Vision, Modeling, and Visualization (2019)
H.-J. Schulz, M. Teschner, and M. Wimmer (Eds.)

Open-Box Training of Kernel Support Vector Machines: Opportunities and Limitations

Mohammad Khatami and Thomas Schultz

University of Bonn, Germany

Abstract

Kernel Support Vector Machines (SVMs) are widely used for supervised classification, and have achieved state-of-the-art performance in numerous applications. We aim to further increase their efficacy by allowing a human operator to steer their training process. To this end, we identify several possible strategies for meaningful human intervention in their training, propose a corresponding visual analytics workflow, and implement it in a prototype system. Initial results from two users, on data from three different domains suggest that, in addition to facilitating better insight into the data and into the classifier's decision process, visual analytics can increase the efficacy of Support Vector Machines when the data available for training has a low number of samples, is unbalanced with respect to the different classes, contains outliers, irrelevant features, or mislabeled samples. However, we also discuss some limitations of improving the efficacy of supervised classification with visual analytics.

CCS Concepts
• ***Visualization application domains*** $\rightarrow$ *Visual analytics;* • ***Kernel methods*** $\rightarrow$ *Support vector machines;*

1. Introduction

Classification is the task of assigning an observation, which is commonly described by a vector of numerical features, to one of a finite number of classes. It occurs in many applications, from predicting optimal medical treatment based on genetic profiles to telling legitimate from fraudulent uses of credit cards. Supervised classification learns a function that performs this assignment based on a set of training data. It is a standard problem in machine learning [HTF11].

Visual analytics is "the science of analytical reasoning facilitated by interactive visual interfaces." [TC05]. There is a strong current interest in using visual analytics to improve supervised classification [ERT*17]. However, most work has focused on two specific families of classifiers: One are decision trees, which are relatively easy to interpret. The second one are deep neural networks, which have achieved outstanding performance on many difficult problems, due to their ability to learn suitable representations from large and high-dimensional training data. Our work is concerned with Kernel Support Vector Machines (SVMs) [SS02], an important supervised classification method that remains a method of choice when the amount of training data or computational resources are too limited to permit effective training of parameter-rich deep neural networks.

Visual analytics increases human understanding of the training data, the most relevant features, and other factors that determine the classifier's performance. In addition to this, we aim to increase SVM efficacy via human interaction. In agreement with [RFT17], we use efficacy as a generic term for desirable properties which, depending on the application, might be best quantified as accuracy, precision, recall, or area under the receiver operating characteristic (ROC) curve, while simultaneously reducing the set of features required by the classifier. Compared to decision trees, an empirical study on 121 data sets has ranked SVMs considerably higher with respect to classification accuracy [FDCBA14]. This makes it more difficult to increase their efficacy with open-box training.

It is another difficulty that SVM training is based on solving an abstract global optimization problem. Our first contribution (in Section 3) is to *analyze* this training process with respect to how a human expert can steer it in a meaningful way. This represents the first stage in developing and justifying the design of our method. In the second stage of our design, we decide which specific visual analytics techniques should be included in our system, and how they should work together. In Section 4, we base this on a specific *workflow* that integrates the previously identified opportunities for steering SVMs, and we describe our prototype implementation of a *visual analytics system* that implements this workflow.

In Section 5, we provide evidence for the usefulness of our system by comparing results from our system to three automated training strategies. Based on our findings, and on theoretical results from the machine learning literature, we *discuss benefits and limitations of open-box* SVM training.

2. Related Work

Most approaches to SVM visualization aimed to support their interpretability, but were not specifically concerned with improving

DOI: 10.2312/vmv.20191319

efficacy. For example, Caragea et al. use projection-based tours to identify variables on which the SVM relies most strongly for its classification [CCH01], and to illustrate how decision surfaces vary when randomly subsampling the training data, or changing hyperparameter values [CCWH08]. Hamel [Ham06] derives a two-dimensional visualization of support vectors and decision surfaces using self-organizing maps, to visually convey how clusters in the training data relate to the final decision. The nomograms proposed for SVM visualization by Jakulin et al. [JMD*05] provide a graphical overview of how different feature values affect the probability that the corresponding sample will be assigned to a specific class.

Similar to us, Ma et al. [MCM*17] include a human in SVM training. However, we aim to steer the training of kernel SVMs without fundamentally changing this widely used technique, while they replace kernel SVMs with local linear SVMs. Poulet [Pou04] and Do and Poulet [DP04] also involve humans in SVM training, but their goal is to achieve interactive data reduction, and to enable training of SVMs when large-scale inputs would otherwise impose unreasonable demands on memory and processing times.

Most existing visual analytics systems that perform feature selection for classification are agnostic with respect to the classifier. Examples are the SmartStripes software presented by May et al. [MBD*11], or the INFUSE system proposed by Krause et al. [KPB14]. Some has been application-specific, such as an approach to using visual analytics for constructing white matter hyperintensity classifiers by Raidou et al. [RKS*16]. Our system is generic with respect to the application, but specific to kernel SVMs. Some existing extensions and variants of SVMs are relevant to our work. Their discussion is left to Section 3.1.

Different terms have been used to describe different facets of the interplay between humans and computational systems. Computational steering describes a human intervention into a computational process, based on a visual representation of that process, and interactive feedback about the consequences of changes made by the user [vLMvW97]. This term originates from numerical simulations, but it matches our intended intervention into the SVM training process. Guidance has been characterized as “a computer-assisted process that aims to actively resolve a knowledge gap encountered by users during an interactive visual analytics session” [CGM*17]. This is another important aspect for our work: We expect that the user will not have all knowledge required for meaningful steering to begin with, and has to acquire it by interacting with our system.

3. Opportunities for Steering SVM Training

In this section, we examine the mathematical formulation of kernel SVMs with the goal of identifying opportunities for meaningful human intervention. Several textbooks describe the SVM training process in a comprehensive and accessible manner [SS02, HTF11]. Appendix A recapitulates the facts that are immediately relevant to our further discussion, and clarifies our notation. Equations with numbers in the form (A.x) are part of the appendix.

We propose three complementary strategies that allow us to influence the training in a meaningful way: Sample weighting and outlier elimination (Section 3.1), feature weighting and selection (Section 3.2), and semi-automated parameter tuning (Section 3.3). They provide the basis of our workflow and visual analytics system, which we describe in Section 4.

3.1. Sample Weighting and Outlier Elimination

By default, all samples that are used to train an SVM are given equal weight. This can be seen from the fact that, in Eq. (A.2), all slack variables ξ_i are penalized by the same factor C, independent of sample i. When given training data that is unbalanced in the sense that the two classes are represented by different numbers of samples, this will lead to a bias in favor of predicting the more frequent class. When such a bias is undesired, one strategy is to pick different values of C for samples from the two classes, based on their inverse frequencies, e.g.,

$$C^{\{+,-\}} = \frac{n^+ + n^-}{2n^{\{+,-\}}} C, \tag{1}$$

where $C^{\{+,-\}}$ is used for samples from the positive or negative class, respectively, and $n^{\{+,-\}}$ are the number of training samples from the corresponding class.

