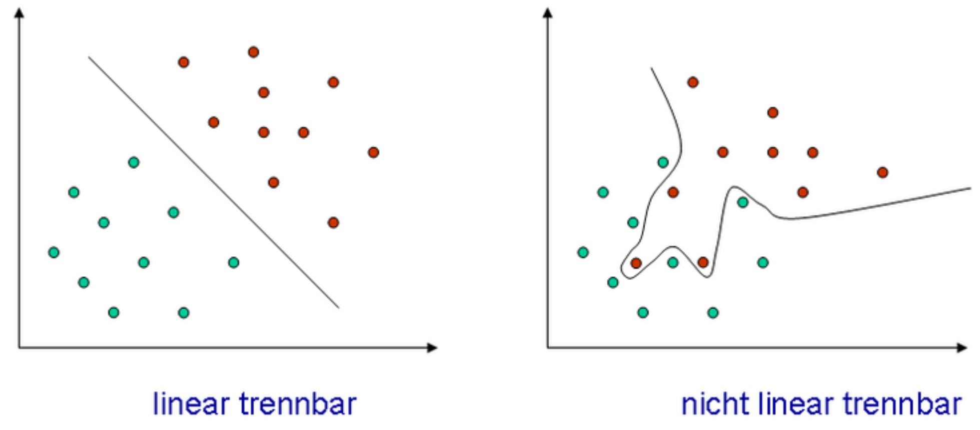
3.1 Supervised

Um einen Abfragepunkt zu klassifizieren, nutzt diese Methode die gespeicherten

3.2 Support Vector Machine

3.2.1 Introduction

Support Vector Machines (SVM) are supervised learning algorithms for classification. They are one of the most widely used supervised learning algorithms. SVMs offer a high accuracy when it comes to classification. The idea behind a SVM is to find a hyperplane that seperates classes of datapoints with a large margin. Where the margin is the smallest distance between the closest datapoint x of one class and the hyperplane (therfore the name 'support vector'). Because of this feature the SVM is sometimes also called **large margin classifier**. The so called kernel trick can be applied when non linear data has to be classified.



3.2.2 Mathematics behind SVMs

For given input data $X = \{x_1, ..., x_m\}$, with $x_i \in \mathbb{R}^n$ and matching result vector $Y = \{y_1, ..., y_m\}$, with $y_i \in \{-1, 1\}$. We want a classifier

$h : \mathbb{R}^n \to \{-1, 1\}$ such that $h(x) = \begin{cases} 1 & \text{,if } w^T x + b > 0, \\ -1 & \text{,if } w^T x + b < 0. \end{cases}$ with $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. We want h to be correct for most samples. This formulation leads to the following primal problem:

$min_{w,b,s} \frac{1}{2} w w^T + C \sum_{i=1}^{m} s_i,$

s.t.: $y_i(w^T x_i + b) \geq 1 - s_i$, $s_i \geq 0$.

Where s_i are the so called slack variables that should denote the distance from the correct margin if a point is misclassified.

Intuitively we want to maximize the margin what results in minimizing $||w||^2$ including a penalty when something is misclassified. C controlls the strength of the misclassification penaly. One could view it as a inverse regularization parameter. A large C results in overfitting and a smaller C results in a "smoother" fit. The dual problem to the above Primal problem is:

