An Introduction to Statistical and Probabilistic Linear Models

Proseminar Data Mining

Maximilian Mozes
Fakultät für Informatik
Technische Universität München
Email: mozes@cs.tum.edu

***Abstract*— With the increasing importance of data analytics and its corresponding relevance for many different fields of applications, consistent approaches for analysing, mining and learning from contextual observations expressed in form of data have taken centre stage in the computational area of data science and analytics. In this paper, we examine basic theoretical concepts associated with statistical and probabilistic learning theory that are utilized for solving regression and classification tasks in a mathematical context. For this purpose, we provide an overview of the principal functionalities and the mathematical derivation of such learning concepts and discuss the underlying linear approaches that are employed by state-of-the-art supervised learning algorithms. In doing so, we focus on both the abstract description as well as the illustration of such concepts by providing examples of common linear algorithms that are used for supervised classification and regression tasks.**

***Index Terms*— statistical learning theory, linear regression, linear classification, maximum likelihood estimation, bayesian statistics, ridge regression, probabilistic generative models, probabilistic discriminative models, logistic regression**

I. Introduction

The concept of linear models for regression and classification resides in the centre of statistical learning theory. The principal idea of linear models and their application in the area of data science and artificial intelligence is derived from their initial establishment in the field of statistics, where linear regression models play an essential role in predictive analysis scenarios. Describing the concepts and the functionality of linear models for learning applications therefore requires the creation of a mathematical notion for how these concepts acquire the ability to learn from data and to make predictions for future observations. In the course of this paper, we focus on the illustration of linear models for the regression and the classification of feature vectors that are applied on trained functions aiming to predict certain patterns from the characteristics of given data. In order to establish a solid foundation of the mathematical concepts provided in this paper, the central terms regression and classification need to be defined and differentiated.

Regression is a statistical term that describes a technique for estimating model parameters using a given amount of data in order to optimize the parameters such that they approximate the given data in an optimal way. This is achieved by modeling the relationship between a dependent variable and one or more explanatory variables [7].

Rather than optimizing parameters to approximate an estimation expression for a given amount of data, the concept of classification in this statistical context comprises the categorisation of given observations into predefined classes. The classification is based on previous observations and therefore the concept aims to train a mathematical function such that the probability or likelihood for a given observation belonging to the assigned class is maximized. The terms probability and likelihood are two central aspects when dealing with classification algorithms as each of them represents a distinct characterization of the models that are used to solve the problem of classification.

Both linear classification and regression find broad applications in research and industry and are used in a variety of disciplines and scientific fields, ranging from linguistic research such as text or document classification (e.g. [12]) to biological and medical applications dealing with topics such as cancer research (e.g. [9]). One particular class of linear classification models, the *support vector machine* classifiers [16], are widely-used in contemporary research scenarios related to image recognition and classification (e.g. [4]). Linear regression algorithms find numerous applications in research dealing with financial predictions and analyses such as stock market forecasting (e.g. [2]). However, the main goal of this paper is to provide a mathematical overview of the basic concepts associated with linear classification and regression tasks and does therefore not discuss the application of the illustrated approaches in the context of a specific scenario.

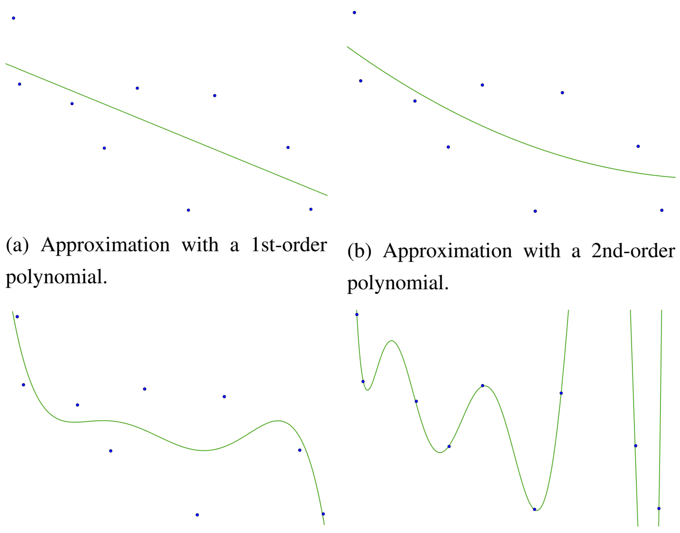
The rest of this paper is structured as follows: first, we provide an overview of and insight into the theoretical description and mathematical derivation of linear models for regression problems in section II. This part covers both the description of the theoretical concepts as well as an examination of related approaches illustrating the realization of such concepts. Section III then provides a mathematical examination of linear approaches for classification and undermines the theoretical concepts with an example for a common linear classification method, the *logistic regression* algorithm. Finally, we conclude our main findings and provide a summary of this paper's principal subjects in section IV.