Such class weights can be generalized to individual sample weights. This opens up a way in which a human analyst can influence SVM training: If she recognizes certain training samples as atypical, or even as erroneous, their impact on the final classifier can be reduced by scaling down the penality incurred by the corresponding slack ξ_i. In the extreme case, weighting a sample by zero has the same effect as removing it from the training set.

A related trend in machine learning has been referred to as “learning using privileged information” [VV09], “learning using hidden information”, or “learning with teacher” [VVP09]. In this context, SVM+ has been developed. It accounts for additional features in the training data, access to which is “privileged” in the sense that they are not available for the test data that is to be classified later on. When selecting suitable kernels and regularization parameters, SVMs are consistent in the sense that, given an increasing number of training samples from a fixed joint distribution $P(\mathbf{x}, y)$ of features and labels, their solution converges to the theoretically optimal Bayes decision rule [Ste02]. It has been proven that a suitable encoding of prior knowledge can improve the rate of that convergence [PV10], and that the same effect can be achieved by a suitable weighting of training samples [LHS14]. In particular, atypical samples $(\mathbf{x}_i, y_i)$ with a low conditional probability $P(y = y_i|\mathbf{x}_i)$ should receive a lower weight. To our knowledge, we are the first to use this idea within a visual analytics workflow.

Atypical samples can be spotted directly by viewing the data distribution, e.g., via dimensionality reduction. In Appendix B, we list several quantities that can be visualized in our system to provide additional guidance for sample weighting and outlier removal. Such outlier scores should not be trusted “blindly”, but should rather support identification of samples that merit further investigation. The final sample weights should be assigned by considering all available information, including prior domain knowledge.

3.2. Feature Weighting and Selection

In most practical scenarios, not all features are equally important. The assignment of feature weights is another way in which a human

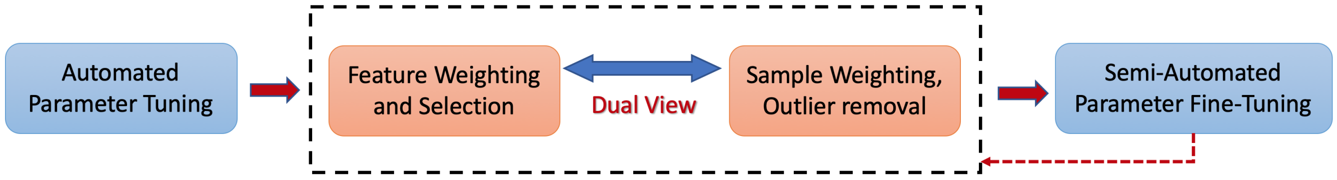

Automated Parameter Tuning
Feature Weighting and Selection
Dual View
Sample Weighting, Outlier removal
Semi-Automated Parameter Fine-Tuning

Figure 1: *All stages of our proposed workflow for open-box training of kernel SVMs involve a human expert. An initial parameter tuning is required to provide guidance for feature and sample weighting. The latter two are so tightly interwoven that we integrate them in a dual view.*

analyst can influence the training of an SVM: Scaling a feature by a factor $f > 1$ ($0 \leq f < 1$) increases (decreases) its impact. This observation is not trivial, since some other supervised classifiers, such as Random Forests [Bre01], are unaffected by such scalings. It might even seem counter-intuitive since, in Eq. (A.1), scaling a feature could be compensated by an inverse scaling of the corresponding coefficient in the weight vector **w**. However, this equivalent solution is generally not the one found by training the SVM with re-scaled features. Rather, scaling distorts the space in which Eq. (A.2) optimizes the margin, or Eq. (A.4) measures distances.

Feature weighting generalizes feature selection, which can be achieved by assigning a feature zero weight. Feature selection has been more widely considered in machine learning [GE03] and visual analytics [MBD*11, KPB14]. It is often based on feature importance scores. However, all of these are, to some extent, heuristic, and their rankings may contradict each other. We propose that visual analytics can help a human analyst make an informed choice based on investigating the relationships between multiple importance scores, and understanding the reasons for potential discrepancies between them by referring back to the original data distribution. If she finds certain features to be less relevant, but not entirely useless, the analyst can decide to assign a smaller weight to them, rather than eliminating them completely. A list of specific importance scores implemented in our system, along with a brief explanation and justification for including them, is given in Appendix C.

3.3. Parameter Tuning

Achieving good results with Radial Basis Function (RBF) kernel SVMs requires selecting an appropriate regularization weight C (Eq. (A.2)) and kernel parameter γ (Eq. (A.4)). These values are commonly estimated automatically, via a grid search with cross-validation on the training data [HCL03]. However, depending on the application requirements, there are different criteria that one might optimize, such as accuracy, precision P, recall R, and the F_1 score $F_1 = 2 \times P \times R/(P+R)$, i.e., the harmonic mean of precision and recall. In general, different parameter settings will optimize different criteria. In addition, it is informative to account for the fraction of support vectors corresponding to each setting: If it is very high, this indicates that the settings caused the SVM to memorize the data, rather than to learn a suitable abstraction.

Therefore, we propose that SVM training can benefit from having a human analyst examine the involved tradeoffs, especially when a single criterion, such as accuracy alone, might not indicate a clear and unique optimum in parameter space. Such an investigation might give a human analyst a legitimate reason to deviate from the automatically proposed settings, especially since there is no guarantee that parameters obtained from cross-validation are going to result in optimal performance also on the final test data.

4. A Proposed Workflow and Prototype System

In Section 4.1, we propose a visual analytics workflow that combines the ingredients discussed in the previous section. We then describe and justify the design of a prototype system that implements it. Our design consists of a dual view in which feature and sample weighting can be performed (Section 4.2), and a separate view for parameter tuning (Section 4.3).

