Multiclass Extensions of Support Vector Machines

Additional material for the book
Statistical Analysis Techniques in Particle Physics:
Fits, Density Estimation and Supervised Learning
(Wiley-VCH, 2013)

by Ilya Narsky and Frank C. Porter

December 24, 2013

Over the last two decades, various multiclass extensions of the binary SVM formalism have been put forward. It was hoped that these extensions would lead to more accurate classification than simple-minded reduction strategies such as "one versus one" (OVO) and "one versus all" (OVA). This hope remains unfulfilled, at least for data with a few dozen or fewer classes.

Here, we give a more elaborate description of these multiclass extensions, briefly discussed in Section 13.5.4. We refrain from presenting the full mathematical formalism of these techniques and merely outline their main ideas. As in the book, we do not recommend using these extensions in practice. Instead, we encourage the reader to use the standard schemes summarized in Chapter 16. If, despite our recommendation, you wish to invest time in learning SVM-specific multiclass extensions, you may find this material useful.

Let us introduce some notation. For data with $K > 2$ classes, replace the scalar soft score $f(\boldsymbol{x})$ with a score vector $\boldsymbol{f} = \{f_k\}_{k=1}^{K}$. For simplicity, let us focus on the linear problem. The k-th element of this vector,

$$f_k(\boldsymbol{x}) = \boldsymbol{\beta}_k^{\mathsf{T}} \boldsymbol{x} + \beta_{0k}, \tag{1}$$

defines a decision boundary for class k. Concatenating column-vectors $\boldsymbol{\beta}_k$ into a matrix $\mathbf{B}$ of size $D \times K$ for D variables and K classes, write an equivalent matrix representation,

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{B}^{\mathsf{T}} \boldsymbol{x} + \boldsymbol{\beta}_0, \tag{2}$$

where $\boldsymbol{\beta}_0$ is a vector of bias terms with K elements. As usually, $\boldsymbol{x}$ is a column-vector of length D, and the training matrix $\mathbf{X}$ is formed by transposing $\boldsymbol{x}$ for every observation into a row-vector and concatenating such row-vectors horizontally. The label is predicted into the class k with the maximal score f_k.

[Vap98] and [WW99] use multiclass hinge loss

$$\ell(y, \boldsymbol{f}) = \sum_{k \neq y} \left[2 + f_k - f_y\right]_+ . \tag{3}$$

Above, y is the true class label and the summation is taken over the predicted scores for all classes but the true one. The solution is not unique since adding a constant to each f_k does not change the loss value. To ensure uniqueness, [LLW04] use a modified version of the loss,

$$\ell(y, \boldsymbol{f}) = \sum_{k \neq y} \left[f_k + \frac{1}{K-1} \right]_+ , \tag{4}$$

requiring $\sum_{k=1}^{K} f_k = 0$. In their approach, a support vector for class y is given by $\boldsymbol{f}$ with $f_y = 1$ and $f_{k \neq y} = -1/(K-1)$. Either formulation can be reduced to the binary problem $K = 2$ by putting $f_{\overline{y}} = -f_y$ for the false class $\overline{y}$. The loss is then zero for $f_y \geq 1$, and we get an equivalent of the binary expression (13.34).

[WW99] and [LLW04] search for the optimal separation by minimizing the trace tr $\left(\mathbf{B}^\mathsf{T} \mathbf{B} \right)$, or equivalently the square of the Frobenius norm $\|\mathbf{B}\|_F^2$. They use one slack variable per observation per class. The $K-1$ terms in the loss function for every observation translate into $K-1$ linear constraints reducing the number of independent slack variables to $N(K-1)$, where N is the size of the training set. The solution,

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{A}^\mathsf{T} \mathbf{X} \boldsymbol{x}, \tag{5}$$

is given by an $N \times K$ matrix of kernel expansion coefficients $\mathbf{A}$ composed of elements α_{nk} with $\alpha_{ny_n} = 0$. The coefficients $\mathbf{A}$ can be found by solving a quadratic problem similar to the one in Eq. (13.38). A naive implementation of a quadratic programming solver would require $O(N^2K^2)$ memory storage. A decomposition technique [HL02] reduces the memory requirement and thus allows training these multiclass extensions on large datasets.

[CS01] set the multiclass loss to

$$\ell(y, \boldsymbol{f}) = \sum_{k \neq y} \left[1 + f_k - f_y \right]_+ . \tag{6}$$

