EasySVM: A Visual Analysis Approach for Open-Box Support Vector Machines

Category: Technical Paper

Abstract— Support vector machines (SVMs) are supervised learning models traditionally employed for classification and regression analysis. Despite being one of the most popular classification models because of its strong performance empirically, understanding the knowledge captured in a SVM remains difficult. SVMs are typically applied in a black-box manner where the details of parameters tuning, training and even the final constructed model is hidden from the users. This is natural since these details are often complex and difficult to understand without proper visualization tools. However, such an approach often bring about various problems including trial-and-error tuning and suspicious users who are forced to trust these models blindly. The contribution of this paper is a visual analysis approach for building SVMs in an open-box manner. Our goal is to improve an analyst's understanding of the SVM modeling process through a suite of visualization techniques that allow users to have full interactive visual control over the entire SVM training process. Our visual exploration tools have been developed to enable intuitive parameter tuning, training data manipulation, and rule extraction as part of the SVM training process.

Index Terms—Support vector machines, rule extraction, visual classification, high-dimensional visualization, visual analysis

1 Introduction

A support vector machine (SVM) is a supervised learning method that is widely used in a variety of application areas. While SVMs have been shown to have high accuracy in classification, they also face a variety of challenges when we want to use them for data analytics. First, conventional SVM approaches are "black-box" schemes in which details of the model construction and prediction process are hidden from the user. The user simply provides the SVM with a training dataset, relevant input parameters, and a model will be constructed for making prediction of unlabelled data. Other outputs that can be retrieved from the trained SVM model are a vector that represents the feature weights, and a set of training data instances called *support vectors*. These outputs are unable to provide any insights for domain users who want to better understand the reasoning behind certain classification results. Thus, while the automatic "black-box" model releases the user from laborious interactions, it may hinder the user in terms of understanding and insight. Another issue that makes gaining insight from SVMs difficult is the use of non-linear kernels which typically improve the classification accuracy. There is however a lack of systematic and effective method to select the correct kernel functions and input parameters to give good classification result [3]. As such, tuning parameters in this black-box environment can be extremely difficult and time-consuming for the user. In addition, the non-linear characteristic further complicates the difficulties of interpreting the classification process.

This paper presents our efforts in opening the black-box of model building and knowledge extraction from SVMs. We propose our design and implementation of a web-based visual analysis system that supports model building using SVMs. The main contributions include:

- An interactive visualization method for exploring data instances, linear SVMs and their relationships;
- A visual analysis approach for building local linear SVM models that overcomes the non-linearity of the underlying dataset, and;
- A visual rule extraction method that allows the user to extract rules that best interpret the models.

2 EasySVM: Open-box Visual Modeling of SVMs

Figure 1 shows the analytical scheme for our solution which consist of three components:

Open-box Visual Modeling for SVMs To enable the user to quickly determine item relationships, we map the data instances and SVM models into a 2-D plane with an orthogonal projection. We design an interactive projection control scheme to support the flexible exploration of the dataset and the model from different aspects.

Local SVM Building through Visualization Once the data instances and models are visualized, the user may recognize non-linearities within the model space. The underlying SVM model can then be progressively approximated with a sequence of linear localized SVM models. A suite of visual linkage and comparison operations are integrated to enable analysts to explore relations between data instances and SVM models as well as manipulate local models.

Visual Rule Extraction Rule extraction is interactively performed along each axis. The user can either select segments on the axes or select regions from the projected results of data instances and the SVM models. Each extracted rule can be represented with a hierarchical tree structure.

This scheme is encapsulated in our EasySVM system, a web-based interactive visualization system depicted in Figure 2. The system consists of four main views: a scatterplot view, a projection management view, a dimension selection view, and a rule extraction view.

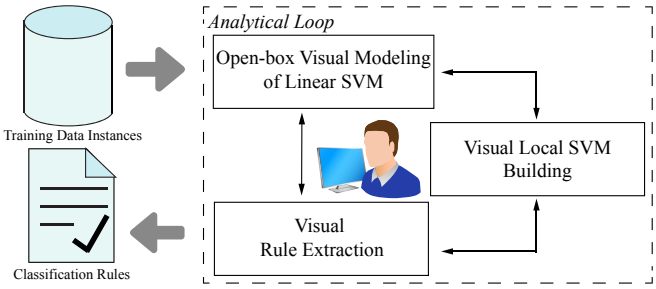

Fig. 1. Overview of our approach.