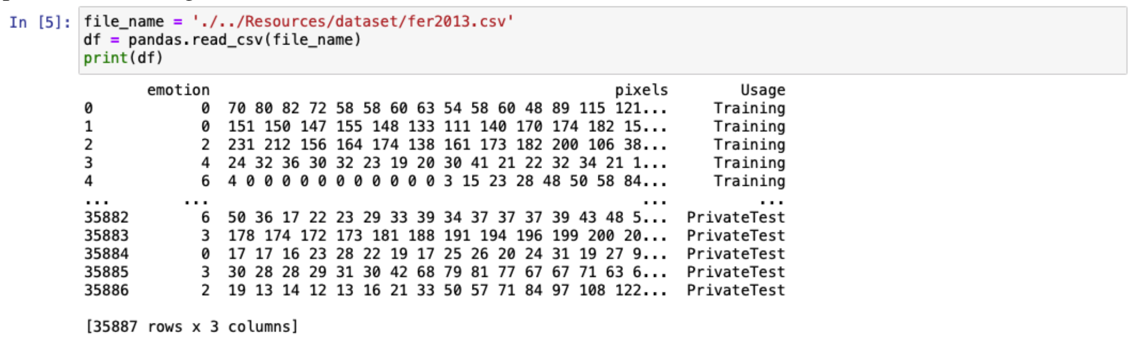
$min_\lambda \frac{1}{2}\lambda^T Q\lambda - e^T\lambda$

s.t.: $y^T\lambda = 0$ and $0 \leq \lambda_i \leq C$ for all $i = 1, ..., m$. The entries of the matrix $Q \in \mathbb{R}^{m\times m}$ are given by $Q_{i,j} = y_i y_j \langle x_i, x_j\rangle$.

This shows that the minimization only depends on the scalar product of x_i and x_j and we can apply the kernel trick. Instead of the standard scalar product we can use a kernel function $K(x_i, x_j)$.

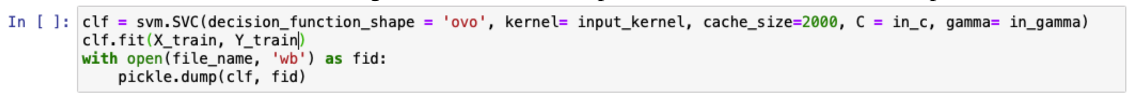
3.2.3 In our case

We use the python library sklearn, which can easily be installed via pip. First we get the training data and split it into training and test data.



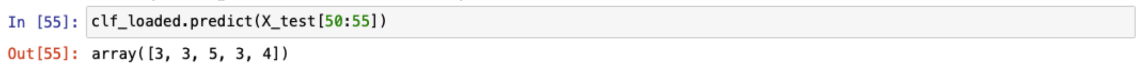
```
In [5]: file_name = './../Resources/dataset/fer2013.csv'
        df = pandas.read_csv(file_name)
        print(df)

               emotion                                             pixels        Usage
        0            0  70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...     Training
        1            0  151 150 147 155 148 133 111 140 170 174 182 15...     Training
        2            2  231 212 156 164 174 138 161 173 182 200 106 38...     Training
        3            4  24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...     Training
        4            6  4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...     Training
        ...        ...                                                ...          ...
        35882        6  50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...  PrivateTest
        35883        3  178 174 172 173 181 188 191 194 196 199 200 20...  PrivateTest
        35884        0  17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...  PrivateTest
        35885        3  30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...  PrivateTest
        35886        2  19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...  PrivateTest

        [35887 rows x 3 columns]
```

Next we fit the model to the training data for the chosen parameters and safe it with the pickle model.



```
In [ ]: clf = svm.SVC(decision_function_shape = 'ovo', kernel= input_kernel, cache_size=2000, C = in_c, gamma= in_gamma)
        clf.fit(X_train, Y_train)
        with open(file_name, 'wb') as fid:
            pickle.dump(clf, fid)
```

Now the model can give a prediction on new images.



```
In [55]: clf_loaded.predict(X_test[50:55])

Out[55]: array([3, 3, 5, 3, 4])
```

3.2.4 Regression

Like for all other subsubsections i would sugest that we structure the Algorithms like in the table at Ch.2.(**Algorithm**, **possible application**, etc.), but also add some links etc. Trainingsdaten und berechnet den Abstand zu den jeweiligen Merkmalen. In

3.2.5 Classification

See regression

3.3 Unsupervised

See regression

3.3.1 Clustering

See regression

3.3.2 Pattern search

See regression

3.3.3 Dimension Reduction

See regression

4 Neural Networks and Deep Learning

4.1 Convolutional Neural Networks CNN

A Convolutional Neural Network is a Deep Learning algorithm which is primarily used to classify images. It can more easily capture spatial (and temporal) dependencies since the convolution filters use a linear combination of neighboring pixels to calculate an output value. After a convolution filter is applied, a Max-Pooling filter can be applied, which lowers the amount of nodes in the network. A 2x2 Max-Pooling filter, for example, replaces every 2x2 square with the maximal value in that square, resulting in lowering the amount of nodes after that layer to a quarter. At the end the multidimensional layers are flattened to a one dimensional vector that is connected to the output via a fully connected layer.