II. Linear Models for regression

The initial conceptual background of linear models for regression analyses is based on the statistical concept of linear regression. Given an $(M-1)$-dimensional input vector $\mathbf{x} = \{x_1, ..., x_{M-1}\}$, linear regression uses a predefined weight vector $\mathbf{w} = \{w_0, ..., w_{M-1}\}$ such that

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_{M-1} x_{M-1}, \tag{1}$$

where the output $y(\mathbf{x}, \mathbf{w})$ denotes the approximated value based on the weighted inputs. This model is called linear regression as it utilizes a linear combination of the given input variables x_i. However, considering the weight parameters w_i as simple variables implies a significant limitation on the model as in this case equation (1) simply describes a linear function that approximates the given input data using the weight parameters w_i [5].



(a) Approximation with a 1st-order polynomial.

(b) Approximation with a 2nd-order polynomial.

(c) Approximation with a 6th-order polynomial.

(d) Approximation with an 8th-order polynomial.

Fig. 1: Polynomial approximation for a sample of $n = 9$ data points with four different polynomials. The figure shows that a higher polynomial degree increases the approximation accuracy, but also overfits the training data.

Hence, this model needs to be extended such that the linear combination of inputs is extended by a set of non-linear functions ϕ weighted with the parameters $\mathbf{w} = \{w_0, ..., w_{M-1}\}$, leading to

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \tag{2}$$

where M denotes the total number of parameters of the model and $\boldsymbol{\phi} = (\phi_0, ..., \phi_{M-1})^\top$ describes an M-dimensional vector of non-linear functions. Here $\phi_0(\mathbf{x}) = 1$ conventionally holds for all inputs $\mathbf{x}$.

The non-linear functions ϕ_j are known as *basis functions* and allow the model $y(\mathbf{x}, \mathbf{w})$ to be a linear combination of non-linear functions. The parameter w_0 serves as a fixed offset and is often referred to as the *bias parameter* [5].

In order to illustrate this basic model with an example we refer to the concept of polynomial regression, where a given function is approximated using a polynomial of degree $M-1$ [5]. Using equation (2) this can be achieved by choosing the basis functions $\phi_j(x) = x^j$. Figure 1 shows an example of a polynomial regression for $n = 9$ data points. Subfigures (c) and (d) clearly illustrate that the higher the polynomial degree, the better the approximation. Nonetheless, the figure also illustrates that polynomials whose degree is close or equal to the total amount of the given data points tend to overfit the approximation. Another decisive disadvantage of polynomial regression is that the ϕ_j are global functions of the input variables and hence changes in one input region automatically affect all the other regions [13].

A. Sum of least squares

Equations (1) and (2) raise the question of how to choose the optimal weight parameters w_j with $j \in \{0, ..., M-1\}$ for a given approximation problem. A widely-used approach is the method of *least squares*. Let $\mathbf{x}_i$ with $i \in \{1, ..., N\}$ denote a vector of feature measurements and let z_i be the given target values. The method then aims to minimize the *residual sum of squares* (RSS) by optimizing the w_j such that

$$\begin{aligned} RSS(\mathbf{w}) &= \sum_{i=1}^{N} (z_i - y(\mathbf{x}_i, \mathbf{w}))^2 \\ &= \sum_{i=1}^{N} \left(z_i - w_0 - \sum_{j=1}^{M-1} x_{i_j} w_j \right)^2 \end{aligned}$$

is minimized. RSS assumes the target values z_i to be approximated by $y(\mathbf{x}_i, \mathbf{w})$. Hence the target values can be expressed as

