CHAPTER

38 OUTLIER DETECTION

CHAPTER CONTENTS

The objective of *outlier detection* is to find outlying samples in input-only training samples $\{\boldsymbol{x}_i\}_{i=1}^n$. If the labels of inliers and outliers are available for $\{\boldsymbol{x}_i\}_{i=1}^n$, outlier detection can be formulated as supervised classification. However, since outliers may be highly diverse and their tendency may change over time, learning a stable decision boundary between inliers and outliers is often difficult in practice. In this chapter, various unsupervised outlier detection methods are introduced.

In Part 4, supervised learning methods that are *robust* against outliers were introduced. When the number of outliers is expected not to be too large, robust learning from a data set contaminated with outliers may be useful. On the other hand, if outliers are expected to be more abundant in the data set, removing them in advance by an outlier detection method would be more appropriate.

38.1 DENSITY ESTIMATION AND LOCAL OUTLIER FACTOR

A naive outlier detection method is based on *density estimation*. More specifically, the probability density $p(\boldsymbol{x})$ of samples $\{\boldsymbol{x}_i\}_{i=1}^n$ is estimated using, e.g. one of the density estimators described in Part 3, and then samples having low probability densities are regarded as outliers. However, since estimating the probability density in low-density regions is difficult due to the shortage of samples, such a density estimation approach may be unreliable for outlier detection purposes. The *local outlier factor* [23] is a stabilized variant of the density estimation approach that finds samples isolated from other samples.

Let us define the *reachability distance* (RD) from $\boldsymbol{x}$ to $\boldsymbol{x}'$ as

$$\mathrm{RD}_k(\boldsymbol{x},\boldsymbol{x}') = \max\left(\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|, \|\boldsymbol{x}-\boldsymbol{x}'\|\right),$$

An Introduction to Statistical Machine Learning. DOI: 10.1016/B978-0-12-802121-7.00049-2

where $\boldsymbol{x}^{(k)}$ is the kth nearest neighbor of $\boldsymbol{x}$ in $\{\boldsymbol{x}_i\}_{i=1}^n$. The RD can be regarded as a stabilized variant of the Euclidean distance $\|\boldsymbol{x}-\boldsymbol{x}'\|$ so that the distance is not less than $\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|$. Based on the RD, the *local RD* of $\boldsymbol{x}$ is defined as

$$\mathrm{LRD}_k(\boldsymbol{x}) = \left(\frac{1}{k}\sum_{i=1}^{k}\mathrm{RD}_k(\boldsymbol{x}^{(i)},\boldsymbol{x})\right)^{-1},$$

which is the inverse of the average RDs from $\boldsymbol{x}^{(i)}$ to $\boldsymbol{x}$. When $\boldsymbol{x}$ is isolated from surrounding samples, the local RD takes a small value.

The *local outlier factor* of $\boldsymbol{x}$ is defined as

$$\mathrm{LOF}_k(\boldsymbol{x}) = \frac{\frac{1}{k}\sum_{i=1}^{k}\mathrm{LRD}_k(\boldsymbol{x}^{(i)})}{\mathrm{LRD}_k(\boldsymbol{x})},$$

and $\boldsymbol{x}$ is regarded as an outlier if $\mathrm{LOF}_k(\boldsymbol{x})$ takes a large value. $\mathrm{LOF}_k(\boldsymbol{x})$ is the ratio of the average local RD of $\boldsymbol{x}^{(i)}$ and the local RD of $\boldsymbol{x}$, and $\boldsymbol{x}$ is regarded as an outlier if $\boldsymbol{x}^{(i)}$ is in a high-density region and $\boldsymbol{x}$ is in a low-density region. Conversely, if $\boldsymbol{x}^{(i)}$ is in a low-density region and $\boldsymbol{x}$ is in a high-density region, $\mathrm{LOF}_k(\boldsymbol{x})$ takes a small value and thus $\boldsymbol{x}$ is regarded as an inlier.

A MATLAB code for computing the local outlier factor is provided in Fig. 38.2, and its behavior is illustrated in Fig. 38.1. This shows that isolated samples tend to have large outlier scores. However, the behavior of the local outlier factor depends on the choice of nearest neighbors, k, and it is not straightforward to optimize k in practice.

