TENSOR VOTING BASED BINARY CLASSIFIER

Mandar S. Kulkarni, Shankar M. Venkatesan, M. Arunkumar

Philips Research India, ManyataTech Park, Nagavara, Bangalore 560045, India
{Mandar.Kulkarni, Shankar.Venkatesan, Arunkumar.M}@Philips.com

ABSTRACT

We propose two novel Tensor Voting (TV) based supervised binary classification algorithms for N-Dimensional (N-D) data points. (a) The first one finds an approximation to a separating hyper-surface that best separates the given two classes in N-D: this is done by finding a set of candidate decision-surface points (using the training data) and then modeling the decision surface by local planes using N-D TV; test points are then classified based on local plane equations. (b) The second algorithm defines a class similarity measure for a given test point t, which is the maximum over all inner products of the vector from t (to training point p) and the tangent at p (computed with TV): t is then assigned the class with the best similarity measure. Our approach is fast, local in nature and is equally valid for different kinds of decisions: we performed several experiments on real and synthetic data to validate our approach, and compared our approaches with standard classifiers such as kNN and Decision Trees.

1. INTRODUCTION

Classification of data points based on training data is an important and highly researched problem. A number of approaches exist for both labeled (supervised) and unlabeled (unsupervised) training data [16] [11]. Where the data has high dimensions like object recognition and data mining, the class decision surfaces are usually non-linear – here methods like Support Vector Machines (SVM) [5][6][8][9] come to our help. Even SVM with its theoretical and practical advantages (e.g. testing complexity is proportional only to the number of support vectors as opposed to the large set used in k Nearest Neighbors (kNN)) cannot handle decision surfaces which are very highly complex, and many modifications of SVM and localized methods like Local SVM (LSVM) [2] have been attempted for that. Here we introduce two novel localized binary classification approaches based on the theory of Tensor Voting (TV) [7][1][13][14][15] that can handle complex decision surfaces (with a simple scale parameter), and evaluate them experimentally.

In what follows, we propose two TV based binary classification approaches (TVBC 1 and TVBC 2), which can deal with both linear as well as non-linear decision boundaries. The first (TVBC 1) approach (Section 2) uses a greedy implementation of Euclidean bipartite (induced by the two classes) minimum cost matching to identify potential points on the decision boundary, prunes these points, and smoothes this boundary with local planes with a call to TV: then during the test

David C. Wyld (Eds) : ICCSEA, SPPR, CSIA, WimoA, SCAI - 2013
pp. 431–439, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3545

phase, these planes on the pruned boundary are used to decide class membership. Details of this approach are as follows: For each training data point in class 1, based on Euclidean distance, we find the closest training data point from class 2. These form a pair, and these two data points are excluded when this process is repeated until no data points are left to be considered. As explained later, the average vector of this pair (i.e. the midpoint of the line connecting these two data points) of closest data points is a candidate to belong to decision boundary. Using the mid points of pairs of closest data points from two classes, we form the decision boundary and model it by local planes using TV. Using the estimated local plane equations, we then assign test data points to either of the class. Time complexity of TV is of order O(N M log M) [1] where N is the dimensionality of data and M is the number of data points as opposed to traditional SVM which is of $O(M^3)$.

The second (TVBC 2) approach (Section 3) computes for any test data point an interesting class similarity measure that combines distance and orientation alignment with training data points and uses this measure for deciding membership: this approach yields a surprisingly accurate classifier (results outlined in later sections) showing the efficacy of TV as a binary classifier; the second approach is also conveniently useful for multiclass problems. We performed numerous experiments on synthetic as well as real data to validate our approach. Accuracy as well as time comparison results are given in the experimentation section.

2. TV BASED BINARY CLASSIFICATION – FIRST APPROACH (TVBC 1)

The major modules of our approach in TVBC1 are finding pair of closest data points from two classes, modeling decision boundary by local planes and classifying data points based on local plane equations. We now explain each of these modules in detail.