2.1 Open-box Visual Modeling of Linear SVM

The traditional SVM model building process can be summarized into following fours steps: 1) preprocess the data, 2) select parameters, 3) train the model, and 4) validate the training result. If the validation result cannot pass the test (e.g. low prediction accuracy on the test dataset), the process will restart from step 2 again until it meets the user's requirement. In our visual model building process, each of the model building steps aforementioned is enhanced by

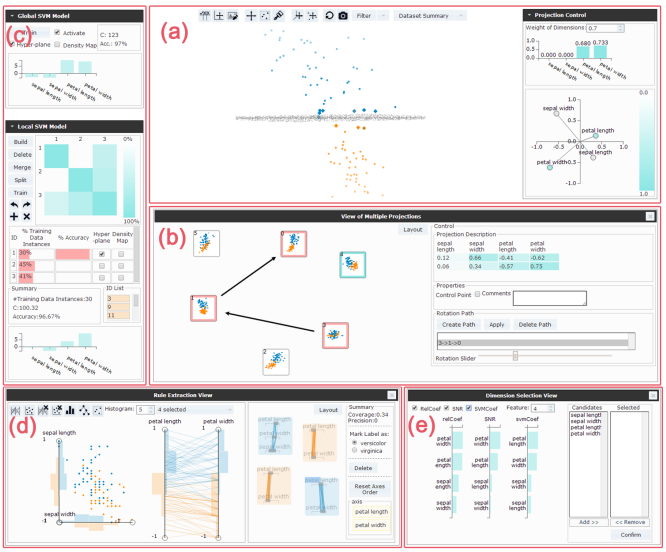

Fig. 2. Interface of EasySVM: (a) the data view, (b) the view of multiple projections; (c) the SVM model building view, (d) the rule extraction view, and (e) the dimension selection view.

interactive visualization and visual exploration methods to facilitate understanding of the model building process. Meanwhile, additional data exploration and manipulation can be performed at any model building step. Figure 3 (a) shows our iterative visual model building process:

Data Exploration and Initial Training The user can explore the training dataset to approximate the classification boundaries. Then a global model is trained with all training data instances and an initial parameter C in the global SVM model building panel (Figure 2(c)). After this initial step, the user can perform analysis and operations iteratively in the following two steps.

Visual Model Exploration and Validation For a trained model, an initial projection is generated at the direction of the side view. The user can evaluate the model starting from the side view to locate misclassified instances, check their distributions and patterns with projections, view the boundaries near the separating hyper-plane, and make decisions on data operations. Compared with the traditional machine learning procedure, the visual exploration of the training result is devoted to providing insight of the reason why the training result is. Meanwhile, the prediction accuracy on its own training dataset is computed and displayed as another reference for model validation.

Data Manipulation and Parameter Tuning After exploration, some training data instances that affect the model building can be modified by changing labels or deleted from the dataset if they are considered to be noise or instances with invalid values. In addition, the parameter C can be tuned in this step to balance the trade-off between margins of the hyper-plane and prediction accuracy on its training dataset. It is required to re-train the model after these two operations to update the model and classification results. The model building process stops when the validation result reaches the user's requirement, such as prediction accuracy on a test dataset. It should be noted that for a dataset with non-linear or even more complex decision boundaries, local linear models are needed.