38.2 SUPPORT VECTOR DATA DESCRIPTION

Support vector data description [109] is an unsupervised outlier detection algorithm that does not involve explicit density estimation.

Let us consider a *hypersphere* with center $\boldsymbol{c}$ and radius $\sqrt{b}$ on $\mathbb{R}^d$, and learn $\boldsymbol{c}$ and b to include all training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ (i.e., the *minimum enclosing ball* is found):

$$\min_{\boldsymbol{c},b} b \quad \text{subject to } \|\boldsymbol{x}_i-\boldsymbol{c}\|^2 \le b \quad \text{for } i=1,\ldots,n. \tag{38.1}$$

Note that, not the radius $\sqrt{b}$, but the squared radius b is optimized for the convexity of the optimization problem [26].

Support vector data description is a relaxed variant of the minimum enclosing ball problem which finds a hypersphere that contains *most* of the training samples $\{\boldsymbol{x}_i\}_{i=1}^n$ (Fig. 38.3):

```
n=100; x=[(rand(n/2,2)-0.5)*20; randn(n/2,2)]; x(n,1)=14;
k=3; x2=sum(x.^2,2);
[s,t]=sort(sqrt(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x'),2);

for i=1:k+1
  for j=1:k
    RD(:,j)=max(s(t(t(:,i),j+1),k),s(t(:,i),j+1));
  end
  LRD(:,i)=1./mean(RD,2);
end
LOF=mean(LRD(:,2:k+1),2)./LRD(:,1);

figure(1); clf; hold on
plot(x(:,1),x(:,2),'rx');
for i=1:n
  plot(x(i,1),x(i,2),'bo','MarkerSize',LOF(i)*10);
end
```

FIGURE 38.1

MATLAB code for local outlier factor.

$$\begin{aligned} &\min_{\boldsymbol{c},b,\boldsymbol{\xi}} \left[b + C \sum_{i=1}^{n} \xi_i \right] \\ &\text{subject to } \|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \le b + \xi_i \quad \text{for } i = 1,\ldots,n, \\ &\qquad\qquad \xi_i \ge 0 \quad \text{for } i = 1,\ldots,n, \\ &\qquad\qquad b \ge 0, \end{aligned} \tag{38.2}$$

where $C > 0$ controls the number of training samples included in the hypersphere and ξ_i is the *margin error* for $\boldsymbol{x}_i$. Samples outside the hypersphere are regarded as outliers, i.e., a test sample $\boldsymbol{x}$ is regarded as an outlier if

$$\|\boldsymbol{x} - \widehat{\boldsymbol{c}}\|^2 > \widehat{b}, \tag{38.3}$$

where $\widehat{\boldsymbol{c}}$ and $\widehat{b}$ are the solutions of optimization problem (38.2).

Optimization problem (38.2) (and also minimum enclosing ball problem (38.1)) has quadratic constraints, and directly handling them in optimization can be computationally expensive. Below, let us consider its *Lagrange dual* problem (Fig. 23.5).

Given the fact that the constraint $b \ge 0$ can be dropped without changing the solution when $C > 1/n$ [27], the Lagrange dual of optimization problem (38.2) is given by

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \inf_{\boldsymbol{c},b,\boldsymbol{\xi}} L(\boldsymbol{c},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) \text{ subject to } \boldsymbol{\alpha} \ge \boldsymbol{0},\ \boldsymbol{\beta} \ge \boldsymbol{0},$$

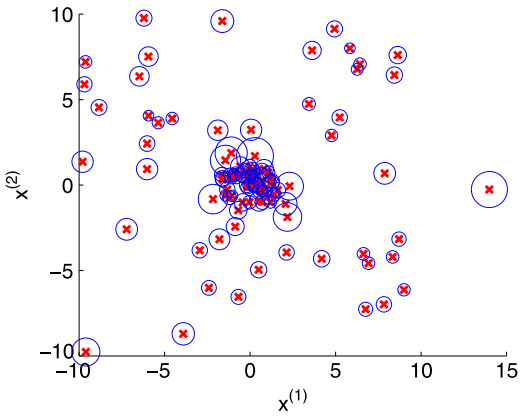


FIGURE 38.2

Example of outlier detection by local outlier factor. The diameter of circles around samples is proportional to the value of local outlier factor.