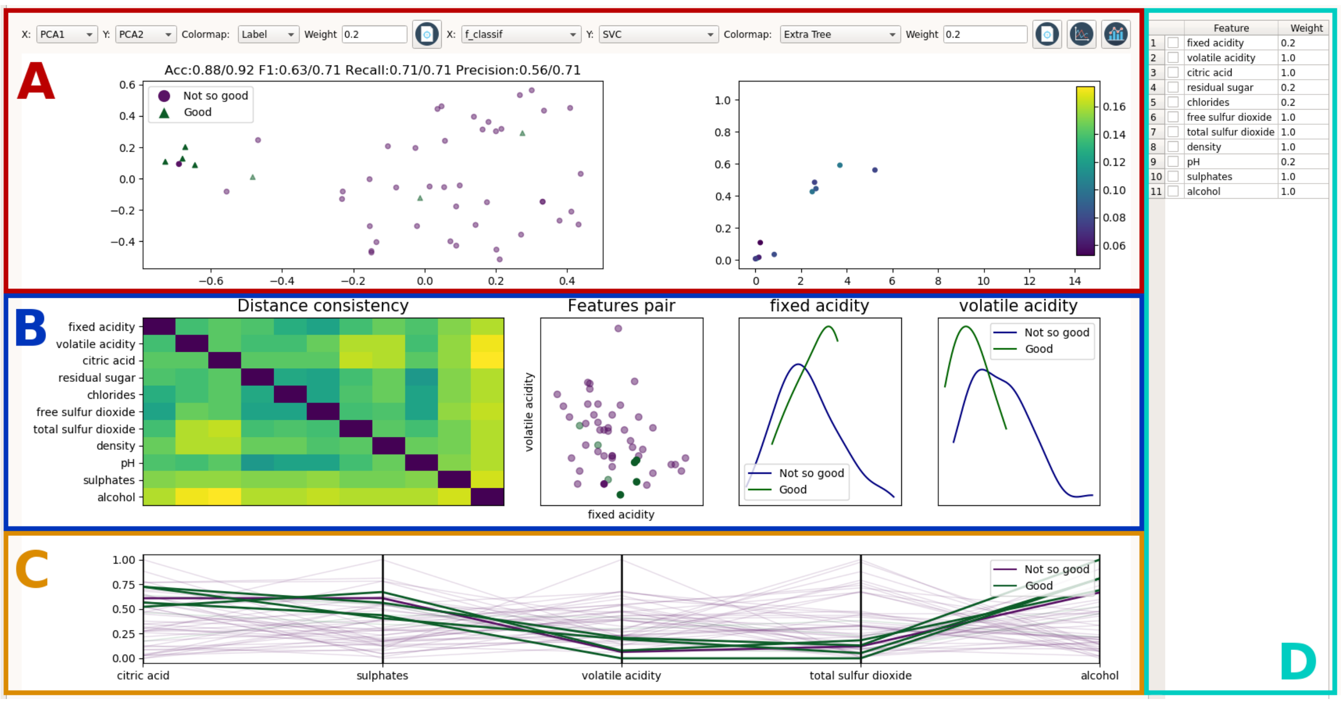
We aimed to create a feature-rich system that should provide a large degree of flexibility to users who have a good understanding of SVMs. We believe that the experience with this system can inform the decision whether and how to combine SVMs with visual analytics in specific applications, and with specific domain experts as users. In Section 6, we discuss this point in more detail.

4.1. A Workflow for Steering SVM Training

As a basis for the design of our system, we reasoned about dependencies between the building blocks identified in Section 3. This led us to a specific workflow for steering SVM training. It is depicted in Figure 1, and motivated as follows:

Parameter tuning is required as an initial step, since several elements that are used to guide feature and sample weighting are based on the internal state of a pre-trained SVM classifier, or on its kernel parameters. It can be done with an established automated grid search [HCL03]. Since removing or re-weighting features and samples is expected to change the optimum in SVM parameter space, another parameter tuning is required at least once at the end of the workflow. A semi-automated fine-tuning, which will be described in Section 4.3, can further improve results. Since some of the data views also depend on the parameters, it can make sense to alternate between feature and sample weighting on the one hand, and parameter tuning on the other, when making substantial changes.

The two core tasks within the pipeline are feature weighting and selection, as well as sample weighting and outlier removal. It was highlighted previously [MBD*11] that these two steps are intricately interwoven, and should not be strictly separated: For example, observing the effects of different feature weights on the sample distribution can provide valuable guidance on which exact setting is the most useful. In the context of training kernel SVMs, we recommend using kernel PCA [SSM97] for plotting the sample


X: PCA1 Y: PCA2 Colormap: Label Weight 0.2 X: f_classif Y: SVC Colormap: Extra Tree Weight 0.2

Acc:0.88/0.92 F1:0.63/0.71 Recall:0.71/0.71 Precision:0.56/0.71

		Feature	Weight
1	☐	fixed acidity	0.2
2	☐	volatile acidity	1.0
3	☐	citric acid	1.0
4	☐	residual sugar	0.2
5	☐	chlorides	0.2
6	☐	free sulfur dioxide	1.0
7	☐	total sulfur dioxide	1.0
8	☐	density	1.0
9	☐	pH	0.2
10	☐	sulphates	1.0
11	☐	alcohol	1.0

Figure 2: *Our visual analytics system supports feature and sample weighting with a window that integrates a customizable dual domain view of sample and feature space (A), in-depth visualizations of individual feature distributions and correlations, guided by a heatmap of distance consistency [SNLH09] (B), parallel coordinates (C), and a table that lists all features and their current weights (D).*

distribution, since it shows us the data "through the same lens" as the SVM: When using the same kernel, the projections created by kernel PCA reflect the same abstract high-dimensional feature space in which the kernel SVM optimizes the margin for classification. Therefore, it makes sense to use the (visually assessed) separability of classes in these projections as guidance for tuning feature weights [RFT17]. Moreover, exploring the data distribution with respect to specific features can help the user understand potential discrepancies between feature importance scores. During this process, some outlier samples can already become apparent, and we find it much more practical to modify their weights right away, rather than having to remember them for a later, separate stage in the pipeline.

The tight integration of feature and sample weighting implies that it can make sense to alternate between them, and there is no mandatory starting point. However, in all cases we encountered, we found it helpful to inspect the features first: When there are many irrelevant ones, removing them has a profound effect on dimensionality reduction based views and outlier scores. Based on them, the weights of atypical samples can subsequently be reduced.

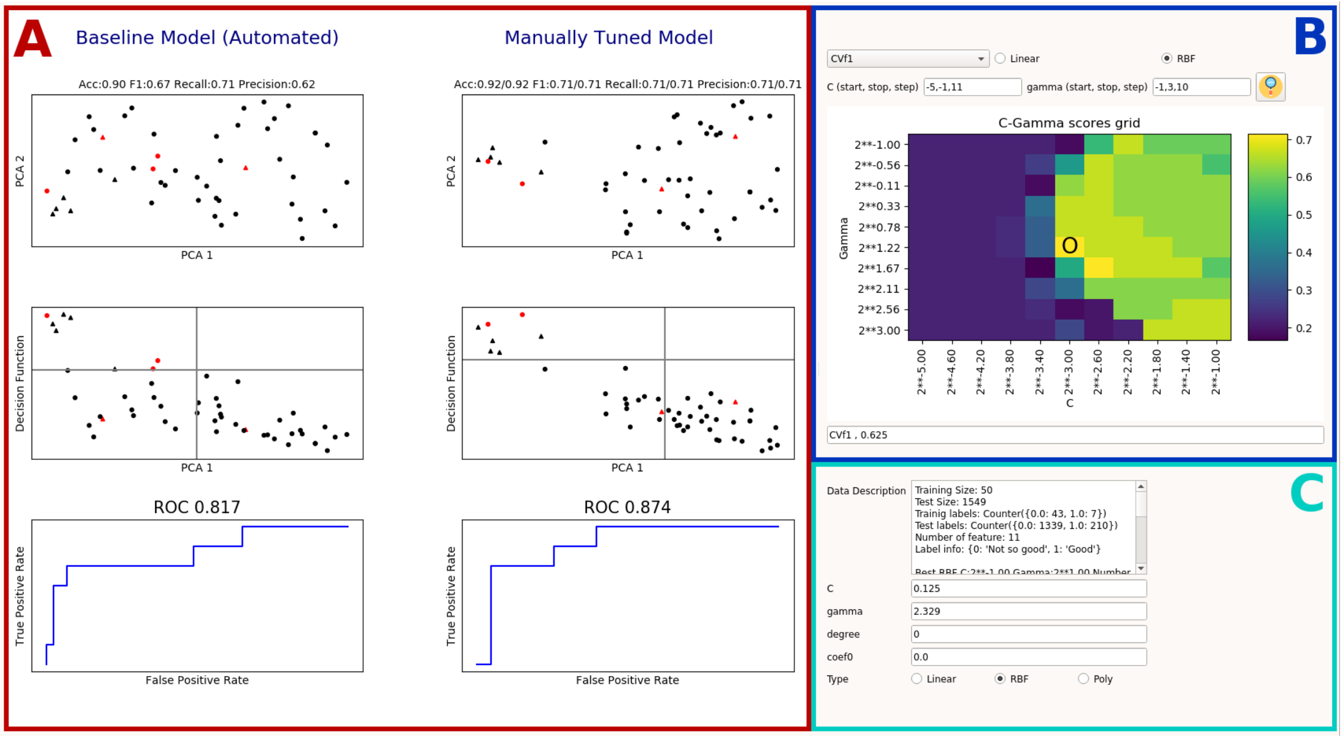
Our prototype implements this workflow with two main views: One for feature and sample weighting and selection, shown in Fig. 2, and one for parameter tuning, shown in Fig. 3. The system has been written in Python 3.7.2, based on PyQt 5.9.2 for the graphical user interface, matplotlib 3.0.2 for creating and interacting with the plots, scikit-learn 0.20.3 for the required machine learning infrastructure, libxml 2.9.9 and pandas 0.24.1 for data storage and management, numpy 1.16.2 and scipy 1.2.1 for basic scientific computing.