2.1 Finding Closest Pairs of Data Points

The best decision boundary is the one which maximally separates the two classes. Ideally, it should be mid-way between the two classes. As each class may have an unknown distribution, decision boundary should locally pass through centre of pair of closest data points from two classes. We aim to find such decision boundary using the training data. We use an average operation for closest inter-class data points which sets decision boundary such that classes are separated maximally in the local sense. Therefore, decision boundary adapts itself according to local variations within the classes. For a class 1 training data point, we find the nearest data point from class 2. We use Euclidian distance as the closeness measure. We then find an average vector of this pair which probably is a part of separating hyper plane/hyper surface. We then exclude this pair from further computation and in the similar way find the average of closest data points from the remaining training set recursively. Fig. 1(a) shows the example of two classes in 3D. The classes are Gaussian distributed with same co-variance and with different mean. Note that these classes are linearly separable. Blue color points indicate class 1 while red color points indicate class 2. Fig. 1(b) shows the average vectors of pairs of closest data points with black color. Note that the boundary formed due to mean vectors lies exactly at the center of two classes. As two classes have similar distribution, decision boundary is planar (linear) in nature as expected. Fig. 2(a) shows an interesting example of non-linearly separable classes. Two classes are part of spheres with different radius. Class 1 and class 2 points are indicated by blue and red color respectively. In Fig. 2(b) the probable decision surface points computed are shown with black color. Note that as we compute the averages of closest inter-class data points, decision boundary adjusts itself according to the local structure of the data.

The above exact nearest neighbor approach can be extended to k-nearest neighbors where k is inversely proportional to the density of the neighborhood, thus giving a more uniformly sampled space consisting of synthetic decision-surface points for TV to work on, but this will add a factor k to the expensive time complexity of $O(M^2)$ computations (in our current implementation this will not improve even with a careful use of data structures like k-d trees which degenerates into almost same complexity for more than a few dimensions). Our timing for this part perhaps will come down substantially if we settle for some reasonable choice of approximate nearest neighbor (ANN) search using approaches like Locality Sensitive Hashing [12], or perhaps invoke some optimized internal data structures in [13]. Also, if we select k closest data points (k is inversely proportional to neighborhood density), but keep the total increase in number of data points to a smaller constant factor, we can perhaps do better.

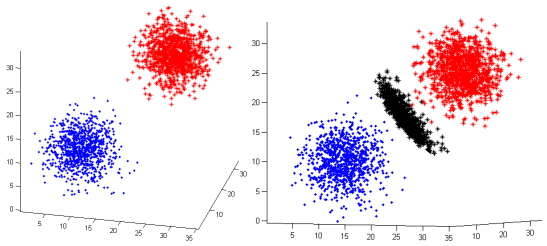
2.2 Decision Boundary Modeling

To build a classifier, we need to model the probable decision boundary points computed in the previous step. If the data is noisy in nature, which is the case for most of the real datasets, the decision boundary computed will be distorted by many outliers. It is important to remove these outliers prior to modeling. We use the geometric structure inference framework TV [7] for this purpose. The only variable parameter in the framework is scale of voting which defines the extent of neighborhood. Higher scale of voting corresponds to larger extent of neighborhood and thus more smoothness. A decision hyper plane in N dimension space has 1 normal and N-1 tangents. As described in [1], the saliency of N-1 dimensional structure in N-D can be computed as

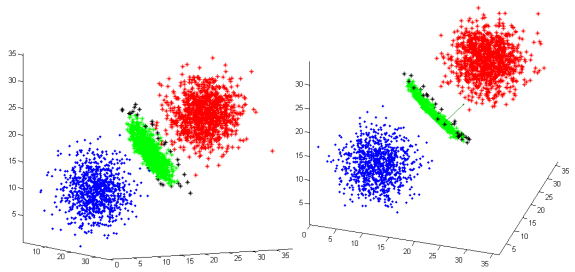
$$\text{Saliency} = \lambda_1 - \lambda_2 \qquad (1)$$

where λ_1 and λ_2 indicate top two eigenvalues of tensor matrix at a given decision boundary point. The more the saliency value, the more likely is the point being part of a decision hyper plane/ hyper surface. Outliers (if any) receive less or no support from the neighborhood and can be identified with their low saliency values. We remove points which have lower saliency value than the predefined threshold. The appropriate threshold for the saliency is chosen experimentally. Thus, the subsequent modeling of the decision boundary becomes robust to noise.

