Multilevel Support Vector Machines

Kyle Seelman

April 2019

1 Abstract

A support vector machine (SVM) is a discriminative classifier defined by an optimal separating hyperplane. The support vector machine is desirable for its ability to find an optimal separating hyperplane even for high-dimensional nonlinear data. This paper will cover the basics of support vector machines and introduce the idea of a multilevel support vector machine, a method for classification that greatly reduces the computational time of traditional SVMs, without sacrificing accuracy.

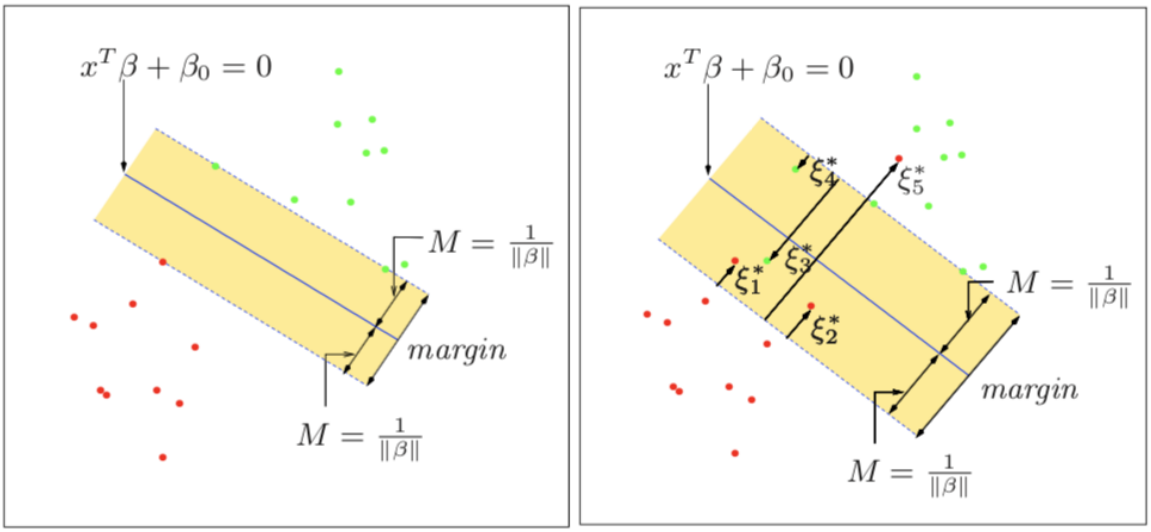
2 Introduction to Support Vector Machines

Support vector machines are one of the most well-known supervised methods for classification. Given data consisting of N pairs $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, with $x_i \in \Re^p$ and $y_i \in (-1, 1)$, a common question is how to separate the data for classification of new data. A SVM finds an optimal hyperplane defined by

$$x : f(x) = x^T\beta + \beta_0 = 0$$

which gives the classifier as

$$G(x) = sign[x^T\beta + \beta_0]$$



(a) Separable case

(b) Overlapping feature space

Figure 1: *On the left, the solid line is the decision boundary with margin 2M. With the closest data point being of length M from the decision boundary line. On the right, data from the two classes overlap and thus need slack variables to compensate*

The optimal hyperplane is the hyperplane that induces the largest margin between the training points, which gives the convex optimization problem

$$\min_{\beta,\beta_0} ||\beta||$$

$$\text{subject to } y_i(x^T\beta + \beta_0) \geq 1, i = 1, ..., N$$

Much detail can be given to show exactly how to solve this optimization problem, but this section is just an introduction to the information needed to understand the multilevel algorithm detailed in later sections.

The multilevel algorithm discussed in this paper was developed for the classification of nonlinear data, where it becomes a time consuming task to use traditional SVMs. Since a support vector machine is a linear solver, we can project the input space into a higher dimension to achieve better class separation, which translates to classifying a nonlinear problem in the base dimension. Polynomials and radial are common basis expansions used to project the nonlinear data onto higher dimensions.