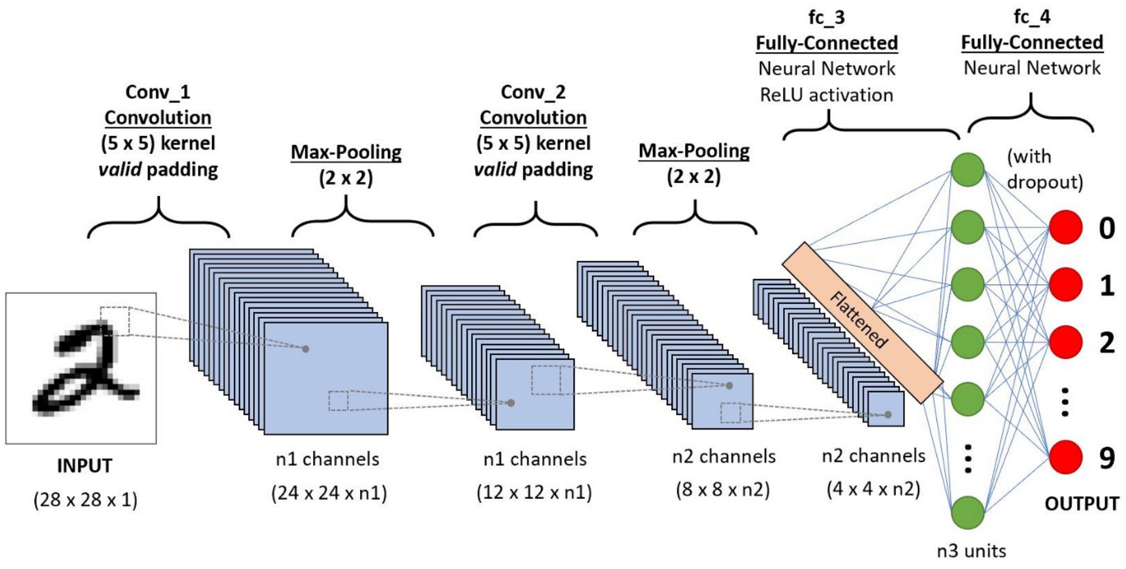


Figure 2: Example of a CNN architecture

Helpful links:
A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way
`https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-net`

Since we want to detect emotions in images, a convolutional neural network would be a good choice. It is the go-to type of network for these kind of computer vision problems.

4.2 Recurrent Neural Networks RNN

A recurrent neural network (RNN) is a type of artificial neural network commonly used in speech recognition and natural language processing. It uses feedback loops, that allow information to persist (the network has memory), to process sequences of data.

RNNs are not useful to analyse facial expression in an image.

4.3 Generative adversarial networks GAN

Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other in order to generate new, synthetic instances of data that can pass for real data. They are used widely in image generation, video generation and voice generation.
One neural network, called the generator, generates new data instances, while the other, the discriminator, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

Helpful links:
A Beginner's Guide to Generative Adversarial Networks (GANs)
`https://pathmind.com/wiki/generative-adversarial-network-gan`

GANs can be used to generate images and videos. We could use this type of network to expand training data. Having a well trained GAN would allow us to create new images for specific emotions that look convincingly real in order to prevent overfitting by creating an endless stream of new images. It may also be used to create video training data that can be used to train a neural network with memory to better capture changes in emotion in a video stream. The training of a GAN however takes a very long time compared to a simple CNN which makes it probably unfeasible for us even though it would create interesting possibilities to explore.