Fig. 1. (a) Linearly separable classes, (b) probable decision boundary points (in black), (c) TV based noise removal. Green points indicate refined points. (d) estimated plane normal (green arrow)



(a) (b)

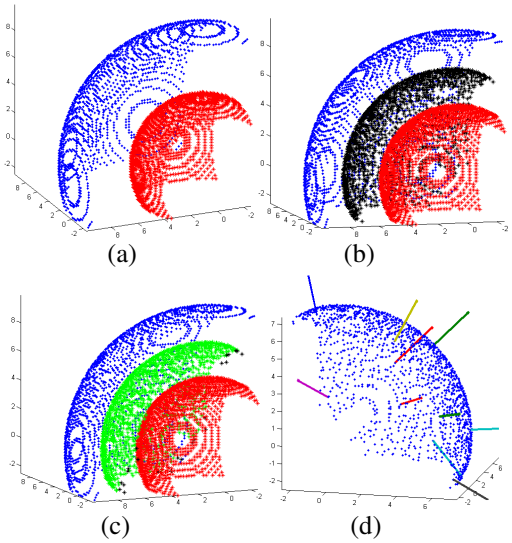


(c) (d)

Fig.1 (c) shows the output of noise removal on probable decision boundary points (shown in black in Fig. 1(b)). In Fig. 1(c), green points indicate the points which satisfy the saliency threshold criterion. Note that points which are little away from the decision boundary plane (still in black) are identified as outliers. Similarly, Fig. 2(c) shows the decision boundary points in green which satisfy the saliency criteria.

We then model the noise removed decision boundary points by linear local planes. For a point of the decision boundary, we first find its neighboring points. The extent of the neighborhood is kept same as the scale of voting. We run a pass of ND TV on these points. With eigen decomposition of the tensor matrix, we get eigenvectors and eigenvalues at each decision boundary point. The eigenvector with the highest eigenvalue represents the normal direction at a point. We take normal directions at all neighboring points and find the co-variance matrix. As all points of a same plane have a unique normal direction, the eigenvector of the co-variance matrix with highest eigenvalue is treated to be the normal direction to the plane. To make normal estimation robust, normal direction vector at each decision boundary point is weighted by saliency value (computed as in Eq. (1)) prior to calculation of co-variance matrix. As can be inferred, TV inherently takes care of local geometry through neighborhood interactions and estimate normal direction accordingly.

Fig. 2. (a) Non-linearly separable classes, (b) probable decision boundary points (in black), (c) TV based noise removal. Green points indicate refined points. (d) estimated local plane normals (thick arrows)



(a) (b)

(c) (d)

We then find a point whose normal vector is most aligned with the estimated plane normal direction. We use dot product operation to find the similarity between normal directions. That

point is considered as the representative plane point. If A is the plane normal direction and x is the representative plane point, then the equation of the plane in ND can be written as

$$Ax=b \tag{2}$$

where b is computed using estimated plane normal direction and plane point. The decision boundary can be a hyper plane i.e. for linearly separable case or can be a hyper surface i.e. for non-linearly separable case. As we do not know a priori whether the separating boundary is a hyper plane or a hyper surface, we first find a local plane equation as discussed above and evaluate all other decision boundary points against the estimated plane equation. A point is on an estimated plane if it satisfies following equation

$$Ax-b<=Threshold \tag{3}$$

To identify the points which are following Eq. (3), the value of Threshold is set close to zero. We eliminate points which satisfy Eq.(3) as they are redundant. For the remaining decision boundary points, we iteratively perform the above procedure until all the points get modeled by their local planes. Thus, we get a set of local plane equations and corresponding plane points which facilitate the classification. Fig. 1(d) shows the modeling of a decision boundary with the approach described above. As separating boundary is a plane, all the points got modeled by a single plane equation as expected. Fig. 1(d) shows the estimated normal direction with the green arrow. Fig. 2(d) shows the separating surface with blue color and estimated normal directions are shown with thick arrows at the corresponding plane point positions. Note that as the separating boundary is a non-linear surface, it automatically got modeled by large number of local planes.

2.3 Classification

Once the separating boundary is completely modeled by local planes, we classify test data points based on local plane equations and corresponding plane points. Intuitively, data points in the direction of the normal belong to class 1 and data points in the opposite direction of normal belong to class 2. Given a test data point, we find the nearest plane point. We again use Euclidian distance as the closeness measure. Then a test data point is evaluated against the corresponding plane equation. We find the value of classification measure CM as

$$CM = Ay-b \tag{4}$$

where A and b are the parameters of the plane selected and y indicates the test data point. Then the test data point is assigned to class 1 or class 2 based on the value of Classification measure as follows

$$\textit{If (CM >=0) assign y to Class 1 Else assign y to Class 2} \tag{5}$$

Classification accuracy was 100% on test data for linearly separable (Fig. 1) as well as for non-linearly separable (Fig. 2) cases described above.