The data view (Figure 2(a)) in our system is based on a scatterplot in which data instances are projected into a low-dimensional subspace. We use an orthogonal projection to embed high-dimensional data instances and the SVM model in a 2-D plane. Given an orthogonal projection matrix $\boldsymbol{A}_{m\times 2} = [\boldsymbol{f}_1, \boldsymbol{f}_2]$, two m-d vectors $\boldsymbol{f}_1, \boldsymbol{f}_2$ span a 2-D plane in the m-dimensional data space where all data instances involved are projected. Applying it to a high-dimensional data instance yields a corresponding data point on the 2-D plane, i.e. the coordinates of data point in the scatterplot view $\boldsymbol{x}'_i = \boldsymbol{x}_i\boldsymbol{A}$. It should be noted that the separating hyper-plane is usually cannot find the formula of its 2-D projection. We first sample a set of points on the separating hyper-plane, and then project sample points on 2-D plane to approximate the hyper-plane. Visual channels including filled color, border color and shape are employed to present the input label, predicted label and whether it is a support vector of a data instance. For the separating hyper-plane of the SVM model, the sample points are drawn on the view as dots in grey color with smaller size than data points. Additionally, to visualize the density of training data instances, a density map is computed and rendered.

To perform visual exploration, an interactive projection control method is provided for manipulating the direction of the selected projection. Our method is based on [1], where the control is similar to the "trackball control" in a high-dimensional data space. A weight is specified first for each dimension to determine which one is going to be manipulated, then the user can rotate the projection plane by dragging the mouse in the scatterplot view. Finally the user's mouse action is translated into a rotation matrix and applied to the projection matrix, thus changing the scatter-plot. To assist the user in the exploration of the relationships among multiple projections, we offer a view of multiple projections (Figure 2(b)). Each projection glyph holds a snapshot of the interesting scatterplot inside the glyph with the projection matrix. We define the similarity between two projection glyphs as the Euclidean distance between two corresponding projection matrices, i.e. $\|\boldsymbol{A}_1 - \boldsymbol{A}_2\|_2$. Thus, the layout of the glyphs is determined via a local affine multidimensional projection algorithm. Among multiple projection glyphs, the user can plan a rotation path containing a sequence of projection glyphs, then a rotation animation is generated based on interpolation between adjacent projections along the path with a slider to control the position of animation.

The orthogonal projection is unable to show high-dimensional data separation which may cause profound visual clutter. A dimension selection view (Figure 2(e)) is proposed for filtering out non-informative dimensions in the classification task. In this view, three bar charts rank all the dimensions of the training data instances according to three ranking measures: correlation coefficients between the dimension and the class labels; signal-to-noise ratio; weighting by a linear SVM. The user can select the most informative dimensions, which will be highlighted in the candidate list. After the selection is confirmed, the projected scatterplot will be updated to apply the changes. The dimensions that are filtered out will not take part in the projection process and the future model building process.

2.2 Visual Local SVM Building

For clarity, we use the term "global model" to represent the SVM model built with the process described in section 2.1, which covers all the training data instances in the dateset. A "local model", on the contrary, is trained on a selected subset of training data instances.

Before building local SVM models, it is necessary to perform a preliminary exploration of the decision boundary and evaluation of complexity. First, it is necessary to explore the data distributions, because the data distribution near the decision boundaries is a strong indication of the boundary complexity. The user can control the projection interactively and inspect view the patterns of boundaries between two classes of data points. Additionally, the user can explore the decision boundaries guided by the global SVM model. Although not applicable with low prediction accuracy under the complex circumstance, the separating hyper-plane of the global SVM model can act as a guide to the decision boundary. The training data instances separated into the opposite side of the hyper-plane always imply local regions containing non-linear boundaries, or even non-classifiable area with mixed distributions of two classes. The user can locate the regions in the projected scatter-plot, check the patterns visually and make further analysis.

Our visual local model building process extends the previous global one in section 2.1. The key issue is to 1) locate regions-of-interest, and 2) select proper subsets of training data instances for each a local SVM

