Chapter 3

Problem formulation

We chose machine learning, in particular the subfield deep learning, as our main approach to solve the problem of time-independent price prediction in terms of the game FIFA 20 and it's game mode FIFA Ultimate Team. Today machine learning is a widely researched scientific field, that constantly gains in relevance and interest. This leads to the fact, that it is split into many different research areas with varying approaches and targets. We will use a fully connected deep neural network to solve the task of price prediction of football players in EA Sports game FIFA Ultimate Team.
To get feedback about the quality of our obtained results of the deep learning approach we also applied other models to the same problem. In the subsequent sections we will discover the theoretical base of those different approaches, where we focus the most on the last section about our main concept, the deep learning model.

The subsequent assumptions hold for all of the whole chapter:

- x is a vector of n input feature values of one data sample like in our specific case the attributes of one football player.

- X denotes the set of m training sample features that are used to train our models. This means $X = \begin{bmatrix} x_1 & \dots & x_m \end{bmatrix}$, where m is the number of training data samples.

- y describes the target value of one sample, in our instance equal to the price of a specific football player.

- Y designates the set of m training sample target values, that are used to train our models, this means $Y = \begin{bmatrix} y_1 & \dots & y_m \end{bmatrix}$

3.1 Problem description

We first want to define and understand what task we are facing in our case of the price prediction of FIFA Ultimate Team football players. Being more precise we want to make the best possible time-independent prediction of the price of a random football player from the online trade market in the game FIFA Ultimate Team by considering various information about this specific player, the so called features. This description of our task matches very well with the definition of a regression problem.
"[...] using data to predict, as closely as possible, the correct real-valued labels of the points or items considered." [8]
The task of regression can be described by the target of finding a function $r : x \to y$ where X and y are the at the beginning defined notations.
As a difference to the very popular classification task we are not aiming to predict the correct label of the sample but try to minimize the resulting error between prediction and label. This error is called loss and is denoted by $L : y \times y' \to \mathbb{R}_+$
One of the most popular loss functions is the squared loss function, defined by: $L_2(y, y') = |y - y'|^2$.
As L only returns the error for one specific prediction, we want to transfer our obtained results and merge it into a more general formula with more significance, taking into account all training samples.
Therefore we use the following equation, called the mean squared error:

$$R(r) = \frac{1}{m} \sum_{i=1}^{m} L(r(x_i), y_i) \tag{3.1}$$

m denotes the number of single training samples and r and L are the former defined functions.
We have now a regression function r that returns a label value y for an arbitrary input-vector X, a loss function L that measures the magnitude of error between the predicted label and the actual label value and the function $R(r)$ that hands back the mean squared error of our regression function and the training data. [8, p. 237-238]

3.2 Linear Regression

At first we have a look at a well known basic approach when it comes to regression tasks, the linear regression.
The main idea is to find a function $f(x)$ that approximates the related y value to a certain x vector as close as possible for all $x \in X$. We define this function by $f(x) = wx + b$, where w is meant to be an adjustable parameter to make the model fit the label values y the best.
$f(x)$ returns y' as a prediction value for the related y value of x. As discussed

in the former section we can use L_2, the squared loss, to evaluate the error of the prediction of $f(x)$. The idea is to minimize the squared error for all $x \in X$, taking into account all training samples simultaneously, like we did in the former section where we defined the mean squared loss. The squared loss can be applied to our optimization by transforming it into the following formula:

$$\min_{\mathrm{W}} F(W) = \frac{1}{m} \| X^T W - Y \|^2 \tag{3.2}$$

where X are our features denoted as $\begin{bmatrix} \theta(x_1) & ... & \theta(x_m) \\ 1 & ... & 1 \end{bmatrix}$ with an additional 1 because of dimensional reasons when multiplying it with the weights vector. W are the weights written as vector in the following form $\begin{bmatrix} w_1 \\ ... \\ w_N \\ b \end{bmatrix}$. Because F is a convex, differentiable function it has a global minimum if $\nabla F(W) = 0$. Thus we can reformulate equation 3.2 as following:

$$\frac{2}{m} X (X^T W - Y) = 0 \Leftrightarrow XX^T W = XY \tag{3.3}$$

The obvious unique solution of the right side of equation 3.3 if XX^T is invertible is: $(XX^T)^{-1} XY$. Otherwise there is a family of solutions, defined by the pseudo-inverse of matrix XX^T.
Linear regression can be easily visualized for the simple case of n=1 which corresponds to the case of one input feature for each x.
In figure 3.1 we can see this visualization, where the blue points our data points (x and y) and the red line is the linear approximation that minimizes the mean squared error.

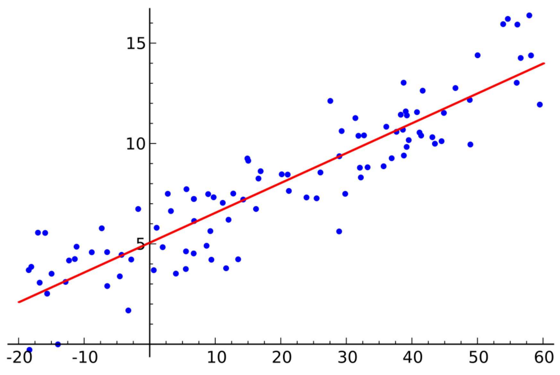

Figure 3.1: linear regression problem []

To close this section about linear regression we wanna have a short look at the expense of this approach.