$$z_i = y(\mathbf{x}_i, \mathbf{w}) + \epsilon, \tag{3}$$

where ϵ denotes Gaussian noise[1] representing the approximation error [5]. However, in order to solve $RSS(\mathbf{w})$, the expression can be simplified by introducing an $N \times M$ matrix $\mathbf{X}$ representing each input vector $\mathbf{x}_i$ as a row, where $x_{i_0} = 1$. Employing this matrix adjusts the residual sum of squares such that

$$RSS(\mathbf{w}) = (\mathbf{z} - \mathbf{X}\mathbf{w})^\top (\mathbf{z} - \mathbf{X}\mathbf{w}), \tag{4}$$

where $\mathbf{z}$ denotes an N-dimensional vector of target values. Now we can minimize this function by computing the first and second derivatives of equation (4) with respect to $\mathbf{w}$:

$$\frac{\partial RSS}{\partial \mathbf{w}} = -2\mathbf{X}^\top (\mathbf{z} - \mathbf{X}\mathbf{w}) \qquad \frac{\partial^2 RSS}{\partial \mathbf{w} \partial \mathbf{w}^\top} = 2\mathbf{X}^\top \mathbf{X}$$

and by setting the first derivative to zero and solving for $\mathbf{w}$ we get

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{z}, \tag{5}$$

with $\widehat{\mathbf{w}}$ denoting the weight parameters that minimize RSS. Equation (5) assumes $\mathbf{X}^\top \mathbf{X}$ to be positive definite which is

[1] In statistical terminology, noise is a measure for the unconsidered variance of a given amount of data. Gaussian noise therefore means that the statistical noise follows a Gaussian distribution.

the case if and only if $\mathbf{X}$ has full column rank. In case $\mathbf{X}^\top \mathbf{X}$ is singular $\widehat{\mathbf{w}}$ would not be clearly defined. This mostly occurs in cases where the given data inputs are redundantly designed, and a common way to solve this is to adjust or delete the redundant columns in $\mathbf{X}$ [10].

B. Maximum likelihood estimation (MLE)

A commonly used technique for estimating the weight parameters of a statistical linear model is called *maximum likelihood estimation* (MLE). MLE was firstly introduced and developed by Fisher in 1922 and is considered as "one of the most important developments in 20th century statistics" [1]. The concept comprises the optimization of the model parameters such that the likelihood of a computed prediction based on given observations and the model parameters is maximized. In its most basic form, MLE is defined as follows. Let $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ denote a given dataset and $\mathbf{w}$ the parameters of our given model. Then

$$\widehat{\mathbf{w}} \triangleq \arg\max_{\mathbf{w}} \log p(\mathcal{D} \,|\, \mathbf{w}) \tag{6}$$

describes the maximum likelihood estimation of the given model. The equation aims to find the set of parameters $\mathbf{w}$ maximizing the probability that our model approximates $\mathcal{D}$ with $\mathbf{w}$. A helpful characteristic used in the mathematical field of probability theory that finds strong application in this method is the consideration of *independent and identically distributed events*. This concept assumes all given probabilistic events to be independent and to follow an identical probability distribution [13], and hence

$$p(\mathcal{D} \,|\, \mathbf{w}) = \prod_{i=1}^{N} p(z_i \,|\, \mathbf{x}_i, \mathbf{w}) \tag{7}$$

holds for the given target values z_i. Using the strictly monotonic characteristic of the log function[2], we can simplify this equation such that

$$\log p(\mathcal{D} \,|\, \mathbf{w}) = \sum_{i=1}^{N} \log p(z_i \,|\, \mathbf{x}_i, \mathbf{w}).$$

However, it is common to consider the negative log likelihood (NLL) instead of the positive and minimze this expression instead of maximizing the positive one. This approach is motivated by the fact that many optimization software packages are designed to minimize functions instead of maximizing them [13]. Therefore, our final likelihood estimation is

$$NLL(\mathbf{w}) \triangleq -\sum_{i=1}^{N} \log p(z_i \,|\, \mathbf{x}_i, \mathbf{w}).$$