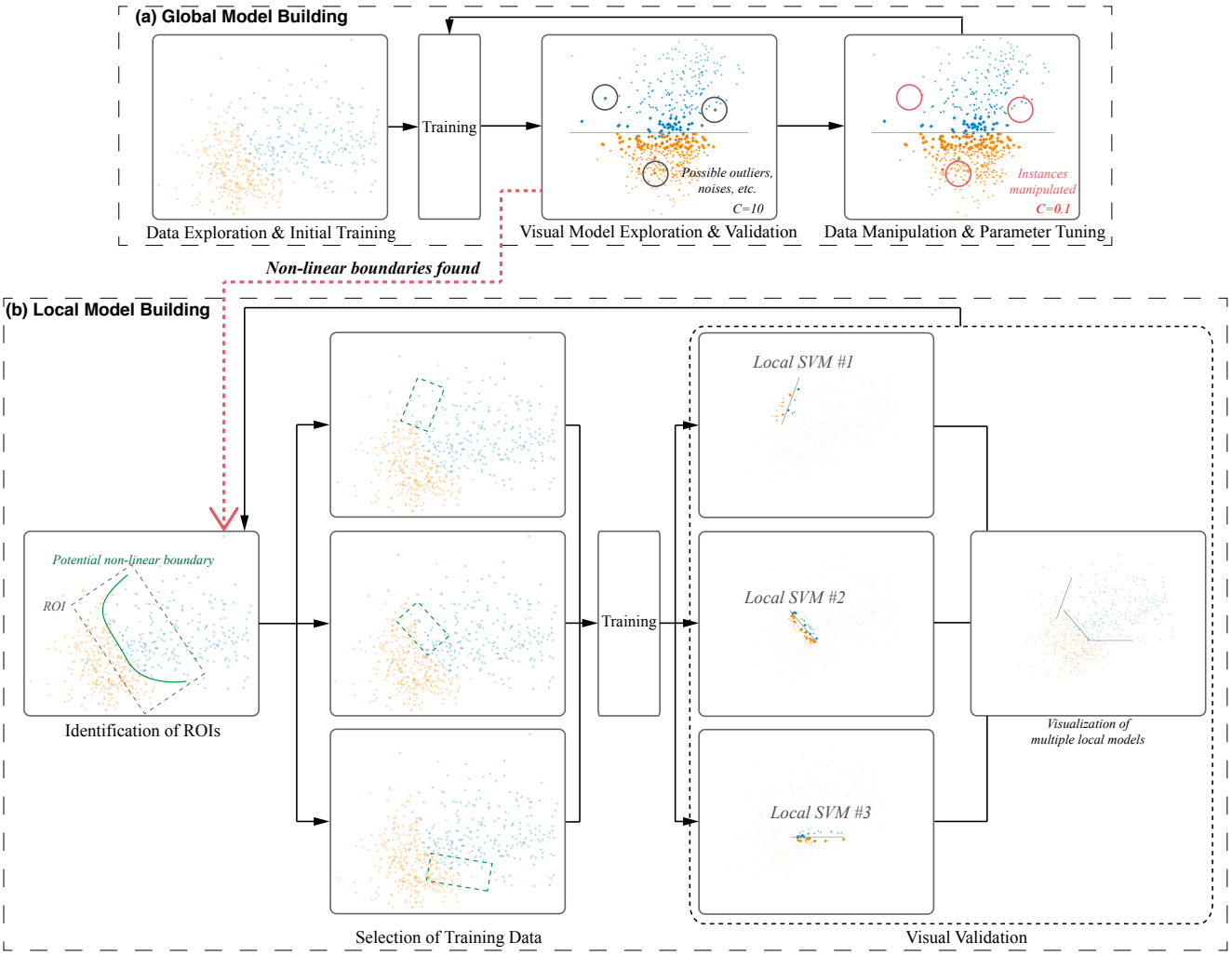

Fig. 3. (a) Global SVM model building process. If a non-linear decision boundary is found, the user can enter the local model building process illustrated in (b).

model. We propose the following four steps to build local models iteratively (Figure 3).

Identification of regions-of-interest The target regions are located via the visual exploration methods illustrated in section 2.2. It should be pointed out that, when some local models have been created, each of them but not only the global one can be considered as a starting point for visual exploration and locating. Local models with different numbers of training data instances and ranges of coverage in high-dimensional data spaces will provide diverse levels of details. The user can select the starting point on demand.

