Probability Models for Open Set Recognition PAMI 2014

Sparse Representation-Based Open Set Recognition PAMI 2017

Nearest neighbors distance ratio open set classifier Machine Learning 2017

Classification Under Streaming Emerging New Classes: A Solution Using Completely-Random Trees

TKDE 2017

2017.3.22

Sparse Representation-Based Classification

Stack the training samples from the i_{th} class as columns of a large matrix: $Y_i \in \mathbb{R}^{M \times N_i}$

$$Y = [Y_1, Y_2, \dots, Y_K] \in \mathbb{R}^{M \times N} \qquad N = \sum_i N_i$$

If the Y_i are sufficiently expressive, a new input sample from the i_{th} class stacked as a vector $y_t \in \mathbb{R}^M$, will have a sparse representation

$$y_t = Yx \qquad x \in R^N$$

$$\hat{x} = \underset{x}{\operatorname{argmin}} ||x||_1 \qquad s.t. \quad ||y_t - Yx||_2 < \epsilon \quad ||x||_1 = \sum_i |x_i|$$

$$r_k = ||y_t - Y_k \hat{x}_k||_2, \qquad k = 1, \dots, K$$

$$k^* = \text{class of } y_t = \arg\min_k r_k$$

Algorithm 1. Sparse Representation-Based Classification

Input: $\mathbf{Y}$, $\mathcal{L}^Y$, ϵ, $\mathbf{y}_t$

$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \;\; \text{s.t.} \;\; \|\mathbf{y}_t - \mathbf{Y}\mathbf{x}\|_2 < \epsilon$

$r_k = \|\mathbf{y}_t - \mathbf{Y}_k \hat{\mathbf{x}}_k\|_2$ for $k = 1, \ldots, K$

$k^* = \arg\min_k r_k$

Output: k^*, $\mathbf{r} = [r_1, r_2, \ldots, r_K]$

$$\mathrm{SCI}(\mathbf{x}) = \frac{\frac{K \times \max_k \|\mathbf{x}_k\|_1}{\|\mathbf{x}\|_1} - 1}{K - 1} \in [0, 1]$$

single class

all class

Extreme Value Theory

an unknown distribution $F(z)$ n i.i.d. samples $\{Z_1, Z_2, \ldots, Z_n\}$

$$Z_m = \max_i Z_i \qquad i \in [1, n]$$

Fisher-Tippett-Gnedenko theorem

if there exists a pair of parameters (a_n, b_n) $\begin{matrix} a_n > 0 \\ b_n > 0 \end{matrix}$ $\Longrightarrow$ $\lim_{n\to\infty} P\left(\frac{Z_m - b_n}{a_n}\right) = E(z)$

$E(z)$ is a **non-degenerate distribution** that belongs to either Frechet, Weibull or Gumbel distribution. These distributions can be represented as a **Generalized Extreme Value distribution** (GEV) as follows

$$E(z;\mu,\sigma,\xi) = \exp^{-p(z)} \qquad p(z) = \left(1 + \xi\left(\frac{z-\mu}{\sigma}\right)\right)^{-1/\xi}$$

μ, σ and ξ are the **location**, **scaling** and **shape** parameters, respectively

- choose which distribution to use among the three based on prior knowledge
- segment the data into several parts and model the maximum in each part as a distribution using GEV

Generalized Pareto distribution (GPD), denoted as $G(z)$ (CDF) was proposed to estimate the **tail distribution of data samples.**

It was shown that given a **sufficiently large threshold** u, the probability of an observation exceeding u by z conditioned on u can be approximated by

$$\lim_{n\to\infty} P(Z > z + u | Z > u) = 1 - G(z)$$

$$G(z) = 1 - \left(1 + \xi \frac{z}{\sigma}\right)_+^{\frac{-1}{\xi}}, \quad z > 0 \qquad \sigma > 0, \xi \in \mathbb{R} \quad x_+ = \max(x, 0)$$

SPARSE REPRESENTATION-BASED OPEN-SET RECOGNITION

Open set Risk was defined as the cost of labeling the open set sample as known sample

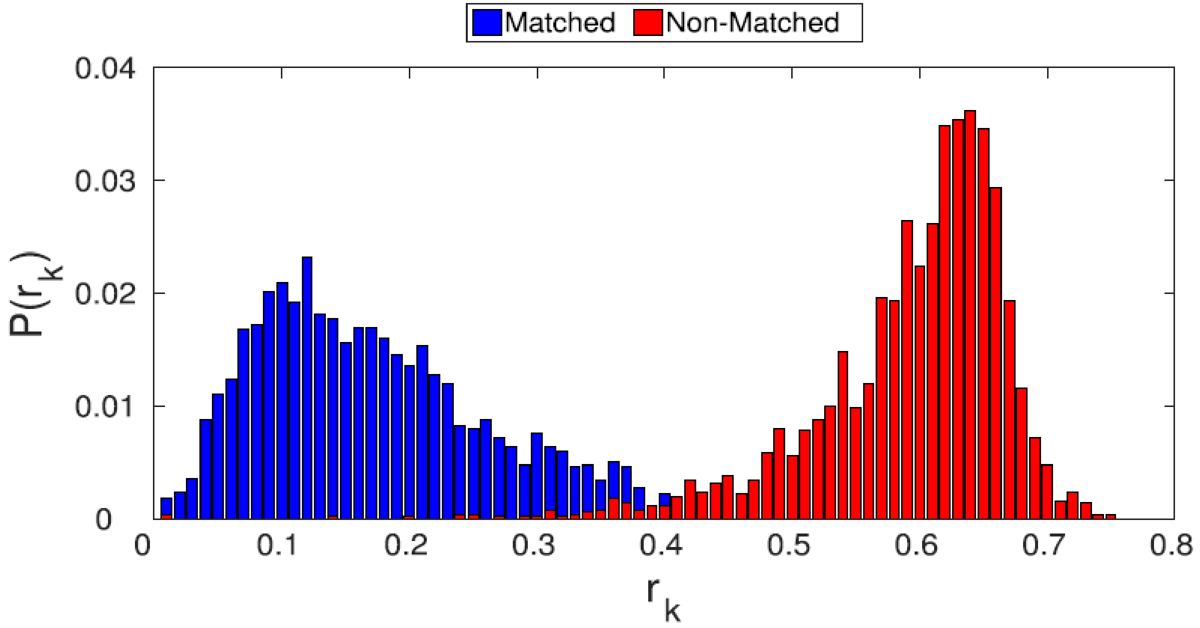
$$\arg\min_{f} C_o(f) + \lambda_r C_\epsilon(f)$$