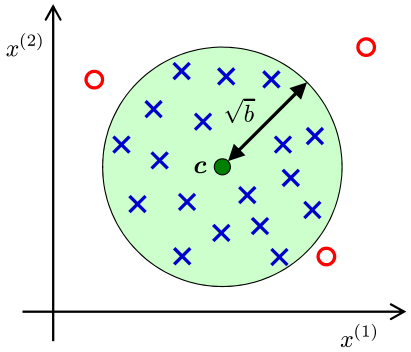


FIGURE 38.3

Support vector data description. A hypersphere that contains *most* of the training samples is found. Samples outside the hypersphere are regarded as outliers.

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the Lagrange multipliers and $L(\boldsymbol{c}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ is the Lagrange function defined by

$$L(\boldsymbol{c}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = b + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(b + \xi_i - \|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \right) - \sum_{i=1}^{n} \beta_i \xi_i.$$

The first-order optimality conditions of $\inf_{\boldsymbol{c},b,\boldsymbol{\xi}} L(\boldsymbol{c},b,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta})$ yield

$$\begin{aligned}
&\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i = 1, \\
&\frac{\partial L}{\partial \boldsymbol{c}} = 0 \implies \boldsymbol{c} = \frac{\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i}{\sum_{i=1}^{n} \alpha_i} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i, \\
&\frac{\partial L}{\partial \xi_i} = 0 \implies \alpha_i + \beta_i = C, \quad \forall i = 1,\ldots,n,
\end{aligned} \tag{38.4}$$

and thus the Lagrange dual problem can be expressed as

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{n} \alpha_i Q_{i,i} - \sum_{i,j=1}^{n} \alpha_i \alpha_j Q_{i,j} \right] \\
&\text{subject to } 0 \le \alpha_i \le C \quad \text{for } i = 1,\ldots,n, \\
&\sum_{i=1}^{n} \alpha_i = 1,
\end{aligned} \tag{38.5}$$

where

$$Q_{i,j} = \boldsymbol{x}_i^\top \boldsymbol{x}_j.$$

This is a *quadratic programming* problem (Fig. 27.5), which can be efficiently solved by standard optimization software. However, note that the above quadratic programming problem is convex only when the $n \times n$ matrix $\boldsymbol{Q}$ is nonsingular. When it is singular, $\boldsymbol{Q}$ may be ℓ_2-regularized by adding a small positive constant to the diagonal elements (see Section 23.2) in practice.

Note that, if $C > 1$, optimization problem (38.2) is reduced to minimum enclosing ball problem (38.1), meaning that the solution does not depend on C [27]. This resembles the relation between *hard margin* support vector classification and *soft margin* support vector classification introduced in Chapter 27. On the other hand, when $0 < C \le 1/n$, $b = 0$ is actually the solution [27], which is not useful for outlier detection because all samples are regarded as outliers. Thus, support vector data description is useful only when

$$\frac{1}{n} < C \le 1.$$

From the KKT conditions (Fig. 27.7) of dual optimization problem (38.5), the following properties hold in the same way as support vector classification (Chapter 27):

- $\alpha_i = 0$ implies $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \leq b$.
- $0 < \alpha_i < C$ implies $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 = b$.
- $\alpha_i = C$ implies $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 \geq b$.
- $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 < b$ implies $\alpha_i = 0$.
- $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 > b$ implies $\alpha_i = C$.

Thus, $\boldsymbol{x}_i$ is on the surface or in the interior of the hypersphere when $\alpha_i = 0$, $\boldsymbol{x}_i$ lies on the surface when $0 < \alpha_i < C$, and $\boldsymbol{x}_i$ is on the surface or is in the exterior of the hypersphere when $\alpha_i = C$. On the other hand, $\alpha_i = 0$ when $\boldsymbol{x}_i$ is in the strict interior of the hypersphere and $\alpha_i = C$ when $\boldsymbol{x}_i$ is in the strict exterior of the hypersphere.