4.2. A Dual View for Feature and Sample Weighting

We discuss feature and sample weighting and selection first, because they are at the core of our workflow. To account for the tight coupling between them, we adapt the idea of dual visual analysis [TFH11, DWF*19], in which samples and features can be investigated simultaneously. This dual view is shown in Fig. 2 A. The effects of changes on one side are immediately reflected in an update of the other side, i.e., the kernel PCA is automatically re-computed after changing feature weights, and feature scores are updated when removing samples.

The views are customizable: Samples can be placed and colored based on different kernel PCA modes, or on any of the additional quantities described in Appendix B. Features can be placed and colored based on various importance scores, including those described in Appendix C. To review the sample and feature weights, they can also be selected for color coding or placement. Depending on the nature of the color coded quantity, our system employs matplotlib's perceptually uniform sequential (e.g., for nonnegative importance scores) or diverging colormaps (e.g., for the modified decision function). In addition, class labels are indicated with marker shape (circle and triangle), and samples can be colored based on cross-validated prediction accuracy (black for correct classification, red for incorrect), or class labels, for which we chose green and purple to avoid suggesting an ordering.

To understand why features are ranked in a certain way, we allow the analyst to refer back to the original data distribution, and to



CVf1 ( ) Linear (•) RBF

C (start, stop, step) -5,-1,11 gamma (start, stop, step) -1,3,10

CVf1 , 0.625

Data Description

```
Training Size: 50
Test Size: 1549
Trainig labels: Counter({0.0: 43, 1.0: 7})
Test labels: Counter({0.0: 1339, 1.0: 210})
Number of feature: 11
Label info: {0: 'Not so good', 1: 'Good'}

Best RBF C:2**-1.00 Gamma:2**1.00 Number
```

C	0.125
gamma	2.329
degree	0
coef0	0.0
Type	Linear [✓ RBF] Poly

Figure 3: *Our system supports parameter tuning by comparing cross-validation results from an automated machine learning pipeline to those of the manually tuned model (A). Results of cross-validation over an adaptable grid are visualized with heat maps of various criteria, such as accuracy, precision, recall, or fraction of support vectors (B). Characteristics of the data set and parameter choices are also shown (C).*

investigate correlations between pairs of features. To this end, we include scatterplots and density estimates of class-specific feature distributions (Fig. 2 B), as well as parallel coordinate plots (Fig. 2 C), because they were successfully used to steer classifier training previously [CLKP10]. They all use the same colors to convey class labels, and are linked so that a selection in any scatterplot is reflected in the others, and in the parallel coordinates. Moreover, to reduce the complexity of the parallel coordinate plot, a subset of features can be selected for inclusion, either in the features scatterplot on the right-hand side of A, or in a separate tabular overview of all features (D), which also lists the currently chosen weights.

Our initial prototype included a scatterplot matrix, but we found that it quickly became unmanageable when the number of features increased. Therefore, we instead present a feature matrix, which is shown on the left-hand side of Fig. 2 B. Clicking on an entry selects the corresponding feature pair for an in-depth investigation in the scatterplot (center) and density plots (right). To guide the user towards features that are relevant to the classification task, the feature matrix displays a heat map of distance consistency [SNLH09], a measure of how well the classes are separated in a scatterplot.

After changing feature or sample weights, our system provides immediate feedback on whether this appears to benefit the classification. For this, the SVM is automatically retrained after each modification. Accuracy, F_1, recall, and precision scores from a five-fold cross-validation are displayed above the left scatterplot in Fig. 2 A. In each case, the second number represents the most recent result, the first one the result before that. It is important to keep in mind that a decrease in cross-validation accuracy does not necessarily imply that a change was counterproductive. It might simply indicate that a given modification resulted in a need to re-tune SVM parameters. To ensure interactivity, our system does not re-tune the parameters each time. However, the user can decide to do so at any point, by switching to the parameter tuning tab, which will be described next.

4.3. Parameter Tuning View

We placed the visual interface for parameter tuning into a separate tab, shown in Fig. 3. We made this decision to avoid further increasing the complexity of the tab shown in Fig. 2, and because initializing this view is time-consuming enough that we decided to perform it only after first loading the data or on request, which is made by switching to this tab. In our experiments, it took between one and seven seconds on a single 3.4 GHz CPU core.

Semi-automated parameter tuning requires the analyst to examine the trade-off between different criteria, and to see whether there is a clear unique optimum, or several promising alternatives that may be worth exploring further. We support this with a grid search over C and γ parameters with five-fold cross-validation. We also tried splitting the data into fixed training and validation sets, and visualizing the results separately. This is faster than cross-validation and we hoped that it might support investigation of potential overfitting. However, it turned out not to be feasible when dealing with limited training data since it left an insufficient number of samples for reliable training or validation. To avoid bias, the test data on which the efficacy of open-box SVMs is finally evaluated, was not available during the visual analytics process in either case.

Accuracy, F_1 score, precision, recall, area under the ROC curve, or fraction of support vectors are visualized as a function of C and γ in the heatmap in Fig. 3 B. To keep the interface simple, we show one user-selected measure at a time. The most favorable value is highlighted with a circle. Since large parameter ranges need to be tried out, we sample them on a logarithmic scale. We follow established recommendations for the range and number of samples [HCL03], and allow the user to modify them in order to explore promising regions of parameter space in greater detail.