3. TV BASED BINARY CLASSIFICATION – SECOND APPROACH (TVBC 2)

We now describe our second classification approach using TV based on the intuition that a test data point will be closer and/or aligned to its true class. We utilize this property for classifying test data points more accurately. Our approach performs better than lazy learning approaches (such as KNN) as it takes into account orientation information along with proximity.

In the training phase, we perform TV on the training data points of each class separately. A tensor at each data point in the training set is computed. A tensor is represented by an ellipse in 2D and by an ellipsoid in ND. Using the eigenvalue decomposition of a tensor matrix, the eigenvectors and corresponding eigenvalues can be obtained. Later, at test time, if a test data point belongs to a class, it should lie in the direction of one of the tangents of the training data points belonging to its true class. So during the training phase, for each training data point, we need to find the eigenvectors which are normals and eigenvectors which are tangents. For each training data point an intrinsic dimensionality is estimated based on the eigenvalue differences of consecutive eigenvalues arranged in descending order as in [1]: if intrinsic dimensionality is k, then first $(N-k)$ eigenvectors are taken as normals and the remaining k eigenvectors as tangents [1]. For each test data point, we first compute a vector V with respect to each training data point from class 1. The vector is computed as

$$V = \textit{test data point} - \textit{training data point} \tag{6}$$

If a test data point actually belongs to class 1, vector V should be aligned to one of the tangents of this training data point. We evaluate this alignment by simply using the dot product of V with a tangent (as defined above) of the training data point under consideration. We compute this dot product with each tangent of the training data point and pick the tangent that gives maximum value among all dot products. Using the best aligned tangent of the training data point, the similarity measure for a single test data point with respect to the training data point under consideration is defined as

$$\text{Similarity measure 1} = \text{dot product/norm (V)} \tag{7}$$

This similarity measure for a test data point with respect to class 1 is chosen to be the maximum of all its similarity measures with respect to each training data point of class 1.

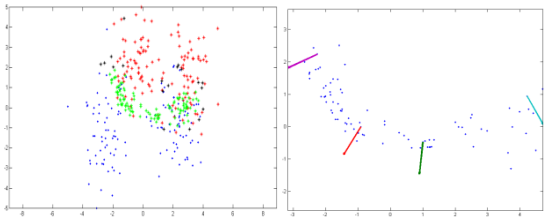
Similarly, we compute the Similarity measure 2 with respect to class 2. A test data point is then assigned to a class which has a better similarity measure value. Note that the similarity measure we use has both proximity and orientation context, and therefore this simple classifier is able to classify data points more accurately. In spite of the simplicity, our following experimentation results show how well it performs on well-known datasets with respect to established classifiers.

4. EXPERIMENTATION

We start demonstration with 2D real data (Ripley) [17], which has 250 training data points and 1000 test data points. Fig.3 (a) shows the training data used where class 1 and class 2 are represented by blue and red points respectively. We obtained the noise removed probable decision boundary points (shown in green) as described in section 2 (shown in Fig. 3(a)). Fig.

3(b) shows the normal of local planes/lines estimated by (with thick arrows) using the TV based approach. Note that the decision boundary is non-linear in nature and hence was modeled by 4 local planes/lines. Test data points were then evaluated based on the plane/line normal equations. Classification accuracy for this case was 89% on the test set.

Fig. 3. (a) Ripley training data, (b) estimated normals of local planes (thick arrows)



(a) (b)

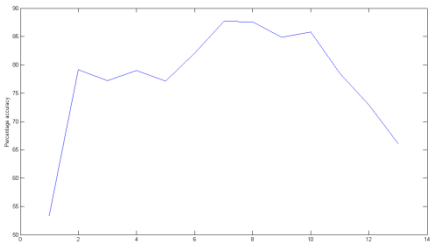
Table 1. Accuracy and Speed Comparison (bold face indicates best result)

Dataset	DT	KNN	TVBC1	TVBC2
Iris 1 [3]	99.33 (4.12s)	**100** (5.52s)	**100** (1.9s)	**100** (6.2s)
Iris 2 [3]	95.33 (4.3s)	96 (5.7s)	**96.67** (1.9s)	**96.67** (6.1s)
SVM guide3 [3]	80.68 (8.9s)	80.13 (10.3s)	87.65 (10s)	**88** (11.1s)
Solar flare [3]	66.61 (4.72s)	66.8 (5.07s)	68.6 (2.8s)	**68.7** (5.5s)
Ripley [17]	90 (0.5s)	**91.5** (1.8s)	89 (0.3s)	89.5 (2.4s)
Liver [3]	64.62 (5.2s)	64.02 (7s)	**76.4** (5.4s)	66.16 (7.3s)

