본 연구는 서울과학기술대학교 교내 일반과제 연구비 지원으로 수행되었습니다.

종신회원: 서울과학기술대학교 전자IT미디어공학과 교수 (Seoul Tech Univ.)
gyemin@seoultech.ac.kr

논문접수: 2014년 9월 16일
(Received 16 September 2014)

논문수정: 2014년 10월 24일
(Revised 24 October 2014)

심사완료: 2014년 10월 27일
(Accepted 27 October 2014)

© 2015 한국정보과학회. 개인 목적이나 교육 목적으로 본 저작물의 일부 또는 전체를 재배포할 수 있습니다. 다른 용도의 이용은 저작권자의 허락 없이 사용할 수 없습니다.
level set of the density. The child nodes of a given parent node correspond to separated regions of the level set at a higher level than the level of the parent node. Figure 1 illustrates an example of the level set tree corresponding to a density function. The level set tree reveals the shape and the number of modes of the density. It also facilitates clustering analysis.

In this paper, we present a level set tree estimation algorithm. Our algorithm finds the tree representation from a family of density level sets, which are estimated from a given set of data points.

Rather than estimating level sets through density estimation followed by thresholding, we use one-class support vector machine (OC-SVM) [4,5] to directly estimate density level sets. This development is based on the recent work by Vert and Vert [6] that the OC-SVM with the Gaussian kernel is a consistent density level set estimator. Since the free parameter $\lambda$ of the OC-SVM implicitly determines which level set is estimated, a whole class of density level sets can be generated by varying $\lambda$ from 0 to infinity. This would normally incur a huge computational burden because of repeated OC-SVM training for each value of $\lambda$. However, the solution path algorithm greatly reduces the computational cost and time to obtain multiple level sets over the entire range of levels [7,8].

Because we use the OC-SVM for density level set estimation, our method is nonparametric in nature. We also make no explicit assumption on the shapes of clusters or on the number of clusters. The resulting level set tree visualizes the data structure, and can be used for further analysis.

We first briefly review the OC-SVM in Section 2, and its solution path algorithm in Section 3. Then we propose an algorithm to construct level set tree from the OC-SVM in Section 4. The proposed algorithm is demonstrated in Section 5. Conclusion follows in Section 6.

2. One-Class Support Vector Machines

The one-class support vector machines (OC-SVM) are proposed in [4,5] as a support vector method to estimate a set where most of a given random sample $\{x_1, x_2, \ldots, x_n\}, \ x_i \in \mathbb{R}^d$ resides in the space. In OC-SVM, each $x_i$ is first mapped to a high dimensional space $\mathcal{H}$ through a function $\Phi: \mathbb{R}^d \rightarrow \mathcal{H}$. An inner product in $\mathcal{H}$ can be evaluated with a kernel function $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$. The OC-SVM finds a hyperplane in $\mathcal{H}$ that maximally separates the data from the origin. The distance from the hyperplane to the origin is called the margin. To maximize the margin, the OC-SVM allows some data points inside the margin by introducing non-negative penalties $\xi_i$.

More specifically, the OC-SVM solves the following optimization problem:

$$\min_{\varepsilon} \frac{1}{2} \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i$$

subject to $\langle w, \Phi(x_i) \rangle \geq 1 - \xi_i, \ \xi_i \geq 0$ for $i = 1, 2, \ldots, n$

where $w \in \mathcal{H}$ is the normal vector of the hyperplane and $\lambda$ is the parameter that controls the margin violations.