Parameter values can be selected by clicking on the matrix view. The corresponding result is then shown in more detail in the right column of Fig. 3 A. Again, the two values printed for each criterion compare the current result to the previous one. Classification results are shown in three ways: For the reasons given above, kernel PCA (top) is the standard sample view of our system. The central view relates the SVM decision function (i.e., Eq. (A.1) without reducing the value to its sign) to the first kernel PCA mode, indicating which samples ended up close to the decision boundary (horizontal line in that plot). Finally, an ROC curve of the classifier is shown at the bottom, indicating possible tradeoffs between true and false positive rates which could be achieved by shifting the decision boundary. The number on top of that plot is the area under the ROC curve.

The left column of Fig. 3 A shows the same plots for the automatically tuned classifier. This allows the user to assess the expected benefit of her modifications. We note that not just the colors of the samples changed between the scatterplots on the left and the right (red indicating misclassification), but also the positions in kernel PCA space, due to changes in feature weights and kernel parameters.

A separate region in this tab, Fig. 3 C, displays the current parameter values for SVM training and optionally allows the user to manually edit them. It also shows details about the current dataset.

5. Results and Discussion

5.1. Experimental Setup and Overview of Results

Similar to the closely related work by Ma et al. [MCM*17], we compare the results achieved with open-box training with those from automated training to provide evidence that our system is useful. Unlike Ma et al., who report results from a single dataset and a single human operator, we selected three datasets from different domains, and had two operators work with each, to obtain an initial impression of between-user variability.

We included multiple datasets because we expected that the extent to which a human in the loop can improve kernel SVMs depends on the characteristics of the data. In particular, we expected that the following limitations of a training dataset should provide room for improvement over automated training:

L1 Only few training samples are available.
L2 The available features differ substantially with respect to their relevance to the classification problem, especially when a large number of irrelevant features is present.
L3 Training data is noisy or contains outliers.
L4 Training data is unbalanced, so that some classes are represented by a much larger number of samples than others.

Table 1: *F_1 scores and area under the curve (AUC) achieved on data from three different domains. Methods are automated SVM training without (row 1) and with automated feature selection (rows 2/3), as well as our proposed system (rows 4/5).*

	Wine		Brain MRI		Liver	
Method	F_1	AUC	F_1	AUC	F_1	AUC
No feature selection	0.42	0.78	0.75	0.86	**0.55**	0.71
F score	0.44	0.66	0.74	0.78	0.51	0.64
Mutual information	0.47	0.72	0.75	0.79	0.48	0.61
Our system (User 1)	0.49	0.85	**0.83**	0.91	**0.55**	0.70
Our system (User 2)	**0.52**	**0.86**	0.82	**0.92**	0.54	**0.72**

While Ma et al. compare their results to those from a linear SVM, we employ three more challenging baselines: First, we followed widely used recommendations [HCL03] for automated training. They combine an RBF kernel with min-max feature normalization and a grid search over a wide range to establish suitable values of C and γ. To account for class imbalance, we set class weights according to Eq. (1), used stratified cross-validation, and optimized parameters with respect to the F_1 score. To create even more challenging baselines, we also included automated feature selection. For this, features were ranked according to the F score in one case, mutual information in the other. In either case, the optimal number of top-ranking features was estimated with stratified cross-validation. For each candidate feature set, a full grid search was run to determine SVM parameters leading to largest F_1.

The two co-authors of this work ("User 1" / "User 2") attempted to optimize SVM efficacy by independently following the above-described workflow, until they felt satisfied with the result. This was typically the case after a few minutes of interaction. We report results on the final test data, which were only available after the users decided to stop the process.

Details on the three datasets ("wine", "brain MRI", and "liver") are given in Sections 5.2– 5.4, respectively. Two example sessions are illustrated in supplementary videos. Results are summarized in Table 1. Since accuracy is not suitable for evaluating classifiers on unbalanced data, we specify the F_1 score and area under ROC curve (AUC), for independent test sets that were not available during automated feature selection or interactive tuning. For each dataset and criterion, the best result is highlighted in bold.

5.2. Wine Quality Estimation

Our first experiment used the wine quality dataset from Cortez et al. [CCA*09], which is available from the UCI Machine Learning Repository [DKT17]. Limitations **L1–L4** are addressed as follows: A small random subsample of 50 wines was used for training **(L1);** features (such as alcohol content, residual sugar, acidity) naturally

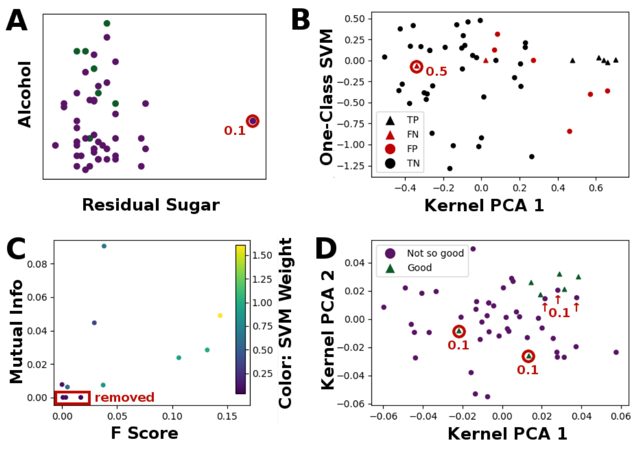


Figure 4: *User interaction with the wine quality dataset. A: Scatterplot in which User 1 assigned weight 0.1 to an outlier with respect to residual sugar. B: Scatterplot in which User 1 assigned weight 0.5 to an apparent outlier. C: Feature plot in which User 2 eliminated three features (names given in text). D: In a kernel PCA plots, User 2 reduced the weights of five atypical samples to 0.1.*

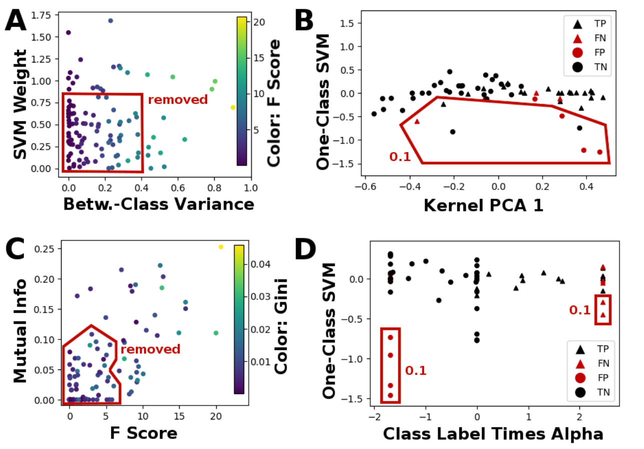


Figure 5: *User interaction with the brain MRI dataset. A: Feature plot in which User 1 removed many low-ranking features. B: Sample plot in which User 1 assigned weight 0.1 to outlier samples. C: Feature plot in which User 2 removed low-ranking features. D: Sample plot in which User 2 reduced the weight of outliers.*

varied with respect to their importance **(L2);** labels were not fully reliable due to the subjective nature of wine quality **(L3);** unbalanced classes were formed by thresholding the quality score into "high quality" (quality$>$ 6) and "medium and low quality" (quality$\leq$ 6), leading to only 14% "high quality" wines **(L4).**

Both users started with the parameters $C = 0.5, \gamma = 2$ from the automated pipeline. User 1 investigated the F scores and SVM weights of all features. Four of them (pH, residual sugar, fixed acidity, chlorides) stood out as scoring low with respect to both measures, so he decided to reduce their weights to 0.1. During this inspection, he also discovered an outlier with respect to residual sugar and reduced its weight to 0.1 (cf. Fig. 4 A). In a scatterplot of one-class SVM scores vs. PCA mode 1, he identified a misclassified sample that was far away both from other samples of the same class, and other misclassified samples (cf. Fig. 4 B). He decided to reduce its weight to 0.5. Fine-tuning the parameters led him to select the parameters $C = 0.1768, \gamma = 2$ for the final classifier.