Selection of training data instances The training data instances of a new local model can be selected directly via user interactions on the projection view. Moreover, we propose a hierarchical model creation method based on split-and-merge operations on the created models. A local model can be split into two new ones by dividing its corresponding training dataset into two subsets and training two new models on each subset. The training data instances from several models can also be merged together. A new local model is trained on the merged dataset to replace the existing ones. Both operations change the level of details from different directions. When a model is split into several multiple ones, the details of the decision boundary can be "carved out", while in merging operation a set of models carrying much detailed information is replaced by a generalized model. The level-of-detail exploration and incremental model creation allow the user to depict the decision boundaries and understand the distributions.

Training Once the parameter C is set for each model, the newly created or updated local models will be re-trained in this step.

Validation In this step, the training result is to be validated from two aspects: For a single local model, the same validation methods for the global model is to be applied; for checking relations and coverage between multiple models, the projection rotations between multiple models can be considered as indications of their positions and directions.

After the local model building process is done, they can be employed for predicting new data instances. A prediction algorithm is provided based on the set of local models, where the query instances are labeled by the nearest local SVM. Algorithm 1 illustrates the prediction process.

2.2.1 Visualization and Interactions of Multiple Models

The statistical information of existing local SVM models are displayed. In particular, a matrix is used to encode the similarity among all models in terms of the number of shared training data instances. Each matrix cell (i, j) is defined as:

$$similarity(i,j) = \frac{\#(TrSet(H_i) \cap TrSet(H_j))}{\#(TrSet(H_i))}$$

where $TrSet(H_i)$ is the training dataset of the local model whose ID number is equal to i. The matrix is visualized with an

Algorithm 1 Prediction procedure of local SVMs

Require:
The decision functions of n local SVMs, $H_i(\boldsymbol{x}), i = 1,2,\ldots,n$;
The training dataset of n local SVMs, $\boldsymbol{X}_i, i = 1,2,\ldots,n$;
The query instance, $\hat{\boldsymbol{x}}$;
Ensure:
Label of $\hat{\boldsymbol{x}}$, $\hat{y}_i$;
1: $\boldsymbol{X}_{knn} = k$ nearest neighbors of $\hat{\boldsymbol{x}}$ in $\bigcup_{i=1}^{n} \boldsymbol{X}_i$
2: $i_{nearest} = \arg\max_i |\boldsymbol{X}_{knn} \cap \boldsymbol{X}_i|$
3: $\hat{y}_i = H_i(\boldsymbol{x})$

opacity-modulated color (green in our system), as shown in Figure 2(c). The opacity of each cell is set to be its similarity.

2.3 Visual Rule Extraction

We denote a rule as a small-sized canonical subspace of the input high-dimensional data space that may encapsulate some domain knowledge. The subspace is bounded by a set of dimension intervals, each of which refers to one or several ranges of a dimension. Thus, determining a rule is identical to specifying one or several intervals in each dimension. Each rule denotes a class and assigns the corresponding class label automatically to the data instances in the rule.

As shown in the rule extraction view (Figure 2(d)), we apply the flexible linked axes method to visualize the training data instance. The position and orientation of the dimension axes can be arranged freely. Between pairs of axes, the data instances are represented by parallel coordinate plot-styled edges or scatterplot-styled dots. The user can directly brushes the axis to select a range or select a region in the projected scatterplot. After the set of selected ranges is committed, a new rule glyph that represents the set of ranges, i.e. the rule, is displayed. In the rule glyph, a parallel coordinates plot is designed to summarize the dimension intervals. The background color of the rule glyph encodes its expected class with a unique color.

data instances included in a range should be maximized, because more training data instances indicate the higher confidence of the rule while classifying new instances.

3 Case Study

For the Wall-follow Robot Navigation dataset [2], a classifier is supposed to be trained on a series of 24-dimensional sensor values to predict the best action. We only use the data instances in "Move-Forward" and "Sharp-Right-Turn" in our binary classification (4302 instances in total) and divide the dataset into two parts: 50 percent as the training set, and the other 50 percent as the test set.