Our approach is generic in nature and is valid for any dimension space. We performed experiments on other real datasets as well from UCI repository [3]: Iris 1, Iris 2, SVM guide3, Solar flare, Liver. Iris 1 is classification of class ”setosa” vs others, Iris 2 is classification of class “Virginica” vs others. Table I shows the comparison results of our TVBC1/TVBC2 approaches with Decision Trees (DT), KNN classifiers in terms of 10-fold cross validation accuracy and speed. We used WEKA’s [4] implementation of C4.5 DT, KNN for comparison. Our approach was implemented in Matlab on 2.8 GHz P4 processor with 4 GB RAM. Total testing time across 10-folds (in seconds) is given in the bracket below the accuracy count. The only variable

parameter for both our approaches based on TV framework is the scale of voting. Higher sigma (i.e. scale of voting) corresponds to larger extent of neighborhood and thus more smoothness. In the case of TVBC1, larger sigma may not be able to model minute structural variations (if any) in the decision boundary. We iteratively increase the value of sigma in steps of 0.01, model the decision boundary and perform 10-fold classification. We then select the sigma which gives maximum accuracy. The plot of change in accuracy with respect to the value of sigma for SVM guide3 [3] dataset is shown in Fig. 4. It can be seen that, accuracy does not vary significantly with change in the value of sigma. We observed that, for linearly separable case, classification accuracy is independent of value of sigma over a large range, as expected.

Fig. 4. Sigma vs accuracy: X-axis: Scale of voting, Y-axis: Percentage accuracy for SVM guide3 [3] dataset



5. CONCLUSION

We proposed two novel binary classification methods (TVBC1/TVBC2) based on tensor voting framework.

In the first approach, the decision boundary was modeled by local planes and test data points were classified based on estimated local plane equations. Our approach is robust, computationally efficient and gives almost same accuracy as other known classifiers.

In the second approach, during training phase we pre-compute the relevant tangent directions at each training data point using tensor voting framework. In test phase, we assign a test data point to one of the classes based on a novel similarity measure that considers both proximity and orientation. We validated the proposed approach on real world datasets and it performs at par/better than other well-known classifiers. We are investigating some approaches to bring the time complexity further down.

REFERENCES

[1] Mordohai P., Medioni G., Dimensionality Estimation, Manifold Learning and Function Approximation using Tensor Voting, JMLR, 2010
[2] Haibin, C., et al.,, Localized Support Vector Machine and Its Efficient Algorithm SIAM Conf. on Data Mining, 2007
[3] Frank A., et al., UCI Machine Learning Repository: http://archive.ics.uci.edu, CA, 2010
[4] Hall M., et al., The WEKA Data Mining Software: An Update; SIGKDD Explorations, 2009
[5] Burgess, C., Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery,1998
[6] Cortes, C., Vapnik, V. N., Support Vector Networks, Machine Learning, 1995

[7] Guy G., Medioni, G., Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3-d data. IEEE Trans. on PAMI, 1997
[8] B. E. Boser, I. M. Guyon, and V .Vapnik. A training algorithm for optimal margin classifiers. In Fifth Annual Workshop on Computational Learning Theory, pages 144–152, Pittsburgh, 1992. ACM
[9] C.J.C. Burges, P. Knirsch, and R. Haratsch. Support vector web page: http://svm.research.bell-labs.com. Technical report, Lucent Technologies, 1996
[10] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias / variance dilemma. Neural Computation, 4:1–58, 1992
[11] C.M. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995.
[12] Gionis, A., Indyk, P., Motwani, R., Similarity Search in High Dimensions via Hashing, Proceedings of the 25th Very Large Database Conference (VLDB), 1999
[13] Tensor voting implementation available at: http://iris.usc.edu/people/medioni/ntensorvoting.html
[14] G. Medioni, M.S. Lee, C.K. Tang, A computational framework for segmentation and grouping (Elsevier, New York, NY, 2000)
[15] P. Mordohai, G. Medioni, Unsupervised dimensionality estimation and manifold learning in high dimensional spaces by tensor voting, International Joint Conference on Artificial Intelligence, 2005
[16] Duda, Hart, and Stork, Pattern Classification, Wiley
[17] B. D. Ripley and N. L. Hjort, Pattern Recognition Neural Networks, Cambridge University Press, 1995