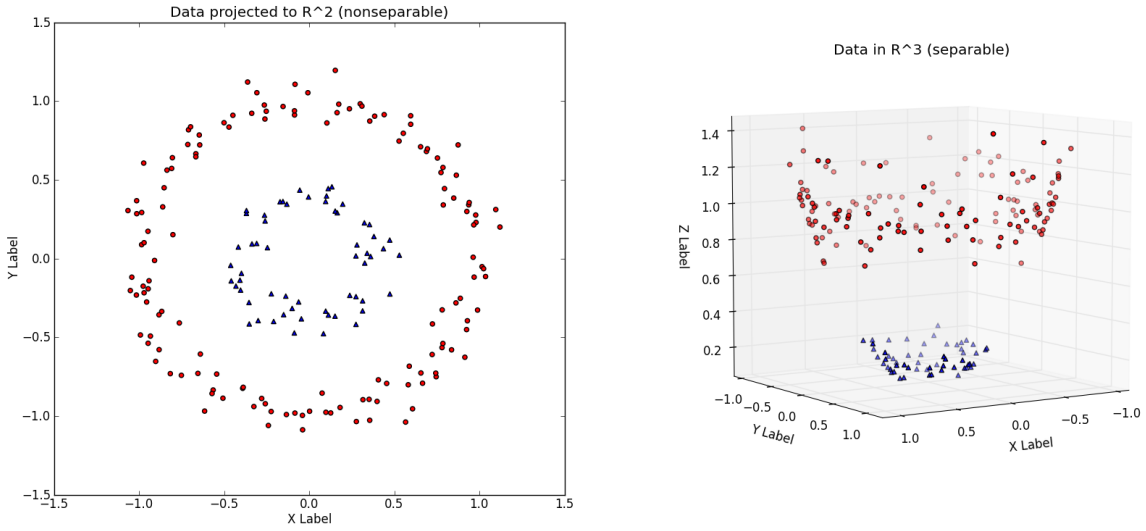


Figure 2: *The left panel shows data that is not easily separable by a linear classifier, but the right panel shows how the projection of the data into higher dimensions can lead to an obvious separation*

Once the basis functions have been chosen $h_m(x), m = 1, 2..., M$, we fit our SVM classifier using the new input space $h(x_i) = (h_1(x_i), h_2(x_i), ..., h_M(x_i)), i = 1, 2, ...N$. This produces the function $\hat{f}(x) = h(x)^T\hat{\beta}+\hat{\beta}_0$ and the classifier now becomes $\hat{G}(x) = sign(\hat{f}(x))$. On top of the data possibly being nonlinear, some data points from each class might overlap. This leads to the idea of slack variables, ξ_i, which allows for maximizing the margin while taking into account the non-separable points. The slack variables, penalize the minimization problem shown above for misclassified data points. This addition of a penalty is called soft-margin SVM and is written as

$$\min ||\beta|| \text{ subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i$$

$$\xi_i \geq 0, \ \sum \xi_i \leq K$$

where K is a bounding constant. This is equivalent to the following, which is the common way of representing an SVM classifer for non-separable data,

$$\min_{\beta,\beta^0} \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{subject to } \xi_i \geq 0, y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i \forall i$$

where C is the cost parameter.

Another difficulty than can occur when using real data is imbalanced classes, which tend to cause the SVM to misclassify data from the minority class. The most common approach to these type of classification problems, is the use of cost-sensitive learning (CSL), where the misclassification of the minority class data is heavily penalized. This gives rise to the weighted support vector machine (WSVM), a SVM that uses cost-sensitive learning to account for an imbalanced dataset.

3 Multilevel Support Vector Machines

This section and following ones will explore the idea of a multilevel framework that efficiently solves these large, high-dimensional data sets. Typically, for finding the optimal hyperplane the whole training set is solved at one optimization step, but this multilevel approach gradually coarsens the data and finds the support vectors at each level of coarseness. This method significantly reduces computation time, without losing accuracy of the classification. This multilevel framework can be applied not only for reducing the computational time of SVM, but for WSVM as well, when an imbalance in the dataset is present.

The main idea behind this multilevel algorithm is a continuous coarsening of the data, a method for approximating the original data, but with fewer degrees of freedom. This is achieved by continuously projecting the data into a lower dimension until the desired level of coarseness is reached. Then, the problem is solved locally at each level of coarseness and the solutions are combined to form the global solution, as the multilevel algorithm uncoarsens the data and projects back into higher dimensions. In

the case of a imbalanced data set, this method creates a balanced coarse representation of the data, which accurately approximates the original problem.