Similarly to the case of support vector classification, sample $\boldsymbol{x}_i$ such that $\widehat{\alpha}_i > 0$ is called a *support vector*. From Eq. (38.4), the solution $\widehat{\boldsymbol{c}}$ is given by

$$\widehat{\boldsymbol{c}} = \sum_{i:\widehat{\alpha}_i > 0} \widehat{\alpha}_i \boldsymbol{x}_i.$$

Since $\|\boldsymbol{x}_i - \boldsymbol{c}\|^2 = b$ holds for $\boldsymbol{x}_i$ such that $0 < \alpha_i < C$, the solution $\widehat{b}$ is given by

$$\widehat{b} = \left\|\boldsymbol{x}_i - \widehat{\boldsymbol{c}}\right\|^2.$$

As explained in Eq. (38.3), with the solutions $\widehat{\boldsymbol{c}}$ and $\widehat{b}$, a test sample $\boldsymbol{x}$ is regarded as an outlier if, for i such that $0 < \widehat{\alpha}_i < C$,

$$\begin{aligned} \|\boldsymbol{x} - \widehat{\boldsymbol{c}}\|^2 - \widehat{b} &= \|\boldsymbol{x} - \widehat{\boldsymbol{c}}\|^2 - \left\|\boldsymbol{x}_i - \widehat{\boldsymbol{c}}\right\|^2 \\ &= \boldsymbol{x}^\top \boldsymbol{x} - 2\sum_{j=1}^{n} \widehat{\alpha}_j \boldsymbol{x}^\top \boldsymbol{x}_j - a_i > 0, \end{aligned} \tag{38.6}$$

where

$$a_i = \boldsymbol{x}_i^\top \boldsymbol{x}_i - 2\sum_{j=1}^{n} \widehat{\alpha}_j \boldsymbol{x}_i^\top \boldsymbol{x}_j.$$

Note that a_i can be computed in advance independent of the test sample $\boldsymbol{x}$.

In the same way as support vector classification, support vector data description can be nonlinearized by the *kernel trick* (Section 27.4). More specifically, for kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$, Lagrange dual problem (38.5) becomes

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)\right] \\ &\text{subject to } 0 \leq \alpha_i \leq C \quad \text{for } i = 1, \ldots, n, \\ &\qquad \sum_{i=1}^{n} \alpha_i = 1, \end{aligned}$$

```
n=50; x=randn(n,2); x(:,2)=x(:,2)*4; x(1:20,1)=x(1:20,1)*3;
C=0.04; h=[C*ones(n,1); zeros(n,1); 1; -1];
H=[eye(n); -eye(n); ones(1,n); -ones(1,n)]; x2=sum(x.^2,2);
K=exp(-(repmat(x2,1,n)+repmat(x2',n,1)-2*x*x'));
a=quadprog(K,zeros(n,1),H,h); s=ones(n,1)-2*K*a;
s=s-mean(s(find((0<a)&(a<C)))); u=(s>0.001);

figure(1); clf; hold on; axis equal;
plot(x(:,1),x(:,2),'rx'); plot(x(u,1),x(u,2),'bo');
```

FIGURE 38.4

MATLAB code of support vector data description for Gaussian kernel. `quadprog.m` included in Optimization Toolbox is required. Free alternatives to `quadprog.m` are available, e.g. from http://www.mathworks.com/matlabcentral/fileexchange/.

and outlier criterion (38.6) becomes

$$K(\boldsymbol{x},\boldsymbol{x}) - 2\sum_{j=1}^{n} \widehat{\alpha}_j K(\boldsymbol{x},\boldsymbol{x}_j) - a_i > 0,$$

where, for i such that $0 < \widehat{\alpha}_i < C$,

$$a_i = K(\boldsymbol{x}_i,\boldsymbol{x}_i) - 2\sum_{j=1}^{n} \widehat{\alpha}_j K(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

