CHAPTER TWO

BACKGROUND

Contents

2.1 SVM error function .................................................. 7
    2.1.1 Linear SVMs and separable data ......................... 7
    2.1.2 Non-separable data ........................................ 9
    2.1.3 Non-linear extension ..................................... 10

2.2 SVM hyperparameters ............................................. 11
    2.2.1 Estimating the generalization error .................... 12
    2.2.2 Minimizing the generalization error ................... 14

2.3 Approaches to solve the SVM primal/dual optimization problem ............. 17

The SVM is a popular pattern recognition algorithm [3], which was first introduced in 1979 by Vapnik [19] for separable datasets, and later refined together with Cortes [20] to handle the case for non-separable datasets as well.

A short overview of SVMs will be given. The overview starts with SVMs in their most basic form, where separating hyperplanes can only be found if the data is linearly separable (Section 2.1.1). The introduction of slack variables is then discussed in Section 2.1.2, as these allow SVMs to find separating hyperplanes even when datasets are not linearly separable. Section 2.1.3 continues with a description of the kernel trick, which allow SVMs to construct non-linear decision boundaries. For a more comprehensive overview of SVMs in general,
the reader is referred to an excellent tutorial by Burges [21] which describes SVMs, as well as the underlying theory, in detail.

SVM hyperparameters are discussed in Section 2.2. Validation functions which are employed to approximate the generalization error (and hence evaluate particular SVM hyperparameters) are discussed in Section 2.2.1. Specific approaches to minimizing these estimates are described in Section 2.2.2. This survey is concluded in Section 2.3 with an overview of recent approaches to optimizing SVM training in general.

2.1 SVM ERROR FUNCTION

2.1.1 LINEAR SVMS AND SEPARABLE DATA

If a dataset is linearly separable, an SVM will construct a hyperplane that separates the corresponding classes by maximizing the margin $\rho$ between them. The margin is defined as the distance from the hyperplane to the two closest points from the respective classes. Mathematically, we derive the the margin from the inequality

$$y_i(\omega'x_i + \omega_0) \geq 1, \forall i, i = 1, \ldots, N \tag{2.1}$$

by considering the hyperplanes at the equality. In Eq. 2.1, $y_i$ is the class label (+1 or -1), $N$ is the number of training samples, $\omega$ is a vector normal to the hyperplane and $\omega_0$ is defined so that the perpendicular distance from the hyperplane to the origin is expressed as $|\omega_0|/\|\omega\|$.

The margin is deduced from Eq. 2.1 to be

$$\frac{2}{\|\omega\|} \tag{2.2}$$

and for reasons elaborated on in Chapter 5, the probability of an error on unseen data can be minimized by maximizing the margin (hence minimizing $\|\omega\|$). Minimizing $\|\omega\|$ is equivalent to minimizing $\frac{1}{2}\|\omega\|^2$, with the latter form preferred since the problem then becomes a quadratic programming optimization problem.

The above optimization problem can be reformulated as a Lagrangian one. This is useful because the constraints in Eq. 2.1 will only appear as constraints on the Lagrange multipliers and also because the training data will be represented only as dot products between vectors [21]. Specifically, the constraints are multiplied by Lagrange multipliers $\alpha_i$ and subtracted from the original objective function to form a Lagrange function. The primal form of the Lagrangian $L_P$ is given by
By substituting into $L_P$ the results of setting the partial derivatives of $L_P$ with respect to $\omega$ and $\omega_0$ to zero, the dual formulation $L_D$ is obtained. $L_D$ is often easier to solve numerically. Solving for $\frac{\partial L_P}{\partial \omega} = 0$ results in

$$\omega = \sum_i \alpha_i y_i x_i$$ \hspace{1cm} (2.4)$$

and $\frac{\partial L_P}{\partial \omega_0} = 0$ gives

$$\sum_i \alpha_i y_i = 0.$$ \hspace{1cm} (2.5)

Substituting Eq. 2.4 and Eq. 2.5 into $L_P$ (Eq. 2.3) gives $L_D$,

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j,$$ \hspace{1cm} (2.6)

which is then to be maximized subject to $\alpha_i \geq 0$ and Eq. 2.5.