Data exploration (Figure 4 (a)) By default, the initial projected scatter-plot is the same as 2-D scatter-plot with only the first two dimensions (US0, US1). The user starts from this initial projection and performs interactive projection control by selecting each of the other dimensions (US2∼US23). While manipulating dimension US14, a rough gap appears on a large branch on the right side, which indicates a potential linear boundary. However, training data instances of different classes in a smaller branch on the left side are overlapping and seem impossible to be linearly separated. A snapshot is taken to support further investigation.

Global SVM model building (Figure 4 (b)) After the preliminary data exploration, the user trains a global SVM model with all training data instances. However, the accuracy on the training dataset is around 80% with different settings of parameter C, meaning that the dataset is not linearly separable. In the side view, a set of wrongly-classified instances are observed to be scattered near the separating hyper-plane.

Local SVM model building (Figure 4 (b)) The user manipulates the dimension US14 again to investigate the probable boundary found before, while the separating hyper-plane is located in a different direction. Now the user decides to build two separated models for the two branches. After training two local SVM models, two side views show that the two corresponding separating hyper-planes are in

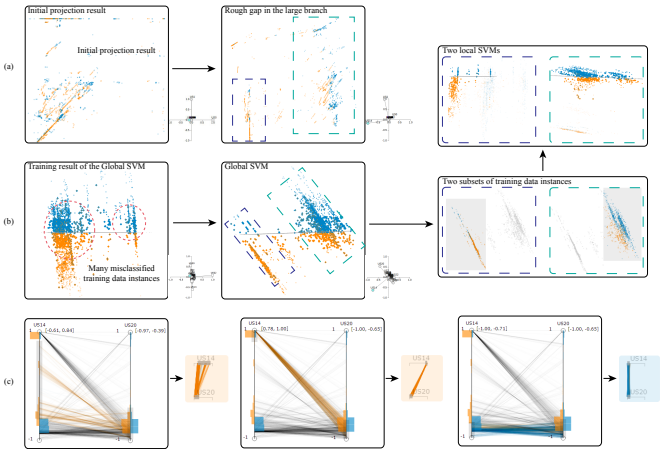

Fig. 4. Analyzing the Wall-follow Robot Navigation dataset. (a) After adding the weight of projection of dimension US14, the dataset is approximately separated into two branches in the blue and green box. A rough gap appears in the large branch on the right side. (b) The result of Global SVM model is not applicable because too many data instances are misclassified (marked in the two red circles). When increasing the projection weight on dimension US14, the projection result shows that the separating hyper-plane of the global SVM model is located in a different direction to the gap found in the previous step. Two separate local models are created based on the two branches. (c) Three classification rules are extracted based on the result of the local model built on the large branch.

the different directions and give better separation in the regions near their training datasets, which is also indicated by the two accuracy values (around 91.3% for the model on the smaller branch and 94.0% for the one on the larger branch). Animated rotation among the side views of the global model and the two local models partially depicts the relations between three separating hyper-planes. Thus, the global SVM model is replaced by the two local linear ones.

Rule extraction (Figure 4 (c)) Rule extraction operations are assisted by the two local models. The user chooses to extract rules for the local model on the large branch. From the weight vector of the local model, it is obvious that the dimension US14 and US20 dominate the direction of the separating hyper-plane. Thus the user constructs a parallel coordinates plot between US14 and US20. The user brushes three combinations of ranges on the two axes and generates three rules.

Prediction Accuracy The global linear SVM receives $81\% \pm 1.0$ on the test set, while the local SVM models receive $88\% \pm 3.0$.

4 Conclusion

In this paper, we propose a novel open-box visual analysis approach for building SVM models. The user can perform visual exploration of the dataset and the relations between data instances and SVM models. Meanwhile, a visual-enhanced local linear model building approach is dedicated to expanding the traditional linear SVM to deal with non-linear decision boundaries. Finally we provide a visual rule extraction method to enable the user to retrieve classification rules from the model building results.

References

[1] D. Cook and A. Buja. Manual controls for high-dimensional data projections. *Journal of Computational and Graphical Statistics*, 6(4):464–480, 1997.

[2] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *Robotics Symposium (LARS), 2009 6th Latin American*, pages 1–6. IEEE, 2009.

[3] L. Ladicky and P. Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 985–992, 2011.