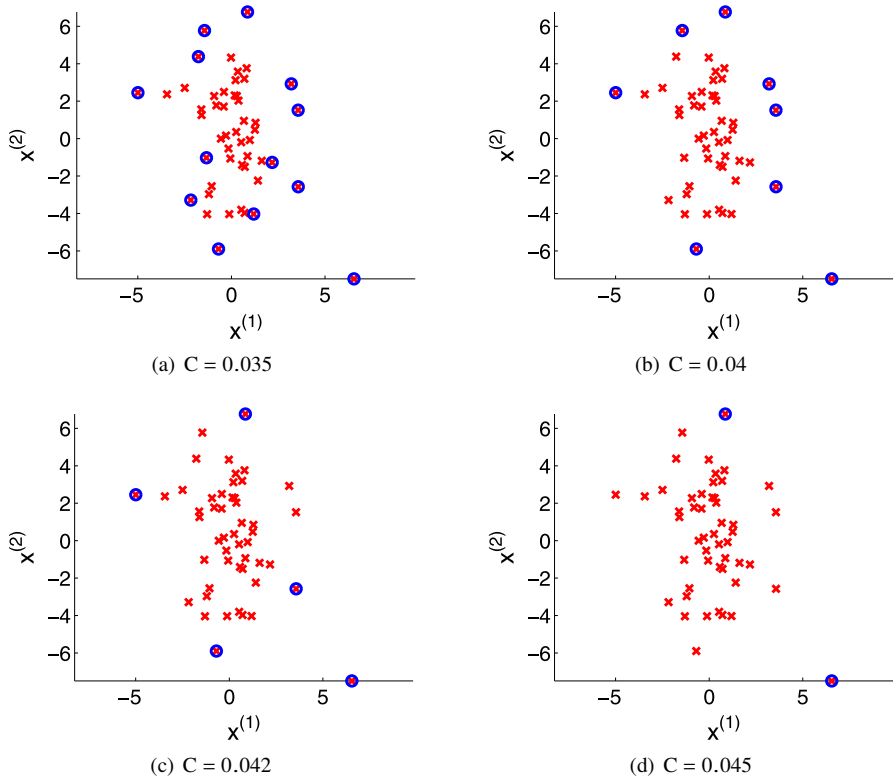
When $K(\boldsymbol{x}_i,\boldsymbol{x}_i)$ is constant for all $i = 1,\ldots,n$, which is satisfied, e.g. by the Gaussian kernel,

$$K(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2h^2}\right),$$

the above optimization problem is simplified as

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{i,j=1}^{n} \alpha_i \alpha_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ &\text{subject to } 0 \le \alpha_i \le C \quad \text{for } i = 1,\ldots,n, \\ &\sum_{i=1}^{n} \alpha_i = 1. \end{aligned}$$

A MATLAB code of support vector data description for the Gaussian kernel is provided in Fig. 38.4, and its behavior is illustrated in Fig. 38.5. This shows that



(a) C = 0.035 (b) C = 0.04

(c) C = 0.042 (d) C = 0.045

FIGURE 38.5

Examples of support vector data description for Gaussian kernel. Circled samples are regarded as outliers.

results of outlier detection depend on the choice of the trade-off parameter C (and the Gaussian bandwidth), and appropriately determining these tuning parameters is not straightforward in practice because of the unsupervised nature of outlier detection.

38.3 INLIER-BASED OUTLIER DETECTION

Since outliers tend to be highly diverse and their tendency may change over time, it is not easy to directly define outliers. On the other hand, inliers are often stable and thus indirectly defining outliers as samples that are different from inliers would be promising. In this section, such *inlier-based outlier detection* is discussed, under the assumption that, in addition to the test data set $\{\boldsymbol{x}_i\}_{i=1}^{n}$ from which outliers are detected, another data set $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ that only contains inliers is available.