When applying MLE to our initial situation described in equation (7), we are interested in maximizing $p(z \,|\, \mathbf{x}, \mathbf{w})$, i.e. the probability of approximating the target values z with our weights $\mathbf{w}$ and the given input vectors $\mathbf{x}$. In section II-A we assumed the noise ϵ of our linear model to follow a Gaussian distribution. Therefore,

$$p(z \,|\, \mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(z \,|\, y(\mathbf{x}, \mathbf{w}), \beta^{-1}) \tag{8}$$

holds for our initial problem, where $\mathcal{N}$ denotes the Gaussian distribution function with the inverse variance β [5]. For a set of input vectors $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and the corresponding target values $\mathbf{z} = \{z_1, ..., z_N\}$ the probability equation results in

$$p(\mathbf{z} \,|\, \mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^{N} \mathcal{N}(z_i \,|\, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}).$$

Applying the log in the same fashion as above leads to

$$\begin{aligned}\log p(\mathbf{z} \,|\, \mathbf{X}, \mathbf{w}, \beta) &= \sum_{i=1}^{N} \log \mathcal{N}(z_i \,|\, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1}) \\ &= \frac{N}{2} \cdot \log \beta - \frac{N}{2} \cdot \log(2\pi) - \beta \cdot E(\mathbf{w}),\end{aligned}$$

with $E(\mathbf{w})$ denoting a measurement function for the prediction error [5] which is defined as

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (z_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))^2. \tag{9}$$

The gradient of this log likelihood function can then be described as

$$\nabla \log p(\mathbf{z} \,|\, \mathbf{X}, \mathbf{w}, \beta) = \beta \cdot \sum_{i=1}^{N} (z_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i)) \cdot \boldsymbol{\phi}(\mathbf{x}_i)^\top$$

and setting the gradient to zero and solving for $\mathbf{w}$ results in

$$\mathbf{w}_{ML} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \mathbf{z}, \tag{10}$$

where $\mathbf{w}_{ML}$ denotes the optimized vector of parameters for a given model (i.e. the weight vector for which the likelihood of approximating the given dataset $\mathcal{D}$ is maximized) and $\boldsymbol{\Phi}$ is called the *design matrix* described as

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Equation (10) is referred to as the *normal equation* for the least squares problem [5].

C. Regularization using ridge regression

Regression models have shown to entail a certain degree of being unsatisfactory when its parameters are estimated by the discussed residual sum of squares model as this approach tends to overfit the training data [11]. A common way to prevent linear models from overfitting the input data is by adding a regularization term that aims to compensate the biased prediction.

A known technique for minimizing the residual sum of squares whilst reducing the degree of overfitting is called *ridge regression*. To our knowledge, the term ridge regression and its

[2]This characteristic implies that the $\arg\max$ does not change when applying the log to a given function.

corresponding concept were initially presented by Hoerl and Kennard [11] in 1970. In its basic idea, the approach consists of adding the sum of squares of the model's weight parameters

$$\frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{11}$$

as a regularization term, where $||\mathbf{w}||_2^2 = \sum_j w_j^2 = \mathbf{w}^\top \mathbf{w}$ and λ denotes a weight for the regularization term that controls the regularizer's relevance with regards to the model's error function. Adding this term to the error function $E(\mathbf{w})$ as described in section II-B results in the combination of equation (9) and equation (11) such that

$$E_R(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(z_i - \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{12}$$

denotes the regularized total error function [5]. Equation (12) now clearly illustrates that the added regularization term penalizes the model's error function. Assume our trained model overfits the training data, and the optimized weight parameters result in large values. Then the sum of squares of the weight parameters impacts the model's total error function in a way such that the minimized error (which is expected to be small due to the overfitting) is penalized. The regularization term then represents a relatively large value and thus has a substantial impact on the error minimization and the corresponding optimized weight parameters. On the other hand, if the model does not overfit the training data and the weight parameters are expected to be relatively small, then the regularization term does not influence the model's total error function in such a decisive manner and the optimal weight parameters are mainly determined by the model's sum of squares function.

In order to determine the optimal weight parameters for the regularized model, equation (12) is solved in the same fashion as in section II-B, leading to the weight parameter optimization function

$$\widehat{\mathbf{w}}_{ridge} = (\lambda I + \boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \mathbf{z}, \tag{13}$$

where I denotes the identity matrix.