$C_o(f)$:   open set risk

$C_\epsilon(f)$:   empirical risk for classification

Matched reconstruction errors here mean that the errors correspond to the sparse coefficients of digit 9

Non-matched reconstruction errors mean that the errors are generated by the sparse coefficients of all other digits



If one can fit a probability model $P(r_k)$ to **describe the distribution of the reconstruction errors** of the matched class, then one can reformulate the open-set recognition problem as **a hypothesis testing** for novelty detection problem as

$$\mathcal{H}_0:\ P(r_k) \le \delta$$
$$\mathcal{H}_1:\ P(r_k) > \delta$$

$$\delta \in [0,1]$$

Use the GPD to model the tail of the matched distribution

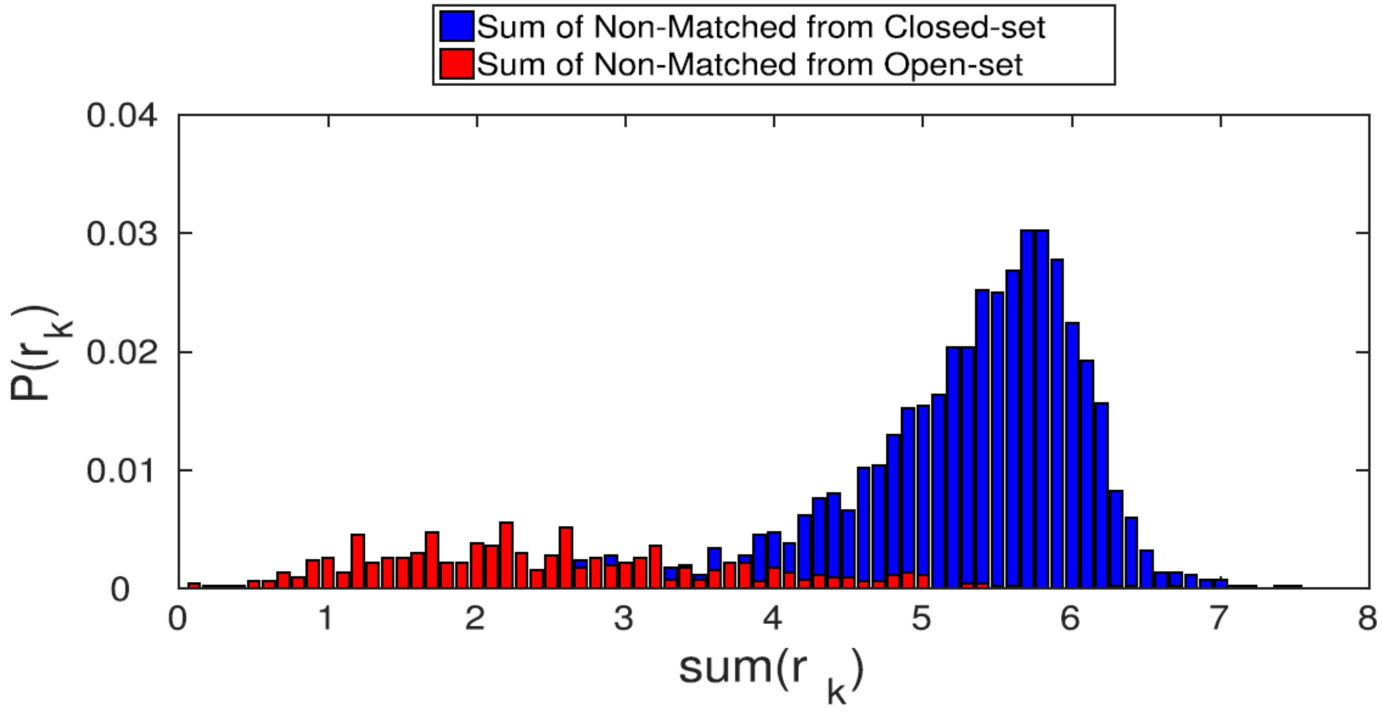
$$\mathcal{H}_0:\ G(r_k) \le \delta_g$$
$$\mathcal{H}_1:\ G(r_k) > \delta_g$$

$G(r_k)$ is the learned GPD distribution for fitting the right tail of r_k and δ_g is the rejection threshold.

As we are only interested in the right tail of the **matched distribution** and the left tail of the sum of **non-matched distribution**, we apply an inverse procedure to the random variable Z as

$$Z_I = -Z$$

So the right tail of Z_I is the left tail of Z



Training

Algorithm 2. Pseudocode for SROSR Training

Input: $\mathbf{Y}$, ρ, ϵ, L, $\mathcal{L}^Y$

Initialization

for $i = 1 : K$ **do**

for $j = 1 : L$ **do**

$\tilde{\mathbf{Y}}_i$ = randomly ordered $\mathbf{Y}_i \in \mathbb{R}^{M \times N_i}$

$N_{tr} = N_i \times 0.8$

$\mathbf{Y}_i^{tr} = \tilde{\mathbf{Y}}_i(:, 1 : N_{tr})$

$\mathcal{L}_i^{tr} = \text{Labels of } \mathbf{Y}_i^{tr}$

$\mathbf{Y}_i^{te} = \tilde{\mathbf{Y}}_i(:, N_{tr} + 1 : \text{end})$

$\mathcal{L}_i^{te} = \text{Labels of } \mathbf{Y}_i^{te}$

$\mathbf{r}_i(j, :) \leftarrow \text{SRC}\left(\mathbf{Y}^{tr}, \mathbf{Y}^{te}, \mathcal{L}^{tr}, \mathcal{L}^{te}, \epsilon\right)$

end for

$\mathbf{R}_i^m = [\mathbf{r}_i(1, i), \ldots, \mathbf{r}_i(L, i)]$

$\mathbf{R}_i^{nm} = [\sum_{p:p \neq i} \mathbf{r}_i(1, p), \ldots, \sum_{p:p \neq i} \mathbf{r}_i(L, p)]$