4 Detailed Multilevel Framework

Let Ω be a set of labeled data points, $\Omega = (x_i, y_i)_{i=1}^m$, where each x_i has n features and a corresponding class label $y_i \in (-1, 1)$. Then, the set Ω is made up of two classes, the majority and minority class, denoted C^+ and C^- respectively. Using a slight change of variable and our expression from the previous section, our convex optimization problem becomes,

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i$$
$$\text{subject to } y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i \quad i = 1, ..., m$$
$$\xi_i \quad i = 1, ..., m \geq 0$$

where ϕ is our kernel function that maps the data into a higher dimension. The magnitude of our penalized slack variables is controlled by the parameter C. Sequential Minimal Optimization (SMO) is used to solve the Lagragian dual problem.

As mentioned in the previous section, the use of CSL is used to compensate for imbalanced data sets. So, for the implementation of a WSVM, different weights are assigned to each class and yields,

$$\min \frac{1}{2}||w||^2 + C^+\sum_{i|y_i=+1}^{m_+}\xi_i + C^-\sum_{j|y_j=-1}^{m_-}\xi_i$$
$$\text{st. } y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i \quad i = 1, ..., m$$
$$\xi_i \quad i = 1, ..., m \geq 0$$

where C^+ and C^- are weights associated with the majority and minority classes.

For the kernel, $\phi(x)$, in both the SVM and WSVM problems, the common radial basis kernel given by

$$k(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma \geq 0$$

is used, where $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ allows the similarity between points x_i and x_j to be measured.

Additionally, to build an optimal classifier, paramter tuning must be done to select the optimal C and kernel function parameter, γ. To solve this parameter tuning problem, a common method used is grid search. But, in this multilevel algorithm, an adaptation of the nested uniform design (UD) model selection algorithm is used. This model has been shown to be more efficient and robust than other common methods. According to [3], to find the range for γ, let A_i^* and A_j^* be a pair of the closest distinct points in the training set and let $\rho = ||A_i^* - A_j^*||_2^2$. Confining the range of the kernel function value for this pair to

$$0.150 \leq e^{-\gamma ||A_i^* - A_j^*||_2^2} = e^{-\gamma \rho} \leq 0.999$$

then using $L_\gamma \leq \gamma \leq U_\gamma$, we can convert the above to

$$L_\gamma = -\frac{ln(0.999}{\rho} \approx \frac{10^{-3}}{\rho} \text{ and } U_\gamma = -\frac{ln(0.150}{\rho} \approx \frac{1.90}{\rho}$$

The lower bound for γ is a lower bound for underfitting and the upper bound is an upper bound for overfitting. By the inequality $e^{-\gamma\rho} \leq 0.999$, pairs with similarity measuer greater than 0.999 are lumped together by the kernel function, the converse goes for the lower bound of 0.150. Using the upper bound of γ, we can transform it into a lower bound for σ by $\sigma = \sqrt{\frac{1}{2\gamma}}$. Giving

$$\frac{\min_{A_i \neq A_j} ||A_i^* - A_j^*||_2}{\sqrt{2 \times 1.90}} \leq \sigma$$

That is, in terms of σ, the kernel size is at least half of the minimum nearest neighbor distance. Thus, the range $[L_\gamma, U\gamma]$ covers the range of underfitting to overfitting.

The multilevel framework has three main phases, the gradual coarsening of the data, solving the coarsest support vector solution, and the gradual refinement of the support vectors. The coarsening phase deals with the two classes separately, building separate coarsening hierarchies for each class, C^+ and C^-. Each coarser

level contains a subset of points from the previous finer level, which are determined by using approximated k-nearest neighbor graphs. Then, after the coarsest level is solved, the support vectors are gradually refined level by level.

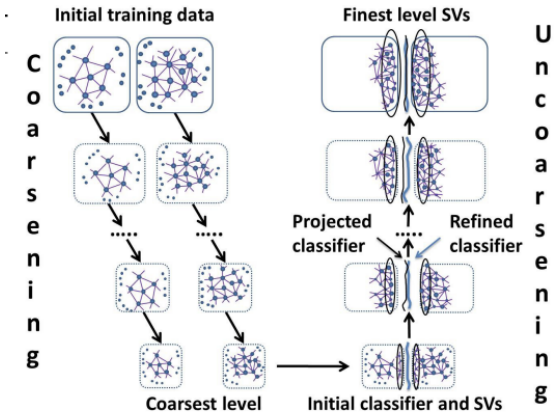


Figure 3: *Visualization of the multilevel framework*

5 Coarsening Phase

As mentioned previously, the classes are gradually coarsened separately, but the same proposed algorithm is used for each class. The algorithm provided in [1] is shown below