User 2 eliminated three features (residual sugar, fixed acidity, chlorides) entirely, since they scored low according to F score, mutual information, and linear SVM weights (cf. Fig. 4 C). At this point, he decided to re-tune the parameters (to $C = 2^{13}, \gamma = 2^{-9}$) and identified five outliers in the kernel PCA plot. He reduced their weights to 0.1 (cf. Fig. 4 D). When color coding the α_i, one point stood out with an exceptionally high value. The user reduced its weight to 0.5 to avoid giving it too much influence. Fine-tuning the parameters led him to $C = 55.72, \gamma = 0.0364$.

5.3. Brain MRI Gender Classification

Our second experiment was to distinguish between men and women based on brain MRI from 145 subjects, 86 (59%) of them male, from the Human Connectome Project [SJX*13]. 59 subjects (35 male) were used for training, the remaining ones for evaluation. 110 features that reflect tissue microstructure in specific white matter bundles have been extracted using the CoBundleMAP approach [KWS19]. Here, the number of samples is naturally low **(L1)** and features vary strongly with respect to their importance **(L2).** However, labels are reliable, and class imbalance is limited.

Both users started with $C = 8, \gamma = 0.125$. Due to the overwhelming number of features, User 1 decided to remove a bulk of low-scoring ones without investigating them in detail, leaving only 30 that exhibited either high linear SVM weights or between-class variance (cf. Fig. 5 A). In a scatterplot of one-class SVM scores vs. PCA mode 1, the weights of seven atypical samples were reduced to 0.1 (cf. Fig. 5 B). The final parameter tuning led to $C = 32, \gamma = 0.5$.

User 2 reduced the number of features to 32, based on F score, mutual information, and Gini importances (cf. Fig. 5 C). He then re-tuned parameters (to $C = 2, \gamma = 0.125$) and, based on a scatterplot of α_i versus one-class SVM scores, reduced the weights of samples that were rated as strong outliers by the one-class SVM (values below -0.25), but received high α_i in the classifier, to 0.1 (cf. Fig. 5 D). Parameter fine-tuning led him to $C = 2^{13}, \gamma = 2^{-13}$.

5.4. Detection of Liver Disease

Our third dataset contained medical records of 579 patients, 414 (72%) of which suffered from liver disease [RBV11]. We split it into 290 samples (207 with liver disease) for training, 289 for testing. The goal is to detect liver disease based on 10 features that include age, gender, as well as results from several blood tests. In this case, there was a moderate class imbalance **(L4)**, but the data was not overly sparse, there was not a large number of irrelevant features, and there were no obvious issues with noise or outliers.

Both users started with $C = 32, \gamma = 2$. After investigating Gini importances, linear SVM weights, and F scores, User 1 assigned

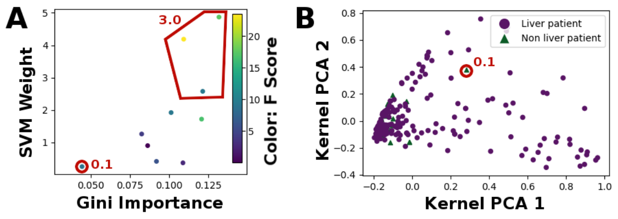


Figure 6: *User interaction with the liver dataset. A: Feature plot in which User 1 assigned weights 0.1 and 3 to the marked features, respectively (features names in text). B: Sample plot in which User 2 assigned weight 0.1 to an apparent outlier sample.*

weight 0.1 to gender, and increased weights of the high-scoring features direct bilirubin, alkaline phosphotase, and alamine aminotransferase to 3 (cf. Fig. 6 A). User 1 decided not to modify any sample weights. In the final parameter tuning, he selected $C = 8192, \gamma = 0.5$.

User 2 assigned weight 0.2 to gender, age, total proteins, albumin, and A/G ratio, based on their low F score, mutual information, and linear SVM weights. He noticed one outlier in the kernel PCA plot, and reduced its weight to 0.1 (cf. Fig. 6 B). Parameter fine-tuning led him to $C = 6208, \gamma = 0.092$.

5.5. Discussion of Experimental Results

It can be seen in Table 1 that the use of our system improved F_1 scores and AUC in two of the three investigated datasets, wine quality and brain MRI. These results are in line with our hypothesis that limitations **L1**–**L4** pose a difficulty for standard SVM training, and provide room for improvement with visual analytics. In the liver case, which suffered least from the limitations, none of the more advanced methods, neither automated feature selection nor manual steering, substantially improved over the baseline.

It might happen that human intervention *decreases* F_1, or any other quality criterion that we might have hoped to optimize. The view in Fig. 3 A provides valuable guidance to guard against this, by directly comparing the accuracy of the automated and the manually tuned classifier, estimated via cross-validation. However, the final evaluation happens on test data that was not available during training, and there can never be a guarantee that observations made on the training data will generalize to them. This limitation affects not just open-box training, but also any automated method for feature selection. In our results, examples can be seen in the liver data, where both automated feature selection methods caused a decrease of F_1 and AUC, even though the option to keep all available features, which would have produced the same result as the baseline without feature selection, was included in their solution space.

The observation that feature selection may decrease performance on the test set can be explained by the fact that it introduces additional degrees of freedom into the training process, which increases the risk of overfitting. One way to reduce this risk is to perform feature selection within a visual analytics system that encourages users to make choices not purely based on a single feature importance score, but by comparing multiple ones, combined with a visual examination of the data itself, and informed by domain knowledge.

Given the good performance of fully automated SVMs, and the fact that we can never guarantee that putting a human into the loop will lead to an improvement, one might ask whether the use of visual analytics to support supervised classification is really worthwhile. In our opinion, the answer depends on the characteristics of the data and application: The more the available data suffers from the above-mentioned limitations, the more we can hope to benefit from visual analytics in terms of prediction accuracy. The greater the practical consequences of a classifier's decision (e.g., in terms of human well-being or economic value), the higher the value of being able to make even small improvements.

Even though users did not take exactly the same steps, between-subject variability of F_1 and AUC scores was low. However, it is a limitation of our work that experiments were performed by only two operators, both of which were experts in the involved methods. We note that most related works are either vague with respect to who performed the experiments ("the user", "test persons") [Pou04, MBD*11, MCM*17] or use language that suggests that authors did the experiments themselves ("we") [DP04, RKS*16, RFT17]. Krause et al. [KPB14] work with a team of four clinical researchers, but report that they had a good understanding of the involved machine learning techniques even before they started using the visual analytics system. We believe that the current practice of having test users who are closely familiar with machine learning amounts to quantifying the benefit from visual analytics under favorable conditions. It is an interesting challenge for future work to design systems that are specifically targeted at domain experts without a strong technical background and to evaluate them with proper user studies.