$\boldsymbol{\sigma}_m(i), \xi_m(i) \leftarrow \text{GPDfit}(\mathbf{R}^m, \rho)$

$\boldsymbol{\sigma}_{nm}(i), \xi_{nm}(i) \leftarrow \text{GPDfit}(-\mathbf{R}^{nm}, \rho)$

end for

Output: $\boldsymbol{\sigma}_m, \xi_m, \boldsymbol{\sigma}_{nm}, \xi_{nm}$

Testing

As the two raw reconstruction errors are all normalized into probabilities by their corresponding GPDs, we can add the two probability scores together with appropriate weights to obtain the final score.

$$w = \frac{1}{3}(1 - \text{Openness})$$

$$\text{Openness} = 1 - \sqrt{\frac{2 \times N_{TA}}{N_{TG} + N_{TE}}}$$

N_{TA}, N_{TG}, and N_{TE} are the number of **training classes**, the number of **target classes** to be identified, and the number of **testing classes**, respectively

Algorithm 3. Pseudocode for SROSR Testing

Input: $\mathbf{y}_t, \mathbf{Y}, \boldsymbol{\sigma}_m, \xi_m, \boldsymbol{\sigma}_{nm}, \xi_{nm}, \delta_t, w, \epsilon$

1: $\mathbf{r} \leftarrow \text{SRC}\ (\mathbf{Y}, \mathbf{y}_t, \mathcal{L}^Y, \epsilon)$

3: $k^* = \arg\min_i r_i$

4: $r_m = r_{k^*}, r_{nm} = \sum_{i=1, i \neq k^*}^{K} r_i$

5: $S_m = G(r_m; \boldsymbol{\sigma}_m(k^*), \xi_m(k^*))$,
$S_{nm} = G(r_{nm}; \boldsymbol{\sigma}_{nm}(k^*), \xi_{nm}(k^*))$

6: $S = S_m + w \ldots S_{nm}$

if $S > \delta_t$ **then**
  Class of $\mathbf{y}_t = \mathcal{O}$
else
  Class of $\mathbf{y}_t = k^*$
end if

Output: k^* or $\mathcal{O}$

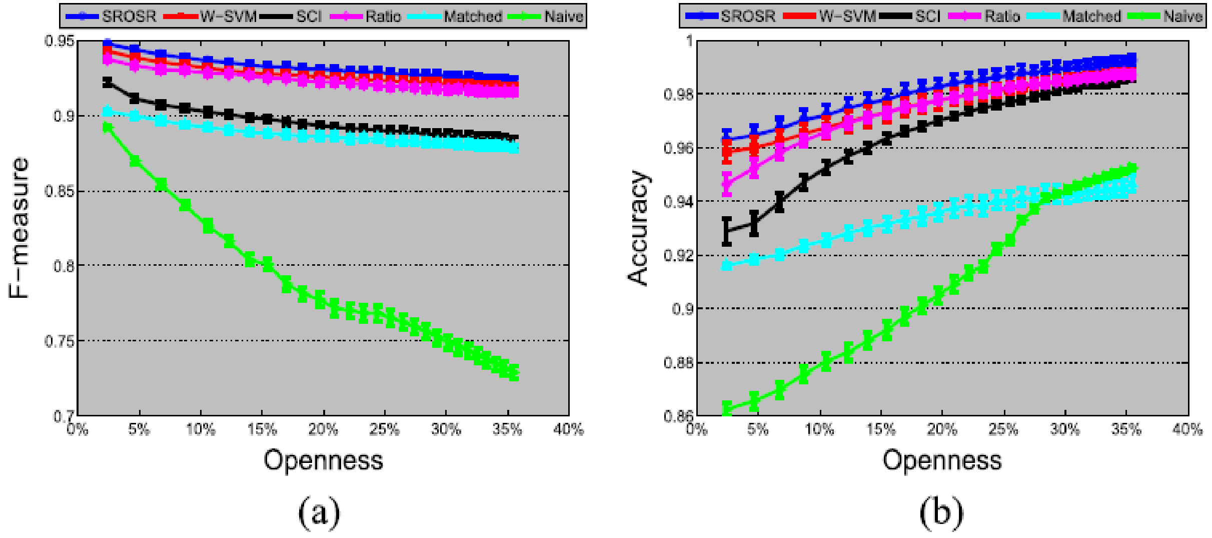


Fig. 4. Results on the extended Yale B dataset. (a) Openness versus F-measure results. (b) Openness versus accuracy results.