and, similar to the other authors, minimize tr $\left(\mathbf{B}^\mathsf{T} \mathbf{B} \right)$. The special case $K = 2$ is recovered by setting $f = f_{+1} - f_{-1}$ and minimizing $\boldsymbol{\beta}^\mathsf{T} \boldsymbol{\beta}$ for $\boldsymbol{\beta} = \boldsymbol{\beta}_{+1} - \boldsymbol{\beta}_{-1}$. They reduce the number of slack variables to N, one per observation. This slack variable corresponds to the constraint imposed on the false class with the largest score, $\max_{k \neq y} (1 + f_k - f_y) \leq 0$. Thus instead of separating the true class from all other classes, [CS01] focus on maximizing the distance between the true class and its closest competitor, also known as the classification margin. The solution $\mathbf{A}$ has $N(K-1)$ independent elements because N elements are removed by requiring $\mathbf{A}\mathbf{1}_{K \times 1} = \mathbf{0}_{N \times 1}$. Unlike in [WW99] and [LLW04], the primal problem with fewer slack variables gives rise to a dual Lagrangian with a simple structure. The latter can be optimized by a simple iterative algorithm. [CS01] optimize $\mathbf{A}$ one row at a time. With all rows but one fixed, each iteration step requires solving a quadratic problem with $K-1$ variables. Further computational improvements, especially useful when the number of classes is large, are proposed in [TJHA05] and [BBGW07].

[PCST00] train $K(K-1)/2$ binary SVM classifiers to separate every class from every other class. Predictions for new observations are then computed by a special voting scheme. The authors explain this voting scheme in terms of *directed acyclic graphs* (DAG), but it can be just as easily explained in terms of deque operations. Make a deque with K elements, one per class, numbered from 1 to K. When a new observation is received, classify it using the SVM model trained to separate class 1 and class K. Remove the losing class from the deque, that is, pop the front or back element. Then classify this observation using the SVM model trained to separate the remaining first and last elements in the deque. Continue until the deque is composed of one class. Classify the observation into this class. In this approach, the predicted label can depend on the class order in the deque. Empirical studies [PCST00] find little correlation between the classification error and class order. Because in physics analysis one typically wants some measure of classification confidence, the lack of classification score predictions in the DAGSVM voting scheme would be seen as a disadvantage.

[TH07] construct a decision tree based on classification margins between groups of classes. Put the entire training data with K classes in the root node of the tree. Find the optimal split to partition the classes in two groups, G_1 and G_2. Send observations for classes in G_1 to the left child and send observations for classes in G_2 to the right child. Continue splitting recursively until every leaf node contains observations of one class only.

[TH07] propose several optimality criteria for finding the best split. The greedy approach maximizes the SVM margin between groups G_1 and G_2 over all possible partitions. A simple-minded implementation of this approach could be slow for many classes; for instance, one would have to choose the best split among $2^{K-1} - 1$ binary classifiers in the root node. To speed up computation, [TH07] first train $K(K-1)/2$ binary classifiers using the OVO strategy and record the classification margin for each pair of classes. At the second step, they construct a *complete linkage* clustering tree using the recorded margin as a measure of distance. A complete linkage tree works bottom up: At the bottom level, the tree puts every class in a separate cluster, and at each consecutive level, the tree forms a new cluster by combining one of the existing clusters with its furthest neighbor. "Furthest" here implies maximal margin. At the last step, [TH07] start at the top of the constructed linkage tree and move down using a simple fast algorithm to find the optimal binary split for the linkage clusters at each level. The obtained decision tree is identical to the one grown in a top-down manner by inspecting all possible partitions.

The constructed decision tree can provide insights not available from other multiclass schemes. Similar to DAGSVM, the margin tree can predict labels but not scores. [TH07] consider linearly separable data only, but their analysis could be extended to include non-separable data and other kernel functions.

Every proposal mentioned in this section includes an empirical study, often demonstrating a **slight** improvement in classification accuracy over OVA and OVO. Dedicated comparative studies of various multiclass SVM extensions can be found in [HL02] and [SAT$^+$05]. None of these studies convincingly shows superiority of the advanced extensions over OVO and OVA. [RK04] argue that classification accuracy obtained by DAGSVM and the described single machine methods can be reproduced in the OVA approach by fine-tuning the base binary classifiers.

[HL02], [SAT$^+$05], [PCST00], and [WW99] compare the training time and number of support vectors for a broad variety of real-world datasets with up to 26 classes. There is no clear winner in sparsity[1]. The OVO strategy is consistently one the fastest options. The same holds for DAGSVM since its training stage is equivalent to that of OVO. The margin tree algorithm can incur extra time to construct a linkage tree, but it should be small compared with the time required to train $K(K-1)/2$ binary learners. According to [HL02], the algorithms [WW99] and [CS01] can train as fast as OVO for some datasets and be an order of magnitude slower than OVO for others. OVA, although consistently slower than OVO, is less prone to such extreme outcomes. Binary SVM classifiers for OVO and OVA in these studies are typically trained by the SMO algorithm described in Section 13.5.3.

References

[BBGW07] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

[CS01] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[HL02] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[LLW04] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–82, 2004.

[PCST00] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in neural information processing systems*, 12(3):547–553, 2000.

[RK04] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[SAT$^+$05] A. Statnikov, C. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631–643, 2005.

[TH07] R. Tibshirani and T. Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8:637–652, 2007.

[TJHA05] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[Vap98] V. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, 1998.

[1]An SVM model is sparse if it has few support vectors.

[WW99] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *European Symposium on Artificial Neural Networks*, volume 4, pages 219–224, 1999.