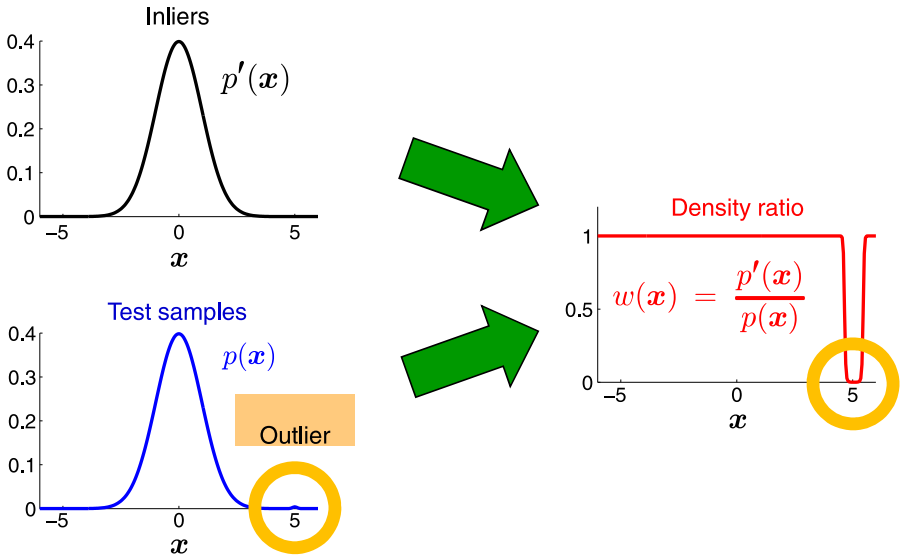


FIGURE 38.6

Inlier-based outlier detection by density ratio estimation. For inlier density $p'(\boldsymbol{x})$ and test sample density $p(\boldsymbol{x})$, the density ratio $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$ is close to one when $\boldsymbol{x}$ is an inlier and it is close to zero when $\boldsymbol{x}$ is an outlier.

A naive approach to inlier-based outlier detection is to estimate the probability density function $p'(\boldsymbol{x})$ of inliers $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$, and a test sample $\boldsymbol{x}_i$ having a low estimated probability density $\widehat{p}'(\boldsymbol{x}_i)$ is regarded as an outlier. However, this naive approach suffers the same drawback as the density estimation approach introduced in Section 38.1, i.e., estimating the probability density in low-density regions is difficult.

Here, let us consider the ratio of inlier density $p'(\boldsymbol{x})$ and test sample density $p(\boldsymbol{x})$,

$$w(\boldsymbol{x}) = \frac{p'(\boldsymbol{x})}{p(\boldsymbol{x})},$$

which is close to one when $\boldsymbol{x}$ is an inlier and it is close to zero when $\boldsymbol{x}$ is an outlier (Fig. 38.6). Thus, the *density ratio* would be a suitable inlier score.

The density ratio $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$ can be estimated by estimating $p(\boldsymbol{x})$ and $p'(\boldsymbol{x})$ separately from $\{\boldsymbol{x}_i\}_{i=1}^{n}$ and $\{\boldsymbol{x}'_{i'}\}_{i'=1}^{n'}$ and taking their ratio. However, division by an estimated density magnifies the estimation error and thus is not a reliable approach. Here, a method to directly estimate the density ratio $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$ *without* density estimation of $p(\boldsymbol{x})$ and $p'(\boldsymbol{x})$, called *KL density ratio estimation* [76, 104], is introduced.

```
n=50; x=randn(n,1); y=randn(n,1); y(n)=5;
x2=x.^2; xx=repmat(x2,1,n)+repmat(x2',n,1)-2*x*x';
y2=y.^2; yx=repmat(y2,1,n)+repmat(x2',n,1)-2*y*x';
m=5; u=mod(randperm(n),m)+1; v=mod(randperm(n),m)+1;
hhs=2*[1 5 10].^2;
for hk=1:length(hhs)
  hh=hhs(hk); k=exp(-xx/hh); r=exp(-yx/hh);
  for i=1:m
    a=KLIEP(k(u~=i,:),r(v~=i,:));
    g(hk,i)=mean(r(u==i,:)*a-mean(log(k(u==i,:)*a)));
end, end
[gh,ggh]=min(mean(g,2)); HH=hhs(ggh);
k=exp(-xx/HH); r=exp(-yx/HH); s=r*KLIEP(k,r);
figure(1); clf; hold on; plot(y,s,'rx');
```

```
function a=KLIEP(k,r)
a0=rand(size(k,2),1); b=mean(r)'; n=size(k,1);
for o=1:10000
  a=a0-0.001*(b-k'*(1./(k*a0))/n); %a=max(0,a);
  if norm(a-a0)<0.001, break, end
  a0=a;
end
```

FIGURE 38.7

MATLAB code of KL density ratio estimation for Gaussian kernel model with Gaussian bandwidth chosen by cross validation. The bottom function should be saved as "KLIEP.m."