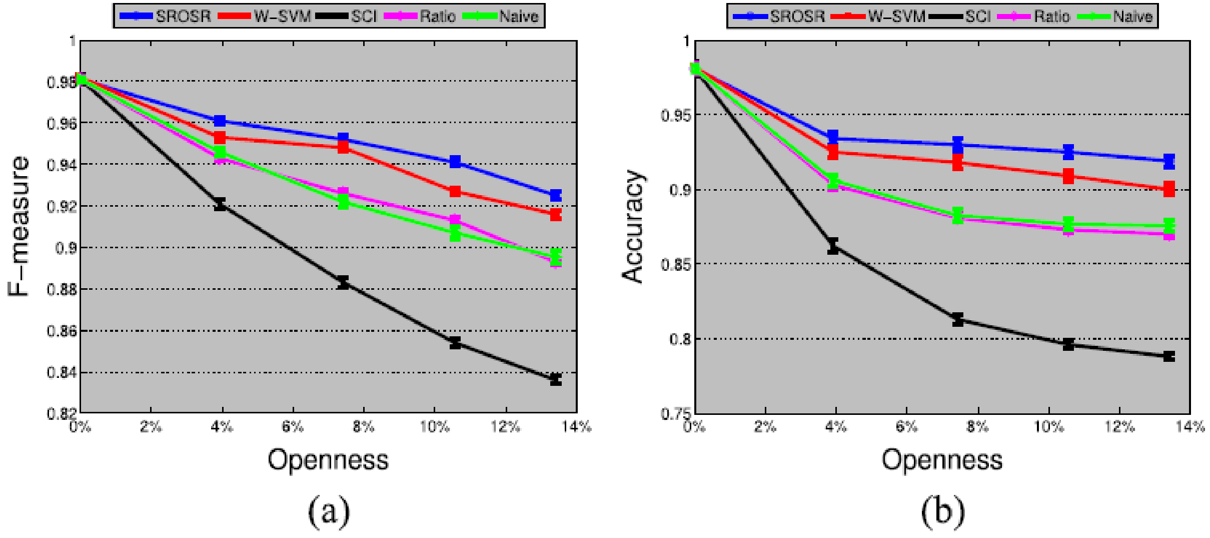


Fig. 5. Results on the MNIST dataset. (a) Openness versus F-measure results. (b) Openness versus accuracy results.

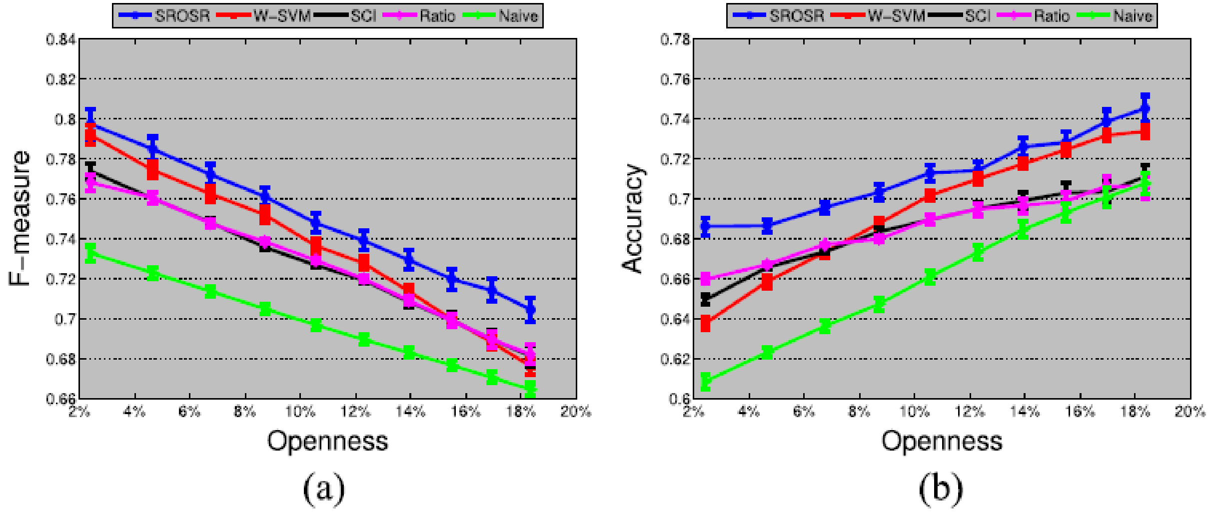


Fig. 6. Results on the UIUC attribute dataset. (a) Openness versus F-measure results. (b) Openness versus accuracy results.

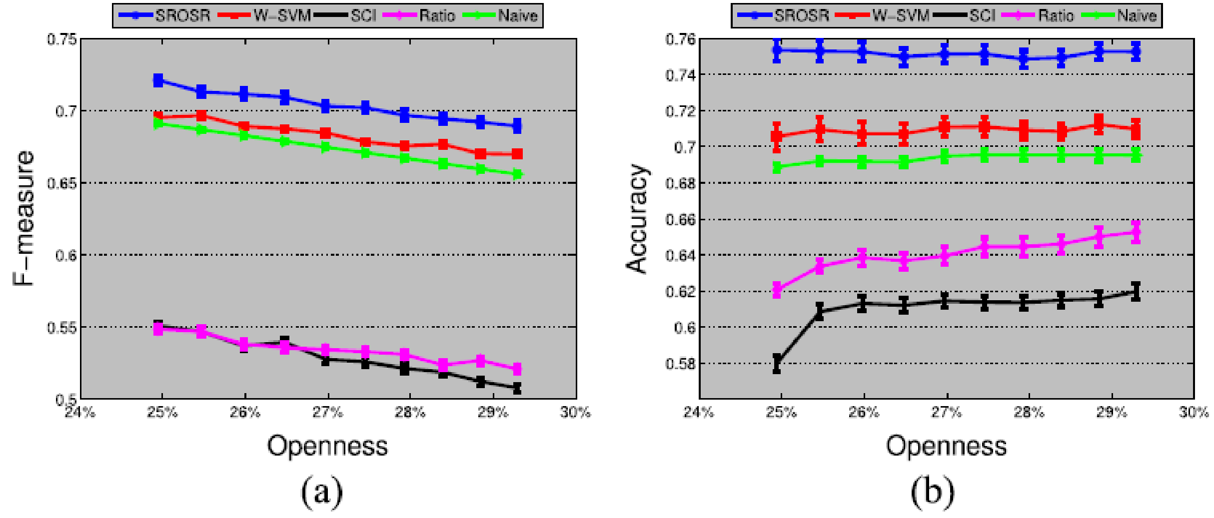


Fig. 7. Results on the Caltech 256 dataset. (a) Openness versus F-measure results. (b) Openness versus accuracy results.