Equation (13) illustrates the impact of the weight term λ in the context of this regularization approach. In comparison to the parameter optimization solution described in equation (10) in section II-B, adding the λ to the parameter optimization function in this regularization approach results in a higher likelihood of $(\lambda I + \boldsymbol{\Phi}^\top \boldsymbol{\Phi})$ being invertible compared to $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ [13]. Furthermore, utilizing this particular regularizer described in equation (11) has the decisive advantage that the corresponding error is still a quadratic function such that its minimization term can be specified in closed form. In a statistical terminology, this regularizer demonstrates an example of a *parameter shrinkage* method as it aims to decrease the parameters towards zero [5].

III. Linear models for classification

Contrary to the models described in the previous section, linear models for classification are trained on discrete datasets and aim to categorize given data inputs into predefined classes. This can be achieved by separating the given data into different regions that are characterized by the properties of the data residing within the corresponding region. However, an important characteristic of linear models for classification is that their decision boundaries are described in a linear fashion [10].

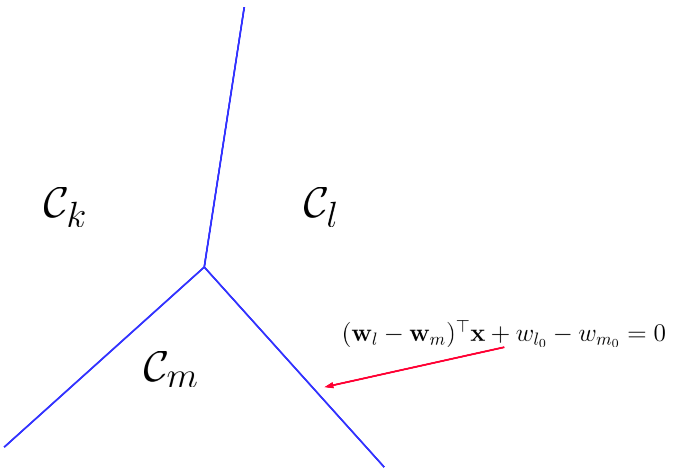

$\mathcal{C}_k$
$\mathcal{C}_l$
$\mathcal{C}_m$
$(\mathbf{w}_l - \mathbf{w}_m)^\top \mathbf{x} + w_{l_0} - w_{m_0} = 0$

Fig. 2: Illustration of a 3-class-discriminant function. The decision boundaries are always connected, making it impossible to find regions that are not assigned to a predefined class. The figure also shows the correlation between equation (16) and the corresponding decision boundary between two classes $\mathcal{C}_l$ and $\mathcal{C}_m$.

A. Discriminant functions

In order to categorize a given D-dimensional input vector $\mathbf{x}$ of a given dataset $\mathcal{D}$ into one of k predefined classes $\mathcal{C}_k$, classification models utilize a set of functions called *discriminators*. While there exist both linear and non-linear discriminant functions, we limit our focus on linear discriminant functions as those are the functions used for linear classification models [5]. The simplest class of discriminators classifies input vectors into one of two given classes $\mathcal{C}_i$ and $\mathcal{C}_j$ using a linear discriminant function

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + w_0 \tag{14}$$

with a vector of weight parameters $\mathbf{w}$ and w_0 being a bias parameter. Using equation (14), $\mathbf{x}$ is assigned to class $\mathcal{C}_i$ if $y(\mathbf{x}) \geq 0$ and to class $\mathcal{C}_j$ instead. This implies that the given decision boundary for the specified model is described by a $(D-1)$-dimensional hyperplane H with

$$H := \{\mathbf{x} \in \mathcal{D} : y(\mathbf{x}) = 0\}.$$

An intuitive approach to extend the described concept to $k > 2$ classes might be to employ $k-1$ classifiers for which each covers a separation between two classes. However, in 1973 Duda and Hart ([8], cited in [5]) showed that this given approach referred to as *one-versus-the-rest* classification can result in classification problems because the decision boundaries would always create an undefined region, i.e. a region that does not belong to a predefined class.