The solution of constrained optimization problems with linear constraints and a quadratic objective function is a well-studied problem, and efficient solutions can be found by consideration of the Karush-Kuhn-Tucker (KKT) conditions [22, 23]. These are conditions which are necessary and sufficient for the solution of an optimization problem subject to inequality and equality constraints [13]. By finding a solution that satisfies the KKT conditions, the SVM problem itself is thus also solved [21, 24]. The KKT conditions are [13]:

$$\frac{\partial L_P}{\partial \omega} = \omega - \sum_i \alpha_i y_i x_i = 0$$ \hspace{1cm} (2.7)

$$\frac{\partial L_P}{\partial \omega_0} = - \sum_i \alpha_i y_i = 0$$ \hspace{1cm} (2.8)

$$y_i (\omega' x_i + \omega_0) - 1 \geq 0$$ \hspace{1cm} (2.9)

$$\alpha_i \geq 0$$ \hspace{1cm} (2.10)
2.1.2 NON-SEPARABLE DATA

To handle the case of non-separable data, slack variables $\xi_i$ were introduced to the error function [20]. These variables allow misclassification while the margin between the remaining samples is maximized. The misclassifications are represented as a sum of slack variables, with a regularization parameter $C$ controlling the amount of misclassification. (In fact, this term also encompasses points which lie closer to the separating hyperplane than to the margin). Training an SVM capable of handling non-separable data entails optimizing the following error function:

$$E_{\text{svm}} = \frac{1}{2} \omega^T \omega + C \sum_i \xi_i$$

subject to the constraints

$$y_i (\omega^T x_i + \omega_0) \geq 1 - \xi_i$$
$$\xi_i \geq 0, i = 1, ..., n.$$  \hspace{1cm} (2.13)

This formulation of an SVM is also known as the 1-norm soft-margin SVM. Other formulations exist, for example the 2-norm soft-margin SVM

$$E_{\text{svm}} = \frac{1}{2} \omega^T \omega + C \sum_i \xi_i^2$$

For the purposes of this thesis, we will only focus on 1-norm soft-margin SVMs.

It is easy to show that the slack variables do not appear in the dual formulation of the Lagrangian; rather, the regularization parameter becomes an upper bound on the Lagrange multipliers. $L_P$ for the soft-margin SVM is given by (from [21])

$$L_P = \frac{1}{2} \|\omega\|^2 + C \sum_i \xi_i - \sum_{i=1}^I \alpha_i (y_i (x_i \cdot \omega + \omega_0) - 1 + \xi_i) - \sum \beta_i \xi_i.$$  \hspace{1cm} (2.15)

The $\beta_i$ are Lagrange multipliers which ensure that the slack variables are positive.

Taking partial derivatives as before, solving for $\frac{\partial L_P}{\partial \omega} = 0$ and $\frac{\partial L_P}{\partial \omega_0} = 0$ again gives Eq. 2.4 and Eq. 2.5 respectively. In addition, solving for $\frac{\partial L_P}{\partial \xi} = 0$ gives

$$\alpha_i = C - \beta_i, \forall i.$$  \hspace{1cm} (2.16)
By substituting Eqs. 2.4, 2.5 and 2.16 into \( L_P \) (Eq. 2.15) and simplifying, we again get Eq. 2.6.

Efficient algorithms for finding solutions to Eq. 2.6 that satisfy the KKT conditions have been developed. We briefly summarize the sequential minimal optimization (SMO) algorithm [25] in Algorithm 1. SMO is particularly well suited to this task, and is used in several popular SVM packages [26, 27].

**Algorithm 1 SVM using SMO**

\[ \forall i, j : \alpha_{ij} (t) = 1 \{ \alpha \text{ commonly used to refer to Lagrange multipliers in SVM literature} \} \]

express error function in dual form

repeat
for all training samples do
mark all samples which violate KKT conditions
end for
while any sample in non-bound set violates KKT conditions do
select first Lagrange multiplier \( \alpha_1 \) from list that violates KKT conditions
if \( E_1 > 0 \) then
\( \alpha_2 \) for which \( E_2 = \text{minimum} \) (cached error values of non-bound set)
else
\( \alpha_2 \) for which \( E_2 = \text{maximum} \) (cached error values of non-bound set)
end if
solve for \((\alpha_1, \alpha_2)\)
if \( \text{stepsize == 0} \) then
search until \( \alpha_2 \) found for which optimization generates a positive step size
if no \( \alpha_2 \) found that generates positive stepsize then
skip example and move to next
end if
end if
end while
until all Lagrange multipliers satisfy KKT conditions