In practice, the quadratic program is solved via its dual:

$$\min_{\alpha} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K_{ij} - \sum_i \alpha_i$$

subject to $0 \leq \alpha_i \leq 1$ for all $i$

where $K_{ij} = k(x_i, x_j)$ is the kernel matrix, and $\alpha_i$ is a Lagrange multiplier corresponding to $x_i$. For the optimal solution $\alpha$, the normal vector of the hyperplane is given by

$$w = \frac{1}{\lambda} \sum_i \alpha_i \Phi(x_i),$$

and $\{x : f(x) = 1\}$ defines the decision boundary, where
\[ f(\mathbf{x}) = (\mathbf{w}, \Phi(\mathbf{x})) = \frac{1}{\lambda} \sum_{j} \alpha_{j} k(\mathbf{x}_{i}, \mathbf{x}), \quad (1) \]

From the Karush–Kuhn–Tucker conditions [9], the following conditions also hold:

\[ f(\mathbf{x}_{i}) > 1 \quad \Rightarrow \quad \xi_{i} = 0, \quad \alpha_{i} = 0 \]
\[ f(\mathbf{x}_{i}) = 1 \quad \Rightarrow \quad \xi_{i} = 0, \quad \alpha_{i} \in [0, 1] \]
\[ f(\mathbf{x}_{i}) < 1 \quad \Rightarrow \quad \xi_{i} > 0, \quad \alpha_{i} = 1 \]

More detailed discussion on SVMs is available in [10].

3. OC–SVM Solution Path Algorithm

For support vector classification, Hastie et. al. [7] showed that the Lagrange multipliers \( \alpha_{i} \) are piecewise linear in \( \lambda \). Lee and Scott [8] extended the method to the OC–SVM to derive the solution path algorithm.

When OC–SVM solutions are necessary for a range of \( \lambda \), an approach is to solve the dual repeatedly. However, it demands huge computational cost and time. On the other hand, the OC–SVM path algorithm finds the entire solution sets over the whole range of \( \lambda \) by exploiting the piecewise-linearity. Thus, it greatly reduces the required training time and is suitable for our purpose.

The path algorithm finds the whole set of solutions by decreasing \( \lambda \) from a large value toward zero. For sufficiently large \( \lambda \), all the data points fall between the hyperplane and the origin so that \( f(\mathbf{x}_{i}) \leq 1 \). As \( \lambda \) decreases, the margin width decreases, and data points cross the hyperplane (\( f(\mathbf{x}_{i}) = 1 \)) to move outside the margin (\( f(\mathbf{x}_{i}) > 1 \)). Throughout this process, the OC–SVM solution path algorithm monitors the changes of the following subsets:

\[ \mathcal{R} = \{ i : f(\mathbf{x}_{i}) > 1, \alpha_{i} = 0 \} \]
\[ \mathcal{E} = \{ i : f(\mathbf{x}_{i}) = 1, 0 \leq \alpha_{i} \leq 1 \} \]
\[ \mathcal{L} = \{ i : f(\mathbf{x}_{i}) < 1, \alpha_{i} = 1 \} \]

3.1 Initialization

We first establish the initial state of the sets defined above. For sufficiently large \( \lambda \), every data point \( \mathbf{x}_{i} \) falls inside the margin; that is, \( f(\mathbf{x}_{i}) \leq 1 \). Then the KKT condition implies \( \alpha_{i} = 1 \) or \( \forall i \), and we obtain \( \lambda \geq \sum_{j} K_{ij} \) for \( \forall i \) from Equation (1). Thus, if

\[ \lambda_{0} = \max_{i} \sum_{j} K_{ij} \]

denotes the maximum row sum of the kernel matrix, then for any \( \lambda \geq \lambda_{0} \), the optimal solution of OC–SVM becomes \( \alpha_{i} = 1 \) for \( \forall i \). Therefore, the path algorithm sets the initial value of \( \lambda \) to \( \lambda_{0} \). Then all the data points are in the subset \( \mathcal{L} \) and the corresponding Lagrange multipliers are \( \alpha_{i} = 1 \).

3.2 Tracing the Path

As \( \lambda \) decreases, either of the following events can occur:

A. A point enters \( \mathcal{E} \) from \( \mathcal{L} \) or \( \mathcal{R} \).

B. A point leaves \( \mathcal{E} \) to enter \( \mathcal{L} \) or \( \mathcal{R} \).

Let \( \lambda_{l} \) and \( \alpha_{l} \) denote the parameter values right after the \( l \)-th event and \( f'(\mathbf{x}) \) the decision function at this point. Define \( \mathcal{E}_{l} \) similarly and suppose \( |\mathcal{E}_{l}| = m \).