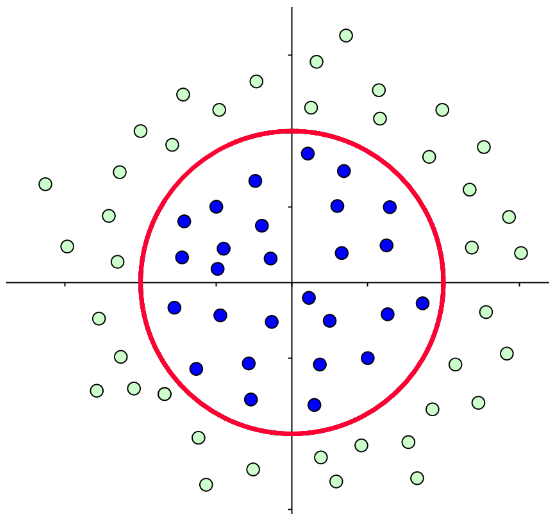


Fig. 3: Illustration of a two-class classification problem where the classes are not linearly separable. In this particular case, the function $\boldsymbol{\phi}$ (in red) serving as a decision boundary is $\boldsymbol{\phi} : (x_1, x_2)^\top \mapsto (r \cdot \cos(x_1), r \cdot \sin(x_2))^\top$, $r \in \mathbb{R}$.

In order to prevent the model from facing such classification difficulties we can utilize one single discriminant function covering all k classes. This can be modeled by introducing a discriminator $y_k(\mathbf{x})$ consisting of k linear functions

$$y_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x} + w_{k_0}. \tag{15}$$

A given input vector $\mathbf{x}$ is then classified into a class $\mathcal{C}_j$ if and only if $y_j(\mathbf{x}) > y_i(\mathbf{x})$ for all $i \neq j$. Hence the decision boundary between two classes $\mathcal{C}_i$ and $\mathcal{C}_j$ is then modeled by $y_j(\mathbf{x}) = y_i(\mathbf{x})$ which can be transformed such that

$$(\mathbf{w}_j - \mathbf{w}_i)^\top \mathbf{x} + w_{j_0} - w_{i_0} = 0. \tag{16}$$

Figure 2 illustrates the k-class-discriminator for a classification problem with $k = 3$. The decision boundaries are always connected to each other which guarantees that the algorithm assigns any input to a predefined class [5].
There exists a strong similarity between the two-class case described in equation (14) and the multi-class case such that the generalized approach applies for a two-class problem as well.

However, it is often not the case that the data is clearly arranged such that categorizing the observations into predefined classes is possible using a linear hyperplane. A common way to solve this problem is to integrate a set of fixed non-linear functions into the discriminant function so that the decision boundaries can be modeled in a non-linear fashion.
Figure 3 illustrates an example of a given scenario where the observations cannot be separated into their corresponding classes by using a linear decision boundary. A widely-used approach for selecting appropriate non-linear functions is by employing a set of predefined basis functions as defined in section II. The following approach relies on the idea of linearly combining basis functions in order to provide a clear non-linear separation of the given data.

B. The perceptron algorithm

A well-known model for providing a discriminant function consisting of the linear combination of non-linear functions for a two-class separation is called the *perceptron algorithm*, introduced by Rosenblatt in 1958 [15]. The perceptron classifies a given input vector $\mathbf{x}$ by firstly transforming it using a vector of fixed non-linear functions $\boldsymbol{\phi}(\mathbf{x})$ with $\phi_0(\mathbf{x}) = 1$ for the bias parameter w_0. The resulting vector is then applied to a generalized linear model

$$y(\mathbf{x}) = f(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})),$$

where $f(t)$ is being referred to as the *non-linear activation function*. This function then serves as the model's classifier as it maps a given input t to an output $f(t) \in \{-1, 1\}$, where each element denotes a predicted class. Hence

$$f(t) = \begin{cases} +1 & \text{if } t \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

The above model raises the question of how to identify the optimal weight parameters $\mathbf{w}$ for the perceptron algorithm. One approach to optimize these parameters is to apply an error function that is referred to as the *perceptron criterion* [5], which we will not discuss in detail in the course of this paper.