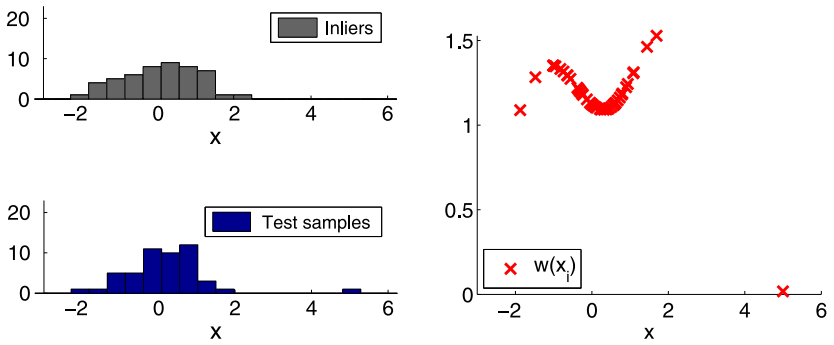
More specifically, let us use a *linear-in-parameter model* for approximating the density ratio $w(\boldsymbol{x}) = p'(\boldsymbol{x})/p(\boldsymbol{x})$:

$$w_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{j=1}^{b} \alpha_j \psi_j(\boldsymbol{x}) = \boldsymbol{\alpha}^\top \boldsymbol{\psi}(\boldsymbol{x}),$$

where the basis functions $\{\psi_j(\boldsymbol{x})\}_{j=1}^{b}$ are assumed to be non-negative. Since $w_{\boldsymbol{\alpha}}(\boldsymbol{x})p(\boldsymbol{x})$ can be regarded as a model of $p'(\boldsymbol{x})$, the parameter $\boldsymbol{\alpha}$ is learned so that $w_{\boldsymbol{\alpha}}(\boldsymbol{x})p(\boldsymbol{x})$ is as close to $p'(\boldsymbol{x})$ as possible.

For this model matching, let us employ the *generalized KL divergence* for non-negative functions f and g that are not necessarily integrated to 1:

$$\mathrm{gKL}(f\|g) = \int f(\boldsymbol{x}) \log \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} - \int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int g(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$



(a) Inliers and test samples (b) Estimated density ratio

FIGURE 38.8

Example of KL density ratio estimation for Gaussian kernel model.

When f and g are normalized, the above generalized KL divergence is reduced to the ordinary KL divergence since the second and third terms vanish. Under the generalized KL divergence, the parameter $\boldsymbol{\alpha}$ is learned to minimize

$$\mathrm{gKL}(p'\|w_{\boldsymbol{\alpha}}p) = \int p'(\boldsymbol{x})\log\frac{p'(\boldsymbol{x})}{w_{\boldsymbol{\alpha}}(\boldsymbol{x})p(\boldsymbol{x})}\mathrm{d}\boldsymbol{x} - 1 + \int w_{\boldsymbol{\alpha}}(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x}.$$

Approximating the expectations by the sample averages and ignoring irrelevant constants yield the following optimization problem:

$$\min_{\boldsymbol{\alpha}}\left[\frac{1}{n}\sum_{i=1}^{n} w_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) - \frac{1}{n'}\sum_{i'=1}^{n'}\log w_{\boldsymbol{\alpha}}(\boldsymbol{x}'_{i'})\right].$$

This is a convex optimization problem and thus the global optimal solution can be easily obtained, e.g. by a gradient method.

A critical drawback of the *local outlier factor* and *support vector data description* explained in the previous sections is that there is no objective model selection method. Thus, tuning parameters should be selected subjectively based on some prior knowledge. On the other hand, KL density ratio estimation allows objective model selection by *cross validation* (see Section 14.4 and Section 16.4.2) in terms of the KL divergence. This is practically a significant advantage in outlier detection.

A MATLAB code of KL density ratio estimation for the Gaussian kernel model,

$$w_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{j=1}^{n'}\alpha_j \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'_j\|^2}{2h^2}\right), \tag{38.7}$$

is provided in Fig. 38.7, where the Gaussian bandwidth h is chosen by cross validation. Its behavior is illustrated in Fig. 38.8, showing that a sample at $x = 5$,

which is isolated from other samples, takes the lowest density ratio value. Thus, it is regarded as the most plausible point to be an outlier.

A possible variation of KL density ratio estimation is to impose non-negativity $\boldsymbol{\alpha} \geq \mathbf{0}$ when the generalized KL divergence is minimized. This additional non-negativity constraint guarantees that a learned density ratio function $w_{\boldsymbol{\alpha}}(\boldsymbol{x})$ is non-negative. Another useful property brought by this non-negativity constraint is that the solution of $\boldsymbol{\alpha}$ tends to be *sparse*.

Implementing this non-negativity idea in MATLAB is straightforward just by rounding up negative parameter values to zero in each gradient step, as described in Fig. 38.7.