Algorithm 1: The Coarsening

Input: $G = (V, E)$ for class C ;

$\hat{V} \leftarrow$ select maximal independent set in G;

$\hat{U} \leftarrow V \setminus \hat{V}$;

while $|\hat{V}| < Q|V|$ **do**

  while $\hat{U} \neq \emptyset$ **do**

    randomly pick $i \in \hat{U}$;

    $\hat{U} \leftarrow \hat{U} \setminus i$;

    $\hat{U} \leftarrow \hat{U} \setminus N(i, U)$;

    $\hat{V} \leftarrow \hat{V} \cup i$;

  end

  $\hat{U} \leftarrow V \setminus \hat{V}$;

end

return $\hat{V}$

Given a class C, the algorithm starts with the construction of an approximated k-nearest neighbors graph, where $V = C$ and E are the edges of the graph. Next is to select a set of points $\hat{V}$ for the next-coarser problem where $|\hat{V}| < Q|V|$. Q in this case is the parameter for the size of the next-coarsest level. Too aggressive coarsening can lead to loss of information. From [1] it was found that $Q = 0.6$ gave good results. A larger Q can be used for even more gradual coarsening, but this can significantly slow the algorithm down.

In (line 2) a random independent set is chosen using greedy approximated k-nearest neighbor, it is then eliminated from the graph and then the next independent set is chosen and added to $\hat{V}$ (lines 5-10).

The multilevel framework continuously calls this coarsening algorithm until a hierarchy of l coarse sets, $\{\Omega\}_{i=0}^{l}$ are created and the corresponding k-nearest neighbors graphs, $\{G_i = (V_i, E_i)\}_{i=1}^{l}$ are saved for use at the refinement phase. Let the data

and labels at each level be denoted $(X_i, Y_i) \in \Re^{k \times (n+1)}$. Once we reach the coarsest level, we can solve exactly for the coarsest classifier and support vectors.

6 Refinement Phase

From the previous phase, the set of support vectors S_{i+1} and parameters C_{i+1} and γ_{i+1} are given. The goal of this phase is to refine and optimize the support vectors for the current level i. This is done by inheriting the solutions from the coarser level $i + 1$ and using the algorithm

Algorithm 2: The Refinement

Input: $\Omega_i, S_{i+1}, C_{i+1}, \gamma_{i+1}$;

if *i is the coarsest level* **then**

  Calculate the best (C_i, γ_i) using UD ;

  $S_i \leftarrow$ Apply SVM on X_i

end

Calculate nearest neighbors N_i for support vectors S_{i+1} from existing k-nearest neighbors graph;

$data^{(i)}_{train} \leftarrow S_{(i+1)} \cup N$;

if $|data^{(i)}_{train}| < Q_{dt}$ **then**

  $C^O \leftarrow C_{i+1}$;

  $\gamma^O \leftarrow \gamma_{i+1}$;

  Run UD using the intial center (C^O, γ^O)

else

  $C_i \leftarrow C_{i+1}$;

  $\gamma_i \leftarrow \gamma_{i+1}$;

end

if $|data^{(i)}_{train}| \geq Q_{dt}$ **then**

  Cluster $data^{(i)}_{train}$ into K clusters;

  $\forall k \in K$ find P nearest opposite class-clusters;

  $S_i \leftarrow$ Apply SVM on pairs of nearest clusters only

else

  $S_i \leftarrow$ Apply SVM directly on $data^{(i)}_{train}$

end

return S_i, C_i, γ_i

As mentioned above, the coarsest level is solved exactly using UD model selection (lines 2-5). If level i is one of the intermediate level, the set of training data $data^{(i)}_{train}$

is built by inheriting the support vectors from the coarser level and adding to them the nearest neighbors from the current level (lines 6-7). If the size of $data^{(i)}_{train}$ is still small, then the parameters from the coarser level are used to retrain the data (lines 9-10, 19). In contrast, if the data is large enough, then $data^{(i)}_{train}$ is clustered and pairs of nearest opposite clusters are retrained and their solutions added to S_i. This algorithm is run recursively until the framework has reached the finest level (base level of original problem).