C. Probabilistic generative models

Instead of optimizing the weight parameters of given classification models, probabilistic generative models aim to learn a classifier by optimizing the probability $p(\mathbf{x}, \mathbf{z})$ with $\mathbf{x}$ representing the given inputs and $\mathbf{z}$ their corresponding class labels. The label with the highest probability for a given input is then computed by applying Bayes' theorem[3] on the probability $p(\mathbf{z} \,|\, \mathbf{x})$ [14]. In order to demonstrate the functionality of this concept, we consider a classification problem consisting of two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. Given a set of data inputs $\mathbf{x}$, we can predict $p(\mathcal{C}_1 \,|\, \mathbf{x})$ using Bayes' theorem such that

$$\begin{aligned} p(\mathcal{C}_1 \,|\, \mathbf{x}) &= \frac{p(\mathbf{x} \,|\, \mathcal{C}_1) p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_1) p(\mathcal{C}_1) + p(\mathbf{x} \,|\, \mathcal{C}_2) p(\mathcal{C}_2)} \\ &= \frac{1}{1 + e^{-a}} \\ &= \sigma(a), \end{aligned}$$

where

$$a = \log\left(\frac{p(\mathbf{x} \,|\, \mathcal{C}_1) p(\mathcal{C}_1)}{p(\mathbf{x} \,|\, \mathcal{C}_2) p(\mathcal{C}_2)}\right)$$

and σ describes the *sigmoid function*

$$\sigma(a) = \frac{1}{1 + e^{-a}}. \tag{17}$$

Based on the calculated probabilities it is then possible to predict the class to which the given data input $\mathbf{x}$ most likely

[3]Bayes' theorem: for a set of disjunct samples $A_1, ..., A_n$ and a given sample B the probability $p(A_i|B), i \in \{1, ..., n\}$ can be computed as follows: $p(A_i|B) = \frac{p(B|A_i) \cdot p(A_i)}{\sum_{j=1}^n p(B|A_j) \cdot p(A_j)}$.

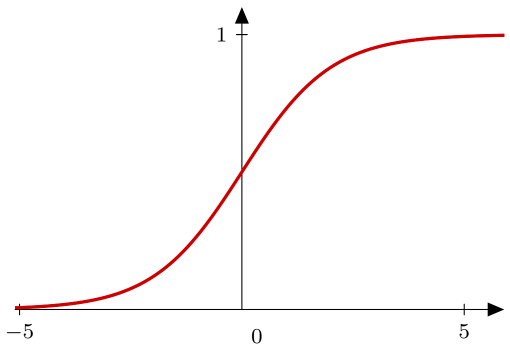


Fig. 4: The sigmoid function σ (in red) "squashes" a given input x such that $\sigma(x) \in (0,1)$ holds for all x, making it possible to treat the estimations like probabilistic values.

belongs to [5]. Figure 4 describes the sigmoid function σ and shows how it maps a computed value to a value between zero and one such that the computed result can be interpreted as a probability.

In the case of a multi-class classification problem this approach can be extended accordingly, leading to a logistic sigmoid function that is generalized for a multi-class problem. In the literature, this function is often referred to as *normalized exponential* or *softmax* (e.g. [5]). However, in the course of this paper we do not illustrate this approach in further detail.

D. Probabilistic discriminative models

In contrast to generative models, probabilistic discriminative classifiers compute the probability posterior $p(\mathbf{z} \mid \mathbf{x})$ directly instead of applying Bayes' theorem. This approach is motivated by the observation that the generative approach generalizes the model as an intermediate step before solving the actual problem ([16], cited in [14]). In the example described in section III-C, computing $p(\mathbf{x} \mid \mathbf{z})$ would be such an intermediate step towards generalizing the actual problem.

Furthermore, discriminative models typically do not contain as many parameters that need to be specified as the generative approach [5]. One disadvantage of discriminative models, however, is that they require only little knowledge of how the given amount of data is distributed and are therefore considered as "'black-box' classifiers" [3].

In order to directly compute the class prediction with the maximum likelihood estimation, discriminative models follow the approach of maximizing a likelihood function that is defined on $p(\mathcal{C}_k \mid \mathbf{x})$. The detailed procedure of how to maximize a likelihood function with a given posterior is described in section II-B. The classification of new data points is then achieved by finding the class label for which the probability

$$p(\mathcal{C}_k \mid \mathbf{x}, \theta^{opt}_{\mathcal{C}_k|x})$$

has its maximum value. In this notation, the term $\theta^{opt}_{\mathcal{C}_k|x}$ represents the optimal model parameters (as described in section II-B) for predicting class $\mathcal{C}_k$ given the data input $\mathbf{x}$.