Recall that

\[ f(\mathbf{x}) = \frac{1}{\lambda} \sum_{i} \alpha_{i} k(\mathbf{x}_{i}, \mathbf{x}) \]

Then, for \( \lambda_{l} > \lambda > \lambda_{l+1} \), we can write

\[ f(\mathbf{x}) = \left[ f(\mathbf{x}) - \frac{\lambda_{l}}{\lambda} f'(\mathbf{x}) \right] + \frac{\lambda_{l}}{\lambda} f'(\mathbf{x}) = \frac{1}{\lambda} \left[ \sum_{j \in \mathcal{E}_{l}} (\alpha_{j} - \alpha') k(\mathbf{x}_{j}, \mathbf{x}) + \lambda_{l} f'(\mathbf{x}) \right]. \quad (2) \]

The second equality holds because for this range of \( \lambda \) only points in \( \mathcal{E}_{l} \) change their \( \alpha_{j} \). On the contrary, all other points in \( \mathcal{R}_{l} \) or \( \mathcal{L}_{l} \) have their \( \alpha_{j} \) fixed to 0 or 1, respectively. Since \( f(\mathbf{x}_{j}) = 1 \) for all \( i \in \mathcal{E}_{l} \), we have

\[ \sum_{j \in \mathcal{E}_{l}} \delta_{j} k(\mathbf{x}_{j}, \mathbf{x}_{i}) = \lambda_{l} - \lambda, \quad i \in \mathcal{E}_{l} \quad (3) \]

where \( \delta_{j} = \alpha'_{j} - \alpha_{j} \). Now let \( K_{ij} \) denote the \( m \times m \) matrix with elements \( [K_{ij}]_{ij} = k(\mathbf{x}_{i}, \mathbf{x}_{j}) \) for \( i, j \in \mathcal{E}_{l} \).

Then Equation (3) becomes \( K_{l} \delta = (\lambda_{l} - \lambda) \mathbf{1} \). If \( K_{l} \) has full rank, we obtain \( \mathbf{b} = K_{l}^{-1} \mathbf{1} \), and hence

\[ \alpha_{j} = \alpha'_{j} - (\lambda_{l} - \lambda) b_{j}, \quad j \in \mathcal{E}_{l}. \quad (4) \]

Then from Equation (2) we have
\[ f(x) = \frac{\lambda}{\lambda'} \left[ f'(x) - h'(x) \right] + h'(x) \]  

where \[ h'(x) = \sum_{j \in \mathcal{E}} b_j k(x, x). \]

This result shows that the Lagrange multipliers \( \alpha_j \) for \( j \in \mathcal{E} \) change piecewise-linearly in \( \lambda \). If \( K_i \) is not invertible, the solution paths for some of the \( \alpha_i \) are not unique. These cases are rare in practice and discussed more in [7].

3.3 Finding the Next Breakpoint

The \((l+1)^{st}\) event occurs when:

A. Some \( x_j \) for which \( j \in \mathcal{L}_l \cup \mathcal{R}_l \) enters the hyperplane so that \( f(x_j) = 1 \). From Equation (5), this event occurs at
\[ \lambda = \lambda_j \frac{f'(x_j) - h'(x_j)}{1 - h'(x_j)}. \]

B. Some \( \alpha_j \) for which \( j \in \mathcal{E}_j \) reaches 0 or 1. From Equation (4), this case, respectively, corresponds to
\[ \lambda = -\frac{\alpha_j'}{b_j} + \frac{\lambda b_j}{b_j}, \quad \lambda = -\frac{1 - \alpha_j'}{b_j} + \frac{\lambda b_j}{b_j}. \]

The next event occurs at the largest \( \lambda < \lambda_j \).

The path algorithm stops when the set \( \mathcal{L} \) becomes empty or the value of \( \lambda \) is smaller than a pre-specified value.

4. Level Set Tree Algorithm

The level set tree is a useful visualization technique that reveals the hierarchical structure of a data set [1,2]. In this section, we present an algorithm that uses OC-SVM to estimate the level set tree.