2.1.3 NON-LINEAR EXTENSION

When the dot product in Eq. 2.12 is expanded in terms of the \( \alpha_i \), we find that all the feature vectors occur only as inner products with one another – that is,

\[
\frac{1}{2} \omega' \omega = \sum_{i \in SV} \sum_{j \in SV} \alpha_i \alpha_j y_i y_j x'_i x_j
\]  

(2.17)

where \( i \in SV \) denotes the set of SVs satisfying \( 0 < \alpha_i \leq C \).

Now, to create more general classifiers than the linear classifiers considered up to this point, a standard approach is to expand the feature vector \( x \) into a higher-dimensional space.
– for example, by concatenating non-linear combinations of feature values along with the original feature values. By replacing the inner product in Eq. 2.12 with a more general function of the feature vectors (as long as this function obeys Mercer’s conditions, see [21] for details), we get:

\[
\frac{1}{2} \omega' \omega = \sum_{i \in SV} \sum_{j \in SV} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \tag{2.18}
\]

This process has become sufficiently widespread to be known as the “kernel trick”, with the RBF kernel shown to produce excellent performance in various situations [11, 16, 28].

### 2.2 SVM HYPERPARAMETERS

The kernel trick and the ability to handle non-separable datasets are two important aspects of SVMs. Both of these aspects introduce additional free parameters which need to be set: the soft-margin formulation requires the regularization parameter \( C \) to be set, while most kernels (the linear kernel is one exception) require so-called kernel parameters to be set. An example of a kernel parameter is the kernel width \( \gamma \) in the RBF kernel

\[
K(x_i, x_j)_{RBF} = e^{-\gamma \|x_i - x_j\|^2} \tag{2.19}
\]

Together, the regularization parameter \( C \) and the kernel parameters are referred to as the SVM hyperparameters [29]. It is well known that the performance of SVMs depend crucially on both the particular kernel function used and the corresponding choice of hyperparameter values [30, 31]. Selecting a particular kernel and tuning the hyperparameter values is often referred to as model selection in literature [32, 33, 34]. Selecting kernels automatically is outside the scope of this thesis, as the main focus will be on RBF kernels (RBF kernels are a popular choice, which allows the SVM to create non-linear decision boundaries [16, 28]). A big part of this thesis thus focuses on the second part of model selection, which entails training of the hyperparameters for RBF kernels.\(^1\)

Finding good values for the SVM hyperparameters is important, since the hyperparameter values have a significant impact on the performance of an SVM [35, 30, 31]. Good values are typically found by minimizing some estimate of the generalization error. Efficiently predicting the generalization error of SVMs and minimizing this estimate are both

---

\(^1\)We refer to “tuning of the hyperparameters” as “training of hyperparameters”. This is appropriate since both the Lagrange multipliers as well as the hyperparameters are part of the same optimization process (minimizing some estimate of the generalization error). The former is simply part of the inner loop of the optimization process and the latter part of the outer loop.
topics which have received considerable attention over the past few decades and both are still active research areas [36, 37, 3, 31]. In Section 2.2.1, we will review ways in which the generalization error is being estimated. Different approaches to minimizing this estimate will then be discussed in Section 2.2.2. Section 2.2.2 is important, since our main contribution is specifically related to efficient training of the SVM hyperparameters. We also present empirical evidence (see Chapter 5) that not all hyperparameters are necessary, specifically for non-separable datasets.

2.2.1 ESTIMATING THE GENERALIZATION ERROR

An accurate estimate or prediction of the generalization error is necessary for finding good hyperparameter values [28]. One of the most common estimates of the generalization error is the k-fold cross-validation error [38, 39, 40, 30]. This estimate is calculated by dividing the available training data into k disjoint sets (the process of dividing the data into the disjoint sets can be either random, or based on a sampling strategy where the original distribution of the classes are retained, for example). One of the sets is then held out as the validation set, while the remaining k - 1 sets are used to train a classifier. This classifier is then evaluated on the held-out set. This process is repeated k times, with the k results then combined by for example averaging, to create a single estimate of the generalization error.