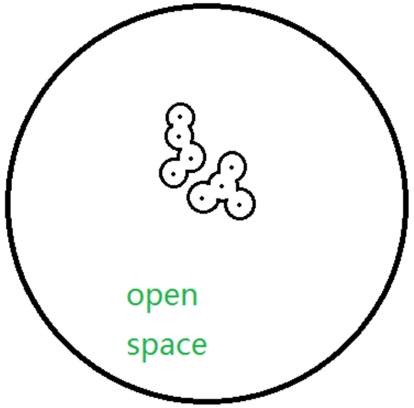
Probability Models for Open Set Recognition

PAMI 2014

The idea of a CAP model is to ensure that the recognition function is decreasing away from the training data

O be the “open space,” and S_o be a ball of radius r_o that includes all of the known **positive training examples** $x \in K$ as well as the open space O. The probabilistic Open Space Risk $R_o(f)$ for a class y can be defined as

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f_y(x)dx}{\int_{S_o} f_y(x)dx}$$



The definition of open space

$$O = S_o - \bigcup_{i \in N} B_r(x_i)$$

$B_r(x_i)$ is a closed ball of radius r centered around training sample x_i

Abating bound $A(r): \mathbb{R} \to \mathbb{R}$ is a non-negative finite square integrable continuous decreasing function.

$$\lim_{r\to\infty} A(r) = 0$$

$$K(x, x_i) = < \Phi(x), \Phi(x_i) > \quad x_i \in \mathcal{K} \quad x \in X$$

We call kernel K abating if there exists an abating bound A such that

$$\forall x, x_i: \quad 0 < K(x, x_i) \leq A||x - x_i||$$

RBF (Gaussian) kernels

Abating Probabilistic Point Model monotonically decreasing probability distribution $p_f(s; y)$

Consider fusing the abating models, for any example $x \in X$ we define the model

$$M(x) = p_f(F\big(K(x, x_1) \ldots K(x, x_m)\big); y)$$

F is the fusion operator canonical sum or canonical product rule

positive definite kernels are closed under canonical sums or products

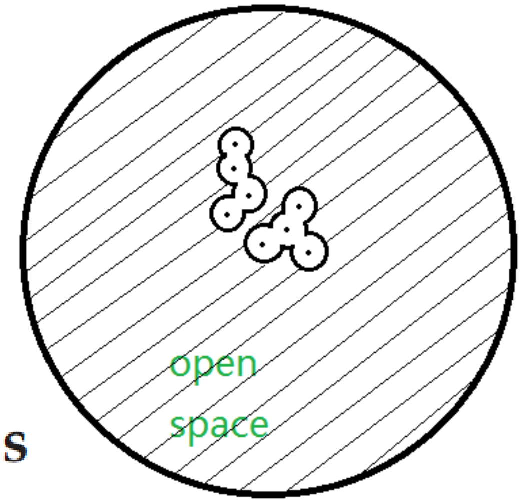
Fused Abating Property $F(K(x, x_1) \ldots K(x, x_m)) \leq A_{x'}(||x' - x||)$ $\lim_{r \to \infty} A(r) = 0$

The model can have non-zero probability over all of $\mathbb{R}^n$

compact abating probability Model M_τ given finite τ and $\forall x \in X$

$$\min_{x_i \in \mathcal{K}} \left| |x - x_i| \right| > \tau \Rightarrow M_\tau(x) = 0$$

Theorem 1 (Open Space Risk of CAP models). *Let $M_{\tau,y}(x)$ be a probabilistic recognition function that uses a CAP model over a known training set for class y, where $\exists x_i \in \mathcal{K} \mid M_{\tau,y}(x_i) > 0$. Let open space risk be $R_{\mathcal{O}}(f)$ and open space be $\mathcal{O}$, defined as in Eqs. (1) and (2) respectively. If r in Eq. (2) satisfies $\boxed{r > \tau}$, then $\boxed{R_{\mathcal{O}}(M_{\tau,y}) = 0}$, i.e., when the CAP distance threshold is smaller than the open space radius, the CAP model has zero open space risk.*



Corollary 1 (Thresholding CAP model probability manages Open Space Risk). *For any CAP model, considering only points with sufficiently high probability will reduce open space risk. In particular, consider a canonical sum kernel-based CAP model with a probability threshold $\boxed{0 \leq \delta_\tau \leq 1}$ such that for the set of points $x_i \in \mathcal{K}$ and coefficients $\vartheta_i > 0$, $p_f(\boxed{\sum_i \vartheta_i K(x, x_i)}; y) \geq \delta_\tau$. Increasing δ_τ decreases open space risk, and there exists a δ_τ^* such that any greater threshold produces zero open space risk.*

A simple CAP example: Nearest Neighbor + CAP

d_x: the distance to the nearest neighbor of x

$$d_x > \tau \Rightarrow p_a(x) = 0 \qquad p_a(x) = \frac{|\tau - d_x|}{\tau}$$

this results in a thresholded nearest neighbor algorithm that can reject an input as unknown

Theorem 2 (RBF One-Class SVM yields CAP model). *Let $x_i \in \boxed{\mathcal{K}}, i = 1 \ldots m$ be the training data for class y. Let O-SVM be a one-class SVM with a square integrable monotonically decreasing RBF kernel K defined over the training data, with associated Lagrangian multipliers $\alpha_i > 0$ [28], then $\sum_i \alpha_i y_i K(x, x_i)$ yields a CAP model.*

Weibull-calibrated SVM.

Unfortunately, the decision score of a binary SVM is **not a canonical sum**, however, still be useful as improved probabilities will generally result in **tighter bounds around the class of interest**.

$$y = \operatorname{sgn}\left(\frac{1}{n_+} \sum_{\{i:y_i=+1\}} \underbrace{\langle \Phi(x), \Phi(x_i) \rangle}_{k(x,x_i)} - \frac{1}{n_-} \sum_{\{i:y_i=-1\}} \underbrace{\langle \Phi(x), \Phi(x_i) \rangle}_{k(x,x_i)} + b \right)$$

$$b = \tfrac{1}{2}\left(\tfrac{1}{n_-^2} \textstyle\sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \tfrac{1}{n_+^2} \textstyle\sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j)\right)$$

$$p_+(x) := \frac{1}{n_+} \sum_{\{i:y_i=+1\}} k(x, x_i), \qquad p_-(x) := \frac{1}{n_-} \sum_{\{i:y_i=-1\}} k(x, x_i),$$

View the SVM as applying a decision rule on which is more similar

Working with only the positive or negative data, we can get **nicely bounded results** from a binary SVM that can be used in conjunction with the **one-class probabilities**.