6. Discussion and Conclusion

We identified opportunities for steering SVM training with visual analytics, proposed a workflow that includes them, and implemented it in a prototype system. Finally, we applied the system to three datasets with different characteristics and from different domains. Results indicate that visual analytics can improve SVM efficacy when the training data is sparse or unbalanced, or when it includes many irrelevant features, noise, or outliers. However, the benefit decreased when comparing against automated feature selection techniques. We expect that, in many use cases, increased understanding of the training data and the SVM's decision process will be a stronger motivation for using visual analytics than the increase in efficacy.

The effectiveness of our visual encodings, and interactive response of our system, is limited to datasets that do not have too many samples (in our examples, up to a few hundred). However, with increasing availability of training data, SVMs have been proven to converge to the optimal solution [Ste02] even without human intervention. Therefore, settings with limited training data are the most relevant ones for putting a human into the loop.

In the future, we are planning to study a wider range of supervised classifiers, both for the binary and the multi-class cases. We also hope to extend the design of our system to include explicit support for collaborative use by a domain expert and a data scientist.

Appendix A: Background on SVMs

In their original form, SVMs are binary linear classifiers that map a given feature vector $\mathbf{x} \in \mathbb{R}^d$ to one of two labels $y \in \{-1, 1\}$ via a weight vector $\mathbf{w} \in \mathbb{R}^d$ and bias term b:

$$y(\mathbf{x}) = \operatorname{sign}\left(\mathbf{w}^T \mathbf{x} + b\right) \tag{A.1}$$

Given training data $\{\mathbf{x}_i, y_i\}$, $i \in \{1, 2, \ldots, n\}$, soft margin SVMs obtain suitable parameters $\mathbf{w}$ and b by optimizing

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \tag{A.2}$$

while, for all i, keeping the constraints

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0. \tag{A.3}$$

The hyperparameter C in Eq. (A.2) has to be provided by the user, and balances the two competing objectives encoded in that equation: Maximizing the margin, i.e., the distance between the available training samples and the decision boundary, while minimizing the overall extent to which individual samples fall into the margin, or even get mis-classified. For the ith sample, this is encoded in the slack variable ξ_i, which is included in the optimization.

Many real-world cases require nonlinear classification. In SVMs, this is achieved via linear classification in a higher-dimensional feature space to which the inputs are mapped implicitly with a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The RBF kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \tag{A.4}$$

is a common choice.

The kernelized version of Eq. (A.1) can be written as

$$y(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{A.5}$$

which includes one Lagrangian multiplier $\alpha_i \geq 0$ per training sample. The weight vector $\mathbf{w}$ can be expressed as $\sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \cdot)$, but is now a member of an abstract higher-dimensional feature space. The decision in Eq. (A.5) only depends on those training samples $\mathbf{x}_i$ for which $\alpha_i > 0$. These are referred to as support vectors.

Appendix B: Outlier Measures

Our system implements the following outlier measures:

- A modified SVM decision function, which can be obtained from Eq. (A.1) or its kernelized counter-part Eq. (A.5): Instead of taking the sign of the expression in parentheses, we multiply it by the known label y. This way, the final value will be positive if the SVM made a correct decision, negative if the sample was misclassified. The modified decision function is evaluated using cross-validation, so that the sample for which it is computed is not included in the training set that was used to make the prediction.
- The value of α_i (cf. Eq. (A.5)) that was assigned to sample i, averaged over all cross-validation folds in which it was part of the training data. This highlights the samples that are actually used for classification, so that changing their weights is most likely to have an impact. Visualizing the product $\alpha_i y_i$ allows us to identify samples that influence the decision towards one specific class.
- A score from a one-class SVM [SWS*99], which we train separately on the training data from each of the two classes. Given a training set $\{\mathbf{x}_i\}$ that is assumed to contain independent samples from a common distribution $P(\mathbf{x})$, one-class SVMs estimate a function that is positive on a subset of feature space that contains a prespecified fraction ν of the probability mass, and which should be as "simple" or regular as possible. Thus, we can interpret samples that are assigned higher values to be closer to the "core" of the respective class distribution; lower values are more likely to be at the fringe, or outliers.

Appendix C: Feature Importance Scores

Our system implements the following feature importance scores:

- The *F score* from a one-way analysis of variance [McD14]. It is fast and simple to compute, and straightforward to interpret: It indicates whether the difference between the class-specific means of a feature is large compared to its within-class variance.
- The *mutual information* between a given feature and the class labels [KSG04]. Compared to the F score, it still works when the relationship between feature and label is more complex. However, it is more difficult to estimate from limited data.
- *SVM feature weights,* obtained from training a linear SVM. Assuming the features have been normalized initially, the absolute value of the corresponding coefficient in weight vector $\mathbf{w}$ (cf. Eq. (A.1)) is taken as an indicator of their relevance for the final decision. Unlike the F score and mutual information, this accounts for all features simultaneously, and is therefore affected by interactions between them. Unfortunately, when using a kernel, the weight vector is defined in an abstract higher-dimensional space, and can no longer serve as an indicator of feature relevance.
- Since SVM feature weights are limited to linear classification, we include *Gini importances* as a measure that accounts for nonlinear classification. It is derived from Gini impurity I_G, a measure of class label heterogeneity. Given a set of samples in which a fraction p_i, $i \in \{1, 2, \ldots, C\}$, belongs to the ith of C classes,

$$I_G(p) = \sum_{i=1}^{C} p_i(1 - p_i). \tag{A.6}$$

$I_G = 0$ when all samples have the same label. The overall decrease of Gini impurity brought by a given feature when training random forests [Bre01] or extremely randomized trees [GEW06] can be taken as an indicator of its importance.

References

[Bre01] BREIMAN L.: Random forests. *Machine Learning 45*, 1 (2001), 5–32. 3, 9

[CCA*09] CORTEZ P., CERDEIRA A., ALMEIDA F., MATOS T., REIS J.: Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems 47*, 4 (2009), 547–553. 6

[CCH01] CARAGEA D., COOK D., HONAVAR V.: Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (SIGKDD)* (2001), pp. 251–256. 2

[CCWH08] CARAGEA D., COOK D., WICKHAM H., HONAVAR V.: Visual methods for examining SVM classifiers. In *Visual Data Mining* (2008), S.J. Simo et al., (Ed.), vol. 4404 of *LNCS*, Springer, pp. 136–153. 2

[CGM*17] CENEDA D., GSCHWANDTNER T., MAY T., MIKSCH S., SCHULZ H.-J., STREIT M., TOMINSKI C.: Characterizing guidance in visual analytics. *IEEE Trans. on Visualization and Computer Graphics 23*, 1 (2017), 111–120. 2

[CLKP10] CHOO J., LEE H., KIHM J., PARK H.: iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST)* (2010), IEEE. 5

[DKT17] DHEERU D., KARRA TANISKIDOU E.: UCI machine learning repository, 2017. URL: `http://archive.ics.uci.edu/ml`. 6