In the case where k is set to the number of available training samples n, the estimate is also known as the leave-one-out (LOO) cross-validation error. The LOO estimate is known to be an almost unbiased estimate of the generalization error [36, 37, 28]. It is thus reasonable to expect that hyperparameter values which optimize the LOO estimate will allow the SVM to generalize well on unseen data. However, this estimate is very expensive to calculate and hence not practical to use for even moderately sized datasets [30].

Since the LOO cross-validation error is a good estimate of the true generalization error, much work has been done to identify theoretical bounds on this error, with the aim of creating an estimate which can be computed much more efficiently. Two theoretical bounds on the SVM error which have received much attention in literature, specifically for 2-norm soft-margin or hard-margin SVMs, are the radius/margin bound [37, 35, 34, 3] and the span bound [41, 37, 32, 36, 3]. Both are upper bounds on the LOO estimate: the radius/margin bound is based on the ratio of the radius $R$ of a sphere that encompasses all training vectors in feature space and the perpendicular distance from the maximal margin $\rho$ to the closest training vector [37]. The bound $T$ is given by

$$T = \frac{R^2}{N\rho^2} \quad (2.20)$$
with $N$ the number of training samples.

The span bound, on the other hand, is based on the span of the SVs [32, 36]. The span $S_p$ is related to the exact number of errors made by the LOO estimate under the assumption that the set of SVs do not change during the LOO procedure. Vapnik and Chapelle [36] showed that the radius of the smallest sphere which encompasses all training points bounds $S_p$, hence the span bound:

$$T \leq \frac{4R^2}{\rho^2}.$$  \hspace{1cm} (2.21)

While variants of these bounds exist for standard 1-norm SVMs, they do not give satisfactory results (for the radius/margin bound; a term needs to be added, which depends on $C$, which leads to results that are comparable but not as good as results achievable with 2-norm SVMs [42, 38, 43, 3]). A variant of the span bound treats all bounded SVs as errors and hence provides only a loose upper bound on the LOO estimate [38, 3]). The span bound is also expensive to compute [38].

Several other bounds have been proposed and used to formulate validation functions. A ratio of the number of SVs and the total number of training examples $\frac{\#SVs}{N}$ also provides an upper bound on the LOO error [37, 43]. $T = \frac{\#SVs}{N}$ does not work as well as the $k$-fold cross-validation estimate [44] and is also not differentiable, which makes it less relevant for gradient based approaches to minimizing the generalization estimate [43] (these approaches will be discussed in Section 2.2.1). Other bounds that have been investigated, but found to be inferior to the cv estimate, are the Xi-Alpha bound, generalized approximate cv error and the Vapnik-Chervonenkis (VC) bound [38].

Another class of measures that are used to tune the SVM hyperparameter values (and are hence heuristics for approximating the generalization error) are based on maximizing class separability. The class separability measure was proposed to train the kernel parameters for a 2-norm soft-margin SVM [45]. Here class separability is defined in terms of within and between class scatter. A very similar approach was proposed in [46] whereby hyperparameter values are chosen to maximize the distance between two classes (DBTC), with the distance computed between class means. The main disadvantage of these measures is that the regularization parameter $C$ is not directly related to the class separability and hence still needs to be set according to another measure ([45], for example used optimal values of $C$ as found by [35], using the radius/margin bound).

Work has also been done on multiple criteria, as a single criterion may not be sufficient when the classes are unbalanced [33]. The importance of each class can be controlled in several ways: a weighted error rate can serve as the validation function [47] or alternatively, the F-measure (the harmonic mean of the precision and the recall) could be employed [47].
Maximizing the area under the corresponding receiver operating characteristic (ROC) curve, or maximization of the precision-recall break-even point have also been suggested [47]. For the purposes of this thesis, we will not consider problems that are severely unbalanced.

