Final Course Assignment

Mark Niehues, Stefaan Hessmann
Mathematical Aspects in Machine Learning

14. Juli 2017

1 Introduction

In the past course we dealt with the broad mathematical foundations of machine learning. To get an idea of what the consequences of those mathematical theorems and approaches are and to get in touch with the standard Python tools, we have evaluated an comparatively easy data science example found on kaggle.com. Since this was our first machine learning project, we decided to deal with an rather simple problem. The example dataset [**Kaggle2017**] consists of the historic passenger records of the disastrous Titanic maiden voyage in 1912. The goal in this challenge was to predict if a passenger survived the accident based on informations like for example age, sex and payed ticket fare. Therefore it is in terms of machine learning a *classification problem* with two classes: survived and not survived.

Inspired by sample solutions from the website, we first took a deeper look on the dataset and tried to select the most significant influences by reviewing the statistical properties of the dataset. In the following we implemented an naive Sequential Minimal Optimization (SMO) algorithm and ran a few tests with them in order to finally compare the results with other machine learning algorithms.

2 Applying Machine Learning Methods on the Titanic Disaster

2.1 Dataset

The given dataset consists of a CSV-file containing data of 891 passengers. The dataset contains an ID for every passenger, a label if the passenger has survived the disaster and the features that are described in table 1. It can be noticed that some of the features are incomplete.

After loading the dataset, it is necessary to process the data for our learning machine. Therefore the different features will be investigated to select meaningful features and the missing data needs to be handled.

2.2 Feature: Sex

The sex-feature divides the passengers into the categories ’female’ and ’male’. Figure 1 shows the probability of survival for male and female passengers. It is obvious that females had a much higher probability to survive than the male passengers. ’Sex’ seems to be a useful feature for the learning machine.

Tabelle 1: Features and their amount of missing data.

PassengerId	Unique ID for every passenger	0.0 %
Survived	Survived (1) or died (0)	0.0 %
Pclass	Passenger's class	0.0 %
Name	Passenger's name	0.0 %
Sex	Passenger's sex	0.0 %
Age	Passenger's age	**19.87 %**
SibSp	Number of siblings/spouses aboard	0.0 %
Parch	Number of parents/children aboard	0.0 %
Ticket	Ticket number	0.0 %
Fare	Ticket-price	0.0 %
Cabin	Number of the passenger's cabin	**77.10 %**
Embarked	Port of embarkation	**0.22 %**

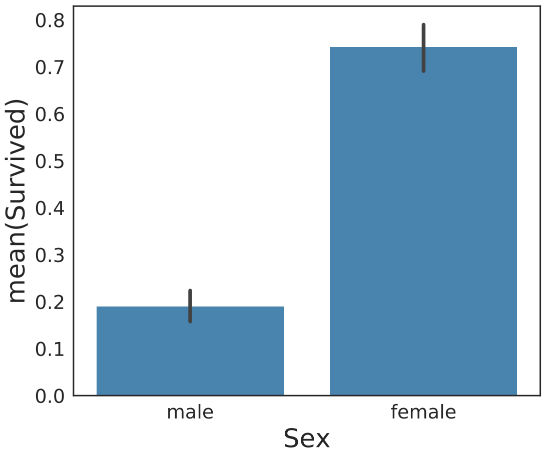

Abbildung 1: Survival probability depending on the passenger's sex.

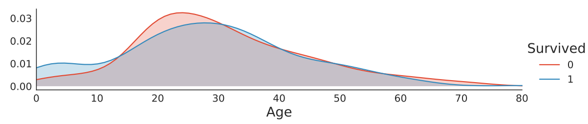

Abbildung 2: Survival probability depending on the passenger's age and sex.

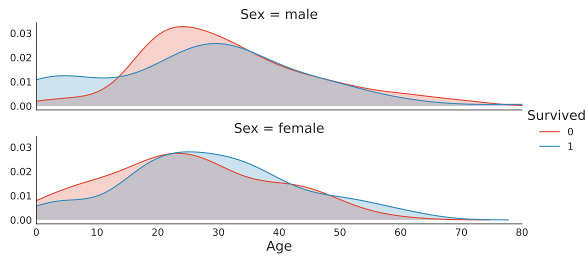

Abbildung 3: Survival probability depending on the passenger's age and sex.