4.1 Level Set Estimation

As discussed above, a decision function from the OC-SVM solution path algorithm can be interpreted as a density level set estimator:
\[ \hat{L}_\lambda = \{ x : f_\lambda(x) = \frac{1}{\lambda} \sum \alpha_i(\lambda) k(x_i, x) > 1 \}. \]

However, the experiments in Section 5 show that this level set estimator may not produce nested level set estimates, while they should. We modify the estimator:
\[ \hat{L}_\lambda = \bigcup_{\mu \in \lambda} \hat{L}_\mu^\prime. \]

Then it is clear that these sets are nested: the set estimate at \( \lambda_2 \) completely contains the set estimate at \( \lambda_1 \) for \( \lambda_1 > \lambda_2 \). Thus, the output from the path algorithm becomes nested multiple level set estimates.

4.2 Level Set Tree Estimation

From the family of level set estimates, we construct a level set tree \( T \). Our approach is top-down and proceeds from leaf nodes at the highest level to root nodes at the lowest level. We associate every tree node \( N \) with a subset of data points and a value \( \lambda \). Algorithm 1 describes our level set tree estimation algorithm.

\begin{algorithm}[h]
1. input: \{\( x_1, x_2, \ldots, x_n \)\}
2. \( (\lambda_0, \alpha_0) \) ← path algorithm(\( \{x_1, x_2, \ldots, x_n\} \))
3. \( \text{nid} \leftarrow 0 \)
4. for each \( \mu \)
5. \( L_{\mu} \leftarrow \{ x_j : f_\mu(x_j) \geq 1, \mu \geq \lambda_0 \} \)
6. \( L_{\mu}^\text{new} \leftarrow L_{\mu} - L_{\mu-1} \)
7. for each \( x_j \in L_{\mu}^\text{new} \)
8. Find nodes without parent connected to \( x_j \).
9. if \( x_j \) is connected to none of the nodes, then \( \text{nid} \leftarrow \text{nid} + 1 \).
10. Create a new node \( N_{\text{nid}} = \{x_j\} \).
11. else if \( x_j \) is connected to only one node, say \( N_i \), then \( N_i \leftarrow N_i \cup \{x_j\} \).
12. else if \( x_j \) is connected to more than one nodes, say \( N_{1}, \ldots, N_m \), then \( \text{nid} \leftarrow \text{nid} + 1 \).
13. Create a new node \( N_{\text{nid}} = N_i \cup \cdots \cup N_m \cup \{x_j\} \).
14. Set \( N_{\text{nid}} \) be the parent of \( N_{1}, \ldots, N_m \).
15. end for
16. end for
17. output: Level set tree \( T = \{N_1, N_2, \ldots, N_{\text{nid}}\} \).
\end{algorithm}

The solution path outputs a set of breakpoints \( \lambda \) and \( \alpha_i \). Each pair \( (\lambda_i, \alpha_i) \) generates a level set estimate \( L_{\lambda_i} \). This level set consists of one or more maximally connected subsets called clusters.
At step $k$, the algorithm finds the data points newly added to $L_k$ (line 6). Each of such data points $x_j$ is tested in line 8 if it is connected to any parentless nodes. Based on the connectivity, three cases arise:

1. if $x_j$ is connected to none of the nodes, then a new leaf node is created with an element $x_j$ and is associated with the value $\lambda_k$,

2. if $x_j$ is connected to exactly one tree node, say $N_1$, then the data point set associated to $N_1$ grows to include $x_j$, and

3. if $x_j$ is connected more than one nodes, say $N_1, \ldots, N_m$, then these nodes are merged to create a new internal tree node $N$. This node $N$ becomes the parent node of $N_1, \ldots, N_m$.

The process continues to the lowest level set estimate to form root nodes.

In line 8, we use a geometric approach to determine the connectivity of a data point $x_j$ to a tree node $N$. A data point $x_j$ is judged to be connected to a node $N$ if a path exists between $x_j$ and any data points in the set associated to $N$. If it is not the case, such a path will cross the boundary $(f(x) = \lambda)$ and contain a segment of points $y$ such that $f(y) < \lambda$. In practice, a number of points can be sampled to check the line segment. We sampled 20 points in our implementation.