[DP04] DO T.-N., POULET F.: Enhancing SVM with visualization. In *Discovery Science* (2004), Suzuki E., Arikawa S., (Eds.), vol. 3245 of *LNAI*, Springer, pp. 183–194. 2, 8

[DWF*19] DOWLING M., WENSKOVITCH J., FRY J., LEMAN S., HOUSE L., NORTH C.: SIRIUS: Dual, symmetric, interactive dimension reductions. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 172–182. 4

[ERT*17] ENDERT A., RIBARSKY W., TURKAY C., WONG B. W., NABNEY I., BLANCO I. D., ROSSI F.: The state of the art in integrating machine learning into visual analytics. *Computer Graphics Forum 36*, 8 (2017), 458–486. 1

[FDCBA14] FERNÁNDEZ-DELGADO M., CERNADAS E., BARRO S., AMORIM D.: Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research 15* (2014), 3133–3181. 1

[GE03] GUYON I., ELISSEEFF A.: An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (2003), 1157–1182. 3

[GEW06] GEURTS P., ERNST D., WEHENKEL L.: Extremely randomized trees. *Machine Learning 63*, 1 (2006), 3–42. 9

[Ham06] HAMEL L.: Visualization of support vector machines with unsupervised learning. In *Proc. IEEE Symp. on Computational Intelligence and Bioinformatics and Computational Biology (CIBCB)* (2006), pp. 1–8. 2

[HCL03] HSU C.-W., CHANG C.-C., LIN C.-J.: *A practical guide to support vector classification*. Tech. rep., Department of Computer Science, National Taiwan University, 2003. 3, 6

[HTF11] HASTIE T., TIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer Series in Statistics. Springer, 2011. 1, 2

[JMD*05] JAKULIN A., MOZINA M., DEMSAR J., BRATKO I., ZUPAN B.: Nomograms for visualizing support vector machines. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (SIGKDD)* (2005), pp. 108–117. 2

[KKEM10] KEIM D., KOHLHAMMER J., ELLIS G., MANSMANN F. (Eds.): *Mastering the information age: Solving problems with Visual Analytics*. Eurographics, 2010.

[KPB14] KRAUSE J., PERER A., BERTINI E.: INFUSE: Interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (2014), 1614–1623. 2, 3, 8

[KSG04] KRASKOV A., STÖGBAUER H., GRASSBERGER P.: Estimating mutual information. *Physical Review E 69*, 6 (2004). 9

[KWS19] KHATAMI M., WEHLER R., SCHULTZ T.: CoBundleMAP: Consistent 2D parameterization of fiber bundles across subjects and hemispheres. In *Proc. IEEE Int'l Symp. on Biomedical Imaging (ISBI)* (2019). 7

[LHS14] LAPIN M., HEIN M., SCHIELE B.: Learning using privileged information: SVM+ and weighted SVM. *Neural Networks 53* (2014), 95–108. 2

[MBD*11] MAY T., BANNACH A., DAVEY J., RUPPERT T., KOHLHAMMER J.: Guiding feature subset selection with an interactive visualization. In *Proc. IEEE Conf. on Visual Analytics Science and Technology (VAST)* (2011), pp. 111–120. 2, 3, 8

[McD14] MCDONALD J. H.: *Handbook of Biological Statistics*, 3rd ed. Sparky House Publishing, Baltimore, Maryland, 2014. 9

[MCM*17] MA Y., CHEN W., MA X., XU J., HUANG X., MACIEJEWSKI R., TUNG A. K. H.: EasySVM: A visual analysis approach for open-box support vector machines. *Computational Visual Media 3*, 2 (2017), 161–175. 2, 6, 8

[Pou04] POULET F.: SVM and graphical algorithms: a cooperative approach. In *Proc. IEEE Int'l Conf. on Data Mining (ICDM)* (2004), pp. 499–502. 2, 8

[PV10] PECHYONY D., VAPNIK V.: On the theory of learning with privileged information. In *Proc. Int'l Conf. on Neural Information Processing Systems (NIPS)* (2010), vol. 2, pp. 1894–1902. 2

[RBV11] RAMANA B. V., BABU M. S. P., VENKATESWARLU N.: A critical study of selected classification algorithms for liver disease diagnosis. *Int'l. J. of Database Management Systems 3*, 2 (2011), 101–114. 7

[RFT17] RAUBER P. E., FALCÃO A. X., TELEA A. C.: Projections as visual aids for classification system design. *Information Visualization 17*, 4 (2017), 282–305. 1, 4, 8

[RKS*16] RAIDOU R. G., KUIJF H. J., SEPASIAN N., PEZZOTTI N., BOUVY W. H., BREEUWER M., VILANOVA A.: Employing visual analytics to aid the design of white matter hyperintensity classifiers. In *Proc. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2016, pp. 97–105. 2, 8

[SJX*13] SOTIROPOULOS S. N., JBABDI S., XU J., ANDERSSON J. L., MOELLER S., AUERBACH E. J., GLASSER M. F., HERNANDEZ M., SAPIRO G., JENKINSON M., FEINBERG D. A., YACOUB E., LENGLET C., ESSEN D. C. V., UGURBIL K., BEHRENS T. E.: Advances in diffusion MRI acquisition and processing in the human connectome project. *NeuroImage 80* (2013), 125–143. 7

[SNLH09] SIPS M., NEUBERT B., LEWIS J. P., HANRAHAN P.: Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum 28*, 3 (2009), 831–838. 4, 5

[SS02] SCHÖLKOPF B., SMOLA A. J.: *Learning with Kernels*. MIT Press, 2002. 1, 2

[SSM97] SCHÖLKOPF B., SMOLA A. J., MÜLLER K.: Kernel principal component analysis. In *Int'l Conf. on Artificial Neural Networks (ICANN)* (1997), pp. 583–588. 3

[Ste02] STEINWART I.: Support vector machines are universally consistent. *Journal of Complexity 18*, 3 (2002), 768–791. 2, 8

[SWS*99] SCHÖLKOPF B., WILLIAMSON R. C., SMOLA A. J., SHAWE-TAYLOR J., PLATT J. C.: Support vector method for novelty detection. In *Advances in Neural Information Processing Systems (NIPS)* (1999), pp. 582–588. 9

[TC05] THOMAS J. J., COOK K. A.: *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005. 1

[TFH11] TURKAY C., FILZMOSER P., HAUSER H.: Brushing dimensions—a dual visual analysis model for high-dimensional data. *IEEE Trans. on Visualization and Computer Graphics 17*, 12 (2011), 2591–2599. 4

[vLMvW97] VAN LIERE R., MULDER J. D., VAN WIJK J. J.: Computational steering. *Future Generation Computer Systems 12* (1997), 441–450. 2

[VV09] VAPNIK V., VASHIST A.: A new learning paradigm: Learning using privileged information. *Neural Networks 22*, 5 (2009), 544–557. 2

[VVP09] VAPNIK V., VASHIST A., PAVLOVITCH N.: Learning using hidden information (learning with teacher). In *Proc. Int'l J. Conf. on Neural Networks (IJCNN)* (2009), IEEE, pp. 1252–1259. 2