A recent approach by [3] is based on maximizing an estimate of a likelihood function for the SVM hyperparameters. This validation function is based on Platt's likelihood estimate [48], where a sigmoid function is used to estimate the conditional probability of a correct sample, given the SVM prediction. Glasmachers [3] optimized a CV based objective function using gradient-based methods to find the sigmoid parameters.

Many competitive approaches have been described in literature to date, without one validation function clearly being superior to the rest. The likelihood function proposed by [3] is very competitive, given the results they provide. Furthermore, it is very well documented and a thorough and well-applied implementation is available on-line together with their paper. For these reasons, the likelihood estimate [3] will be used as a comparison to work presented in this thesis.

While many alternatives have been mentioned in this section, the \( k \)-fold cross-validation measure remains a competitive predictor of the generalization error [28, 38] and hence a popular approach to estimating the generalization error [28, 38, 44, 39, 40, 30]. In this thesis, we will use the 10-fold cross-validation estimate to approximate the generalization error.

2.2.2 MINIMIZING THE GENERALIZATION ERROR

A validation function provides a way to quantify the approximation of the generalization error, given some parameters. In this section, we explore popular approaches to minimizing the validation function, which in turn implies minimization of the generalization error.

2.2.2.1 EXHAUSTIVE SEARCH

Minimizing an estimate of the generalization error is a problem that is still receiving much attention, among others because it is a time-consuming part of the overall SVM training process. A common approach to finding good SVM hyperparameter values involves an exhaustive search (a line or a grid search for one or two hyperparameters respectively) on the log-scale over a predefined range of parameter values. While very easy to implement, an exhaustive grid search has several drawbacks. It is not a feasible approach when more than two parameters need to be trained because of the large number of resulting parameter combinations that need to be considered [39, 43, 37] (the parameters cannot be considered to be independent of one another [34] and hence need to be tuned together). Another drawback
is the use of a discrete parameter space, since it is possible that the true global minimum will not be one of the discrete parameter combinations considered. If k-fold cv is used as a validation function, an exhaustive search is also computationally very expensive, not only because many parameter combinations have to be evaluated, but also because many poor (and hence wasteful) parameter combinations will be evaluated. Despite these drawbacks and in view of a lack of well-understood alternatives that are easy to implement, grid search remains a popular approach to training hyperparameter values [2].

2.2.2.2 FOCUSED EXHAUSTIVE SEARCH

Much work has been done to improve the efficiency of the traditional grid search. Recent approaches include deterministic focused grid search (DFGS) and annealed focused grid search (AFGS) [1]. DFGS starts from the outer grid points and systematically narrows the search region by (1) centering each new search on the best previous parameter choice and (2) halving the grid search area. DFGS is closely related to the procedure described in [49]. While being much faster than a conventional grid-search, DFGS is still unfeasible when searching for more than two parameters. AFGS attempts to be feasible for more than two parameters by randomly selecting a fixed fraction of the grid points and evaluating them. While the annealed approach is more successful at handling a larger number of parameters, convergence is not guaranteed. Furthermore, both approaches are susceptible to becoming stuck in local minima.

Another promising heuristic aimed at identifying a good area of the grid to search for hyperparameter values (specifically for an RBF kernel\(^2\)) was proposed by [16]. They made interesting observations about the behavior of SVMs at extreme values of the hyperparameters, which prompted them to come up with the following heuristic: fix the value of \(C^*\) to the optimal value of \(C\) obtained by training a linear SVM. Then search along the line defined by 
\[
\log(\gamma^{-1}) = \log C - \log C^*
\]
for the optimal \((C, \gamma)\) pair.

2.2.2.3 GRADIENT DESCENT

Gradient descent is a general approach to training hyperparameters that directly addresses some of the disadvantages of exhaustive search [37]. Gradient descent is attractive because it has been found to be much faster than grid search and it also allows the hyperparameter values to be determined with greater precision [47]. Furthermore, since steps are taken in parameter space towards the optimal value, it makes training many parameters simultaneously feasible. There are however several disadvantages to a gradient based approach: the