Binary RBF SVM Incorporating a CAP Model

We use the one-class SVM CAP model as a **conditioner**: if the one-class SVM predicts $P_o(y|x) > \delta_\tau$, even with a very low threshold δ_τ, that a given input x is a member of class y, then we will consider the binary classifier's estimates of $P(y|x)$.

We seek to model the positive and negative scores separately.

$y \in \mathcal{Y}$ $\quad P^+(y|x) \quad P^-(\mathcal{Y}\backslash y|x) \quad P^+(y|x) = 1 - P^-(\mathcal{Y}\backslash y|x)$

Thus to minimize our open space risk, we only consider P^+ and P^+ when $P_o(y|x) > \delta_\tau$

Grounded Probability Estimation

The extreme values of a **score distribution** produced by **any recognition algorithm** can always be modeled by an **EVT distribution**, which is a reverse Weibull if the data are bounded from above, and a Weibull if bounded from below

A **reverse Weibull** is justified for the largest scores from the **negative examples** because they are bounded from **above**

A **Weibull** is the expected distribution for the smallest scores from the **positive examples** because they are bounded from **below**.

$\mathcal{K} = \mathcal{K}^+ \cup \mathcal{K}^-$

$s_i = f(x_i)$: the SVM decision score for x_i

$$s_j \in S^+ \;\; if \; x_j \in \mathcal{K}^+ \qquad \psi$$
$$s_j \in S^- \;\; if \; x_j \in \mathcal{K}^- \qquad \eta$$

location ν, scale λ, and shape κ.

Applying maximum likelihood estimation to estimate $\nu_\eta, \lambda_\eta, \kappa_\eta$ that best fit η and the $\nu_\psi, \lambda_\psi, \kappa_\psi$ that best fit ψ

Given a test sample x, we have two independent estimates for $P(y|f(x))$:

$$P_\eta(y|f(x)) = 1 - e^{-(\frac{-f(x)-\nu_\eta}{\lambda_\eta})^{\kappa_\eta}}$$

P_ψ based on the reverse Weibull CDF derived from the non-match data, which is equivalent to rejecting the Weibull fitting on the non-match data:

$$P_\psi(y|f(x)) = 1 - e^{-\left(\frac{-f(x)-v_\psi}{\lambda_\psi}\right)^{\kappa_\psi}}$$

The W-SVM Algorithm

$P_\eta \times P_\psi$ the probability that the input is from the positive class AND NOT from any of the known negative classes

$P_\eta + P_\psi$ either a positive OR NOT a known negative

$$y^* = \underset{y \in \mathcal{Y}}{\arg\max}\, P_{\eta,y}(x) \times P_{\psi,y}(x) \times l_y \qquad \text{subject to} \qquad P_{\eta,y^*}(x) \times P_{\psi,y^*}(x) \geq \delta_R$$

indicator variable $l_y = \begin{cases} 1 & \text{if } P_O(y|x) > \delta_\tau \\ 0 & \text{otherwise} \end{cases}$

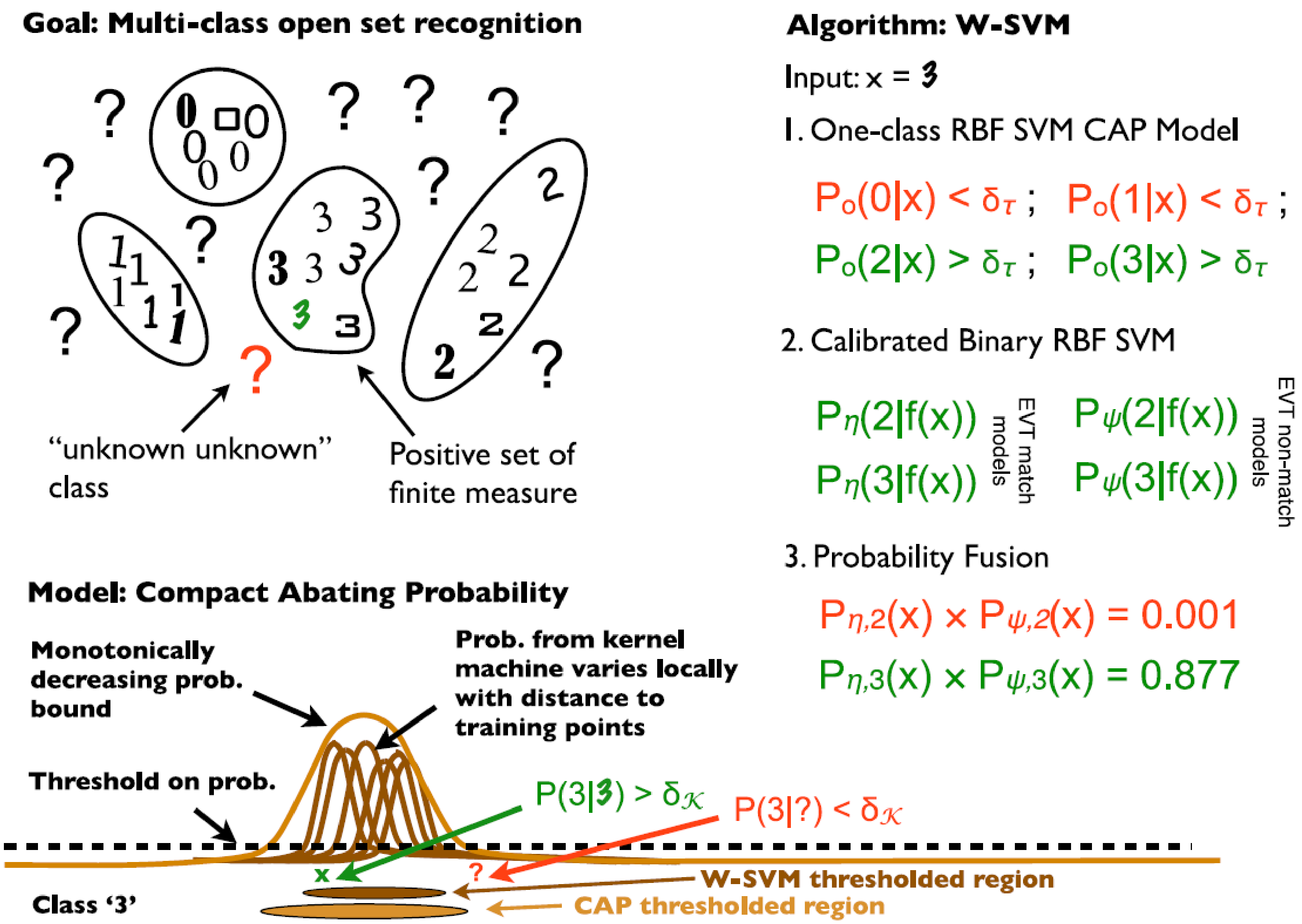


Fig. 1. Open set recognition must address both the known and unknown classes

Nearest neighbors distance ratio open set classifier

Machine Learning 2017

open space : all the region of the feature space outside the support of the training samples