E. Logistic regression

A commonly used example for a probabilistic discriminative model is called *logistic regression*. This concept can be considered as a generalization of linear regression to a (binary) classification problem. To our knowledge, this concept was initially introduced by Cox in 1958 [6]. Logistic regression algorithms are mostly used for solving binary classification problems. Therefore, we only focus on the application of this algorithm for two-class classification scenarios. As we have seen in section II-A (see equation (3)), it is generally assumed that the data noise in linear regression models follows a Gaussian distribution. In order to transform linear regression to a binary classification problem, it is suitable to replace the assumption of a Gaussian distribution of the given data with a Bernoulli distribution

$$\text{Ber}(n) = p^n(1-p)^{1-n},$$

where $n \in \{0,1\}$. This replacement is motivated by the assumption that a Bernoulli distribution approximates the data better than a Gaussian for a binary classification problem [13]. Apart from that, logistic regression takes a similar form as linear regression models as it aims to approximate the given data based on a learned weight vector $\mathbf{w}$. The only decisive difference is that after applying a weighted linear combination on the data input, it needs to be ensured that $\mu(x) \in [0,1]$ holds for a given output transformation function μ and for all inputs x such that the computed result can be considered as a probabilistic value. A widely-used output transformation function is the sigmoid function σ as described in equation (17) as it maps the computed result to a value between zero and one. Logistic regression then assigns a class $\mathcal{C}_k$ to an input vector $\mathbf{x}$ based on the probability

$$p(\mathcal{C}_k \mid \mathbf{x}, \mathbf{w}) = \text{Ber}(\mathcal{C}_k \mid \sigma(\mathbf{w}^\top \mathbf{x})). \tag{18}$$

Computing the class prediction based on the highest probability is then achieved by applying the concept of maximum likelihood estimation on equation (18).

IV. Conclusion

In this paper, we examined a set of basic theoretical concepts of linear methods for classification and regression problems that are employed by a variety of supervised learning algorithms used in contemporary data-driven scenarios in research and industry. We have shown that for both regression and classification models the concept of maximum likelihood estimation plays an essential role for the derivation and the functionality of such models.

For linear classification models we differentiated between probabilistic generative and probabilistic discriminative models and demonstrated the functionality of discriminative models by illustrating them with an example for a commonly used supervised learning classifier, the *logistic regression* algorithm. Considering this particular algorithm also demonstrated the correlation between classification and regression models and provided an example of how linear classifiers can be derived from regression models.

References

[1] J. Aldrich, 1997. *R.A. Fisher and the Making of Maximum Likelihood 1912 - 1922*. Statistical Science, Vol. 12, No. 3, 162-176.

[2] E. Altay and M. H. Satman, 2005. *Stock Market Forecasting: Artificial Neural Network and Linear Regression Comparison in an Emerging Market*. Journal of Financial Management and Analysis, Vol. 18, No. 2, 18-33.

[3] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press. 2012.

[4] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman, 1997. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer. 2006.

[6] D. R. Cox, 1958. *The regression analysis of binary sequences*. Journal of the Royal Statistical Society, Vol. XX, No. 2.

[7] M. J. Crawley, *The R Book*. John Wiley and Sonst Ltd., The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England. 2007.

[8] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley. 1973.

[9] I. Guyon, J. Weston and S. Barnhill, 2002. *Gene Selection for Cancer Classification using Support Vector Machines*. Machine Learning, 46, 389-422.

[10] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*. Second edition. Springer. 2009.

[11] A. E. Hoerl and R. W. Kennard, 1970. *Ridge Regression: Biased Estimation for Nonorthogonal Problems*. Technometrics, Vol. 12, No. 1.

[12] R. Mihalcea and C. Strapparava, 2009. *The lie detector: Explorations in the automatic recognition of deceptive language.* Proceedings of the Association for Computational Linguistics, 309-312.

[13] K. Murphy, *Machine Learning - A Probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts, London, England. 2012.

[14] A. Y. Ng and M. I. Jordan, 2002. *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive bayes*. Advances in Neural Information Processing Systems.

[15] F. Rosenblatt, 1958. *The Perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, Vol. 65, No. 6.

[16] V. N. Vapnik, *Statistical Learning Theory*. John Wiley and Sons. 1998.