\(^2\)The RBF kernel can also be referred to as a Gaussian kernel when \(\gamma^{-1} = 2\sigma^2\)
validation function needs to be differentiable, both with regard to the kernel and the regularization parameters [29]. (This also implies that the kernel itself needs to be differentiable.) The differentiable validation function requirement automatically excludes some reasonable validation functions, for example the number of SVs [43]. In order to be differentiable, smoothed approximations of existing validation functions have to be formulated. A differentiable version of the LOO estimate was proposed in [37]. It entailed computing the inversion of a kernel sub-matrix corresponding to the SVs, which is expensive from both a computational as well as a storage point of view [47]. An efficient way to compute the gradient of a wide range of smoothed validation functions and kernels was however proposed in [47]. The gradient of these smoothed approximations is not guaranteed to point to the true minimum of the original validation function though [29]. Yet another disadvantage is that different validation functions may depend strongly on the initialization of the hyperparameters [33, 49] and there may also be local minima at inferior parameter choices [50, 34].

Despite the disadvantages mentioned, gradient based approaches are an attractive alternative to exhaustive search. Gradient based approaches are attractive not only because the hyperparameter search can be performed much more efficiently, but because more complex kernels (for example automatic relevance detection (ARD) kernels) are feasible. More complex kernels are however difficult to understand and care must be taken to avoid overfitting; severe overfitting was observed on a text-classification problem with thousands of features in [47].

Popular techniques for solving the corresponding optimization problem include quasi-Newton approaches [35], for example the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [51, 52, 53, 54, 55] and Rprop [4, 3].

2.2.2.4 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are another approach to minimizing validation functions. These algorithms directly minimize the validation function using iterative, random operations inspired by biological evolution theory [29]. New parameter combinations are identified by a random mutation operation, given existing good parameter combinations. A selection strategy then follows where the best combinations are retained, according to the validation function (also referred to as a fitness function).

Evolutionary algorithms are a popular alternative to gradient based approaches, since no gradient information is required (and hence approximation of the validation functions by

---

2 ARD kernels have a γ for each dimension, which is an impossible task for all but the simplest problems for exhaustive approaches.
The covariance matrix adaptation evolution strategy (CMA-ES) [56] has been successfully applied to search for approximate hyperparameter values [29, 3]. CMA-ES adapts the hyperparameters in two ways: intermediate recombination and additive Gaussian mutation [29]. Intermediate recombination is performed by calculating the mean of the current population, while mutation is performed by adding a vector of randomly generated values from a Gaussian distribution with zero mean. The covariance matrix of this Gaussian distribution is also continually updated to maximize the likelihood of previous steps that have been successful.

CMA-ES is considered fairly complex, difficult to implement and the parameterization of the algorithm itself is also not straightforward [1]. It has however been found to perform well on objection functions, which have plateaus for multiple dimensional parameter spaces, such as cv and LOO [3].

Other popular evolutionary algorithms that have been applied to hyperparameter training for SVMs include genetic algorithms (GA) [57, 44, 50].

### 2.3 APPROACHES TO SOLVE THE SVM PRIMAL/DUAL OPTIMIZATION PROBLEM

We conclude this survey by mentioning approaches that solve the primal/dual problem significantly faster than previous methods, for example SMO [25]. These methods are relevant and should be noted, since a significant improvement in the time that it takes to optimize the primal/dual form of the SVM may have a significant impact on hyperparameter training as well.

Optimizing SVMs using SGD has been considered in [58]. A thorough analysis of the convergence properties of SGD was performed, as well as early stopping as an alternative to explicit regularization. We follow a similar approach whereby efficient early stopping criteria are explored as an alternative to explicit regularization. It was also found, as is well known in literature, that the learning rate remains a difficult parameter to set. To alleviate this problem, stochastic meta descent (SMD) was proposed [59] where a simultaneous SGD is performed on the step size as well. It was shown to accelerate on-line SVM convergence significantly and will be a useful technique once generalized to other kernel algorithms as well.

PEGASOS (Primal Estimated sub-Gradient Solver for SVM) [60, 61] is an SVM solver that alternates between SGD steps and projection steps. It is a very promising technique for large datasets, in that the method’s runtime does not depend directly on the size of the
training set. In later work, it was even shown that training time should decrease as the data set size increases [61]. This is achieved by ignoring the classic optimization problem and rather focusing on finding a good predictor. While it is claimed that PEGASOS should work with non-linear kernels as well, there is no indication that this is yet the case.