7 Evalutaion Metrics

This multilevel framework was tested on common binary classification benchmarks. Since the data was binary, the performance of the multilevel framework was evaluated by using a classification confusion matrix, which measures the true positives, false positives, false negatives, and true negatives. Then the accuracy of the model was determined by number of correctly classified test points over the whole set. But, for imbalanced problems, accuracy might not be a well-suited method for evaluating performance. Thus, as in [1], sensitivity (SN) and specificity (SP) defined as,

$$SN = \frac{TP}{TP + FN}, SP = \frac{TN}{TN + FP}$$

are used. As a final evaluation metric, the geometric mean defined as

$$\text{G-mean} = \sqrt{SP * SN}$$

is used.

	$\mathbf{C}^+$	$\mathbf{C}^-$
$\mathbf{C}^+$	TP	FP
$\mathbf{C}^-$	FN	TN

Figure 4: *Visualization of the multilevel framework*

The code was implemented in R and is available at **https://kseelma.people.clemson.edu/**.

8 Results

All the data was normalized prior to the application of the multilevel framework. In the case of imbalanced data, weights relative to the proportion of the inverse of the class size, $\frac{C}{2|C^+|}$ and $\frac{C}{2|C^-|}$ were used. If the class sizes were equal, then the weights would clearly be equal. The data used is displayed in Table 1, where the r_imb is the imbalance ratio defined by $r_{imb} = \frac{|C^+|}{|\Omega|}$. The performance of both SVM and multilevel SVM as well as WSVM and multilevel WSVM are shown in Table 2,3,4, and 5, respectively.

Dataset	r_{imb}	n_f	$\|\Omega\|$	$\|C^-\|$	$\|C^+\|$
Ringnorm	0.50	20	7400	3664	3736
Twonorm	0.50	20	7400	3703	3697
Hypothyroid	0.94	21	3919	240	3679

Table 1: Benchmark data sets used

Dataset	ACC	SN	SP	G-mean
Ringnorm	0.97	0.98	0.97	0.98
Twonorm	0.97	0.96	0.98	0.97
Hypothyroid	0.96	0.97	0.65	0.79

Table 2: SVM Performance

Dataset	ACC	SN	SP	G-mean
Ringnorm	0.97	0.98	0.96	0.97
Twonorm	0.96	0.96	0.97	0.96
Hypothyroid	0.96	0.96	0.66	0.80

Table 3: Multilevel SVM Performance

Dataset	ACC	SN	SP	G-mean
Ringnorm	0.97	0.98	0.96	0.97
Twonorm	0.96	0.98	0.98	0.98
Hypothyroid	0.97	0.98	0.71	0.84

Table 4: WSVM Performance

Dataset	ACC	SN	SP	G-mean
Ringnorm	0.98	0.98	0.96	0.97
Twonorm	0.97	0.98	0.97	0.97
Hypothyroid	0.98	0.96	0.70	0.82

Table 5: Multilevel WSVM Performance

Dataset	ML w/ MS	ML w/o MS	MS	No MS
Ringnorm	28	82	140	57
Twonorm	24	77	117	54
Hypothyroid	19	21	37	5

Table 6: Computation time (in sec.). ML is multilevel and MS is model selection

9 Conclusion

Clearly from the tables above, it can be seen that the performance of the multilevel (W)SVM is almost exactly as accurate as tradition (W)SVM. This is good since the computation time that is saved by using the multilevel shown in Table 6, will not come at a cost of accuracy. The multilevel framework discussed in this paper accurately classifies non-linear and imbalanced data sets at a fraction of the cost of tradition (W)SVM. This makes the multilevel framework very useful for real world data applications, where the data is commonly non-linear and imbalanced.

10 References

[1] Talayeh Razzaghi, Ilya Safro, "Scalable Multilevel Support Vector Machines", International Conference on Computational Science (ICCS) 2015, Procedia Computer Science, Volume 51, 2015, pp. 2683-2687

[2] Hastie, Trevor. Tibshirani, Robert. Friedman, Jerome. The Elements of Statistical Learning. New York: Springer, 2017. Print

[3] C.M. Huang, Y.J. Lee, D.K.J. Lin, and S.Y. Huang. Model selection for support vector machines via uniform design. Computational Statistics Data Analysis, 52(1):335–346, 2007