2.3 Feature: Age

The survival distributions in function of the passengers' ages are pointed out in figure 2. The plot shows that children under twelve years were most likely to survive, whereas older children and young adults until about 30 years had bad chances. The chance for adults above their thirties to be rescued is nearly independent of their exact age at about 50 percent.
If we categorize the survival distribution depending on the age additionally by the sex feature as shown in figure 2, it turns out that the age has different effect on the survival probabilities of females and males. Young male passengers were likely to survive while males between about 12 and 30 years were unlikely to survive. This effect is inverted for females, where young girls had fewer chances to be rescued than females between about 25 and 40 years. From this it follows that age is a feature that can have influence on the predictions of our learning machine. Especially if it is combined with the sex feature. The missing values of the age dataset will be handled after the inspection of the other features.

2.4 Feature: Name and Title

The given dataset also contains a list of the passenger-names that have been involved in the titanic accident. At first sight a name does not appear to be a useful feature for our survival predictions, but the name-list contains also contains a persons title. Figure **??** shows the total number of occurrences for all titles that have been found. The titles 'Mr.', 'Master', 'Mrs.' and 'Miss' can are relatively common, whereas the other titles occur only infrequently. Therefore all other titles are grouped into a group named 'Res'. The figure also shows on the chances of survival for people with different titles on the right-handed plot. The plot points out that a persons title has an impact on the survival odds.

3 Implementation of an easy SMO Algorithm

To get a better understanding of what a Support Vector Machine does, we decided to implement one on our own using several publications. Most of them were based on the important paper of Platt [**platt**] where he introduced a new approach for the calculation of the Support Vectors that improves the performance a lot. This algorithm is called Sequential Minimal Optimization. Performance was not the highest priority for us but instead understandability and the costs of implementation. Therefore we implemented a less complex version of the algorithm presented in Platt's paper.

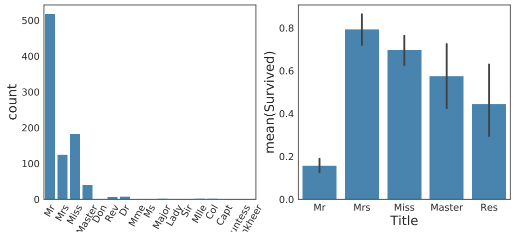

Abbildung 4: Total counts of the different titles (left) and survival probability in dependency of the title (right).

Appendix

A Pseudo Code of the SMO algorithm

B Code

```

import numpy as np


class Kernels:
    """
    Class that holds different Kernels
    """
    def __init__(self, gamma):
        self.gamma = gamma
        self.kernels = {
            "rbf" : self.kernel_rbf,
            "linear": self.kernel_lin}

    def get_kernel(self, kernel_name):
        return self.kernels[kernel_name]

    def kernel_lin(self, x, y):
        """
        Linear kernel
        """
        return x.dot(y)

    def kernel_rbf(self, x, y):
        """
        RBF Kernel
        """
        d = x - y
        return np.exp(-np.dot(d, d) * self.gamma)
```

Input:

C: regularization parameter
tol: numerical tolerance
max_passes: max # of times to iterate over α's without changing
$(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$: training data

Output:

$\alpha \in \mathbb{R}^m$: Lagrange multipliers for solution
$b \in \mathbb{R}$: threshold for solution

- Initialize $\alpha_i = 0, \forall i, \quad b = 0$.
- Initialize $passes = 0$.
- **while** $(passes < max\_passes)$
  - $num\_changed\_alphas = 0$.
  - **for** $i = 1, \ldots m$,
    - Calculate $E_i = f(x^{(i)}) - y^{(i)}$ using (2).
    - **if** $((y^{(i)}E_i < -tol \;\; \&\& \;\; \alpha_i < C) \;||\; (y^{(i)}E_i > tol \;\; \&\& \;\; \alpha_i > 0))$
      - Select $j \neq i$ randomly.
      - Calculate $E_j = f(x^{(j)}) - y^{(j)}$ using (2).
      - Save old α's: $\alpha_i^{(\text{old})} = \alpha_i$, $\alpha_j^{(\text{old})} = \alpha_j$.
      - Compute L and H by (10) or (11).
      - **if** $(L == H)$
        continue to next i.
      - Compute η by (14).
      - **if** $(\eta >= 0)$
        continue to next i.
      - Compute and clip new value for α_j using (12) and (15).
      - **if** $(|\alpha_j - \alpha_j^{(\text{old})}| < 10^{-5})$
        continue to next i.
      - Determine value for α_i using (16).
      - Compute b_1 and b_2 using (17) and (18) respectively.
      - Compute b by (19).
      - $num\_changed\_alphas := num\_changed\_alphas + 1$.
    - **end if**
  - **end for**
  - **if** $(num\_changed\_alphas == 0)$
    $passes := passes + 1$
  - **else**
    $passes := 0$
- **end while**