The level set tree obtained from this procedure will show the structure of the underlying probability density function. A leaf node indicates to a mode of the distribution. Branches at a level imply the existence of separated regions or clusters in the corresponding level set. An internal node means merging of two or more smaller clusters to a larger cluster. Thus, it manifests the hierarchical order of clusters.

5. Experiments

We demonstrate the proposed level set tree estimation algorithm on two data sets: “multi” and “banana”. The former is a three-component Gaussian mixture data of 200 data points. The latter is a benchmark data available online.\(^1\)

In our experiments, we use Gaussian kernel

$$k(x, x') = \exp\left(-\frac{\|x - x'|^2}{2\sigma^2}\right),$$

since this kernel induces a feature space, which maps all data points into the same orthant on a hypersphere [10]. Thus, the OC-SVM principle, separating data from the origin, is justified.

Our MATLAB implementation of the OC-SVM path algorithm and the level set tree algorithm are available online at http://sites.google.com/site/gyeminlee/codes.

The level set estimates from the OC-SVM solution path algorithm in Section 3 are illustrated in Figure 2. The figure shows two level sets from the path algorithm at two different $\lambda$ values. In the figure, each small circle represents a data point in “multi” data. As can be seen, however, the original OC-SVM set estimates are not nested. Thus, we modify the sets as presented in Section 4.1. The modified level sets are properly nested in Figure 2(b).

Figure 3 displays five nested level set estimates from “multi” data. The contours correspond to the set boundaries at five different levels. Inner contours

\(^1\) http://mldata.org/repository/tags/data/IDA_Benchmark_Repository/
그림 3 5개의 서로 다른 \( \lambda \) 값에서 추정한 “multi” 데이터의 nested 레벨 셋. 안쪽의 윤곽선은 높은 레벨에 바깥쪽의 윤곽선은 낮은 레벨에 해당한다. 레벨 셋들이 nested되어 있다.

Fig. 3 Nested level set estimates of “multi” data at five different levels. Inner contours are at higher levels, and outer contours are at lower levels.

그림 4 Section 4에서 제안한 알고리즘을 통해 생성한 “multi” 데이터의 레벨 셋 트리. 각 수평선은 그림 3의 레벨 셋의 \( \lambda \) 값에 상응한다.

Fig. 4 The estimated level set tree of “multi” data. Each horizontal line indicates to the value of \( \lambda \) for the corresponding level set estimate in Fig. 3.

are at higher levels and completely contained in outer contours at lower levels; that is, the sets are nested. The three components in the data can be identified in the figure.

Using the level set tree algorithm proposed in Section 4, the level set tree is created from the family of level sets estimated from “multi” data (Figure 4). At the lowest level corresponding to the support of the density function is a single cluster. As \( \lambda \) increases, the tree bifurcates, and one of the branches further bifurcates. The three leaf nodes indicate to the three components in “multi”. This visualizes that the density function has three modes and the data can be grouped into three clusters.

The nested level sets at five different \( \lambda \) values and the level set tree of “banana” data is shown in Figure 5 and Figure 6, respectively. Contrary to “multi” data, “banana” data has two distinct clusters and this feature is captured by the two trees in Figure 6. The two branches in the left tree indicate the two subcomponents of the larger banana-shaped cluster in the upper region in Figure 5. The tree in the right means a single separated cluster in the lower region.

Therefore, the level set trees provide a useful representation visualizing the underlying structure of the data sets.

6. Conclusion

In this paper, we have presented a level set tree estimator. Our algorithm builds a level set tree estimates from the multiple OC-SVM level set
estimates. A family of level sets can be estimated from the OC-SVM solution path algorithm in a computationally efficient way. We demonstrated the proposed algorithm to visualize benchmark data sets. Future work may include comparing to other level set estimators such as kernel density estimation followed by thresholding, or other multiple level set estimators that naturally yield nested level set estimates [11].

References