4.4 Autoencoders

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact "summary" or "compression" of the input. An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.

Helpful links:
Applied Deep Learning - Part 3: Autoencoders
`https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af`

An autoencoder is used to reduce the dimensionality. Since there is a lot of redundancy in images, especially if they all contain more or less centered faces, it can be expected that an autoencoder could shrink the input size considerably before using a different neural network. This could potentially make the training process a lot quicker. We would however not be able to use a convolutional neural network anymore because the 2 dimensional image structure is lost by removing the redundancy with the autoencoder. Moreover some information may be lost in the process which could affect the overall accuracy of the neural network.

The effectiveness of an autoencoder could possibly be explored if there are given time constraints for the training process. It can also be compared to principal component analysis (PCA) which removes redundancy in a more controlled way.

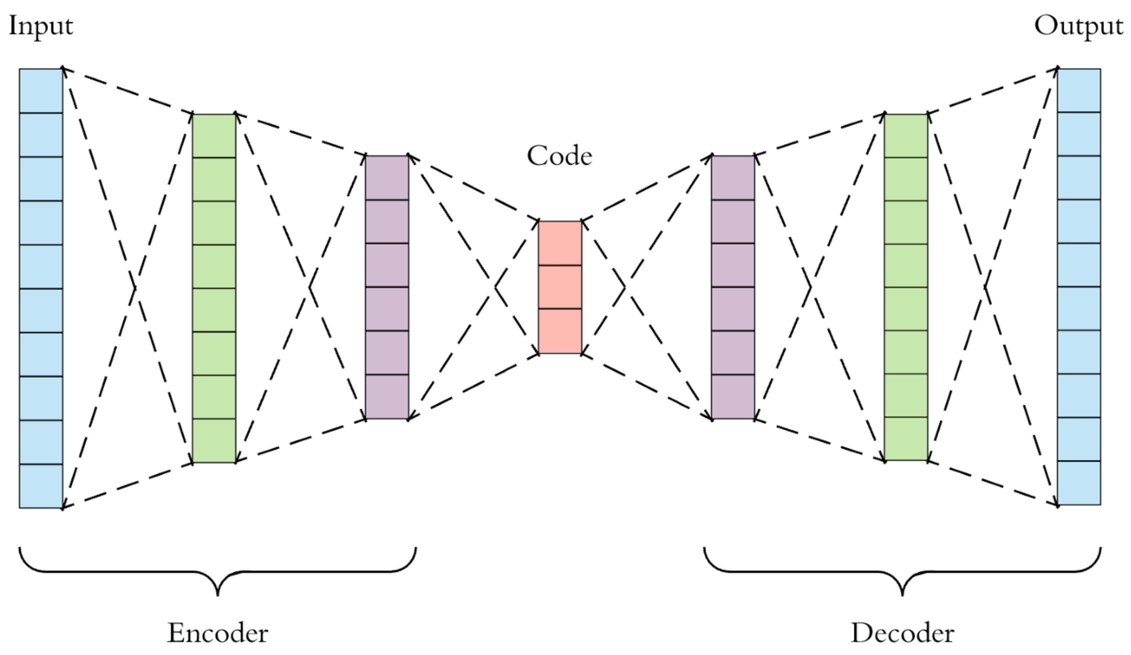


Figure 3: Example of an autoencoder architecture

4.5 Perceptrons MLP

5 Ensemble Methods

5.1 Bagging

5.2 Boosting

5.3 Stacking

6 Reinforcement learning

Bibliography

[1] DEUTSCHE AKTUARVEREINIGUNG E.V., 2012: Unisex-Tarifierung, abgerufen am 10. November 2019: https://aktuar.de/fachartikelaktuaraktuell/Lebensversicherung_Unisex_Aktuar-aktuell_19.pdf#search=unisex

[2] ERTEL, W. (2016): *Grundkurs Künstliche Intelligenz*. Springer Vieweg, Wiesbaden. 4. Auflage.