positively labeled open space (PLOS): the region of the feature space in which a sample would be classified as positive

KLOS : all the region of the feature space, outside the support of the training samples, in which a sample would be classified as belonging to one of the known classes.

open space risk: the ratio of the volume of the PLOS to the volume of a sphere containing both the PLOS and the training samples

$$\arg\min_{f \in \mathcal{H}} \{R_O(f) + \lambda_r R_\varepsilon(f)\}$$

Two inherently multiclass open-set extensions for the NN classifier

Class Verification (CV)

Nearest Neighbor Distance Ratio

Class Verification

Based on the agreement of the labels of the two nearest neighbors with respect to a test sample. The training phase is the same of the NN, i.e., it only requires the storage of the training samples.

Nearest Neighbor Distance Ratio

The nearest neighbor t of the test sample s and then obtains the nearest neighbor u of s

$$\theta(u) \neq \theta(t) \qquad \theta(x) \in \mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$$

$$R = d(s,t)/d(s,u)$$

$d(x, x')$ is the Euclidean distance between samples x and x' in the feature space

$$\theta(s) = \begin{cases} \theta(t) & if \quad R \leq T \\ \ell_0 & if \quad R > T \end{cases}$$

ℓ_0 is the unknown label

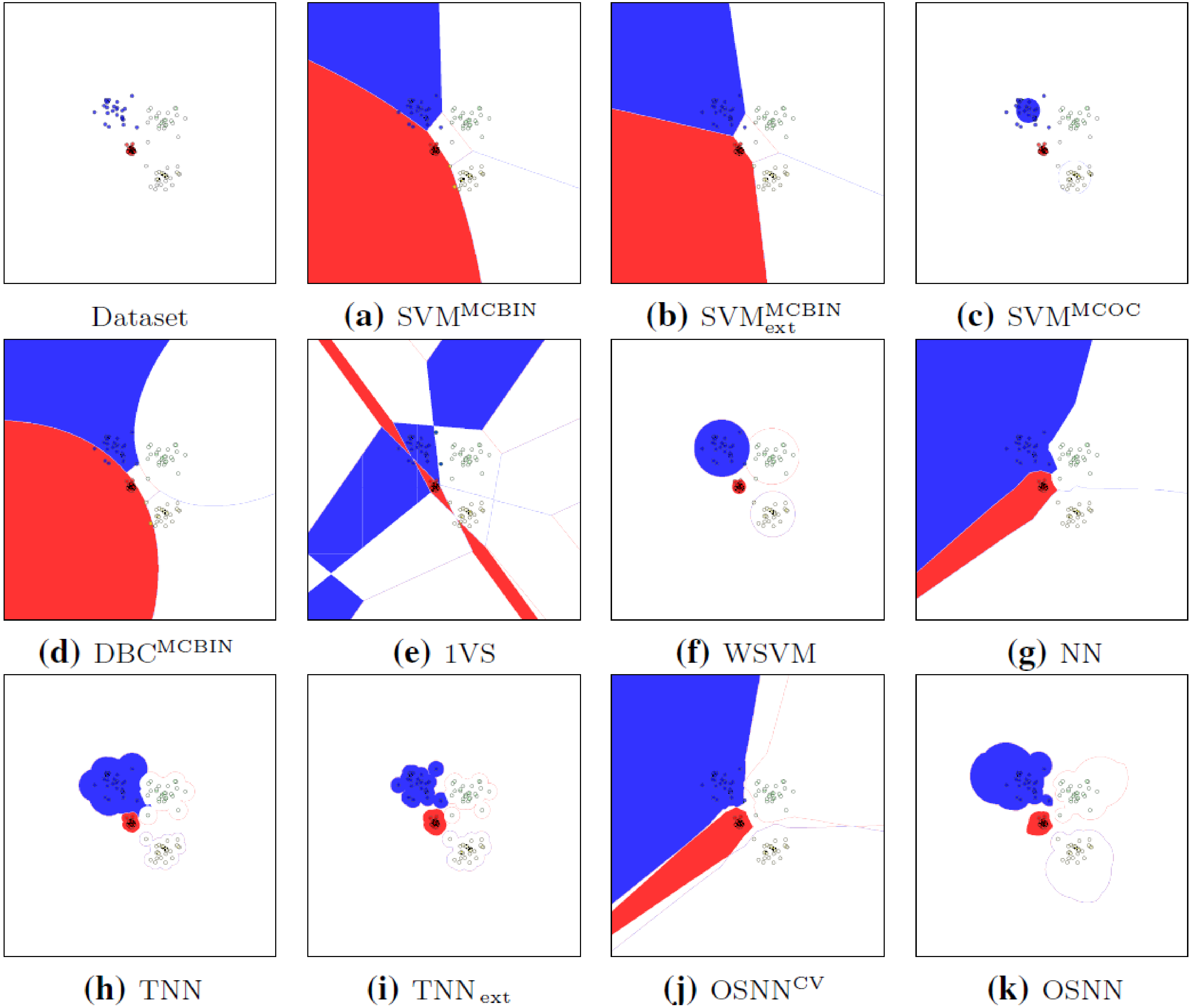
Parameter optimization

fitting set F contains half for the instances of the “known” classes

validation set V contains the other half of the instances of the “known” classes, and all instances of the “unknown” classes

normalized accuracy (NA) accuracy on known samples (AKS) accuracy on unknown samples (AUS)

$$\mathrm{NA} = \lambda_r \mathrm{AKS} + (1 - \lambda_r)\mathrm{AUS}$$



Dataset
(a) SVM$^{\mathrm{MCBIN}}$
(b) SVM$^{\mathrm{MCBIN}}_{\mathrm{ext}}$
(c) SVM$^{\mathrm{MCOC}}$
(d) DBC$^{\mathrm{MCBIN}}$
(e) 1VS
(f) WSVM
(g) NN
(h) TNN
(i) TNN$_{\mathrm{ext}}$
(j) OSNN$^{\mathrm{CV}}$
(k) OSNN

Isolation Forest

ICDM 2017

Classification Under Streaming Emerging New Classes: A Solution Using Completely-Random Trees

TKDE 2017

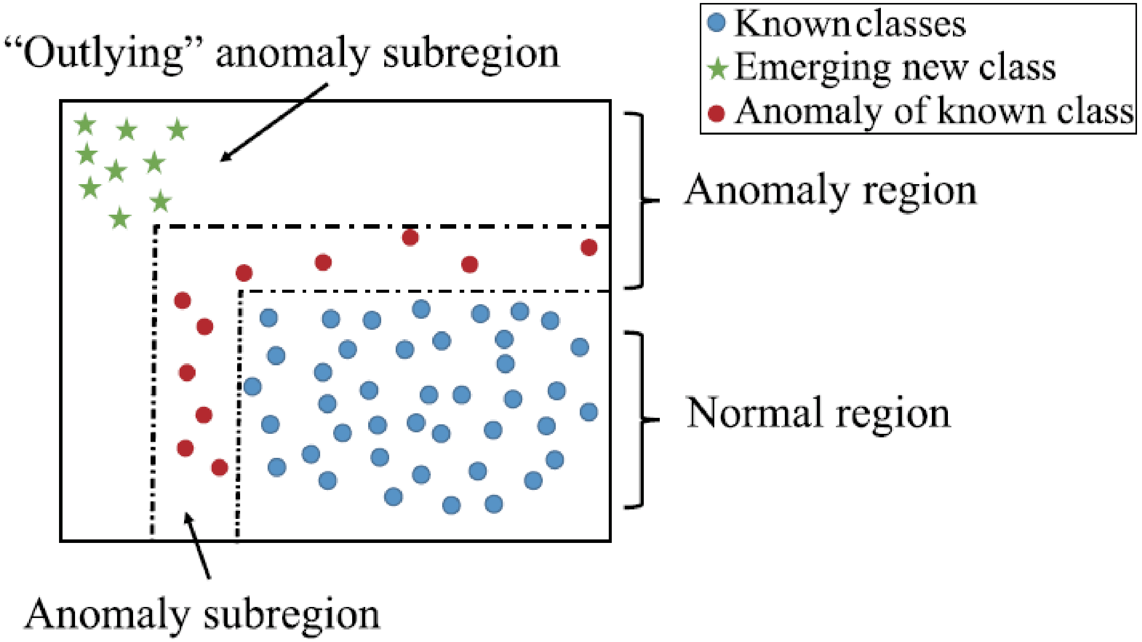
SENC problem

1 detecting emerging new classes,
3 updating models to enable classifica ... ore emerging new classes.

SENC

$D = \{(x_i, y_i)\}_{i=1}^{L}$

$S = \{(x_t', y_t')\}_{i=1}^{L}$ $x_i \in R^d$ y



Anomalies of Known Classes

$\mathcal{O} = \{x_1, \ldots, x_n\}$: training instances in an anomaly region A

center of $\mathcal{O}$ $c = \frac{1}{n}\sum_{x \in \mathcal{O}} x$

farthest instance from c $e \in \mathcal{O}$

Ball B centered at c with radius $r = dist(c, e)$ is an anomaly subregion

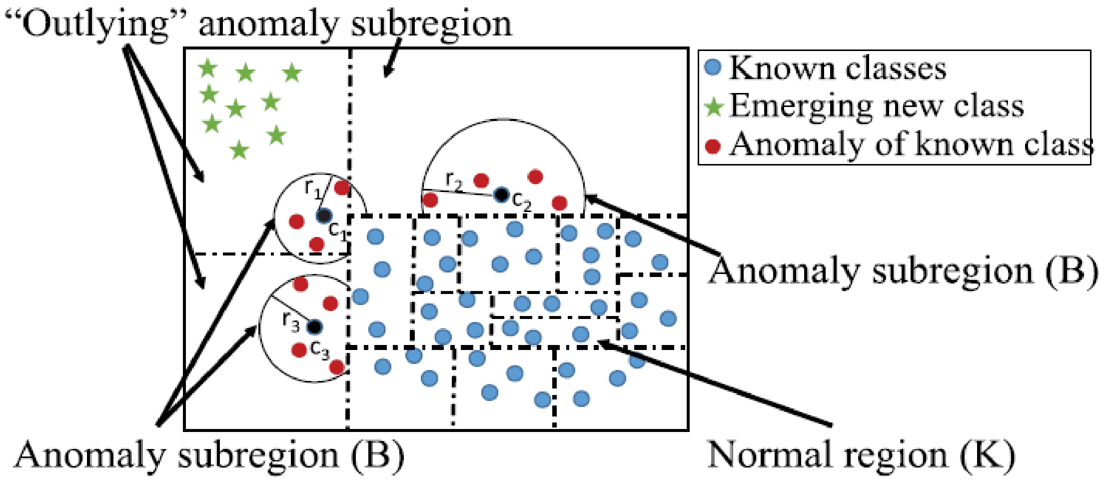
SENCForest: An Overview

1. Train a Detector for Emerging New Classes

1) Build an iForest

2) Determine the path length threshold $\hat{\tau}$, and achieve anomaly region (A) in each tree.

3) Within each region A, construct ball B which covers all training instances which fall into this region.



2. Using the Known Class Information to Build a Classifier from a Detector

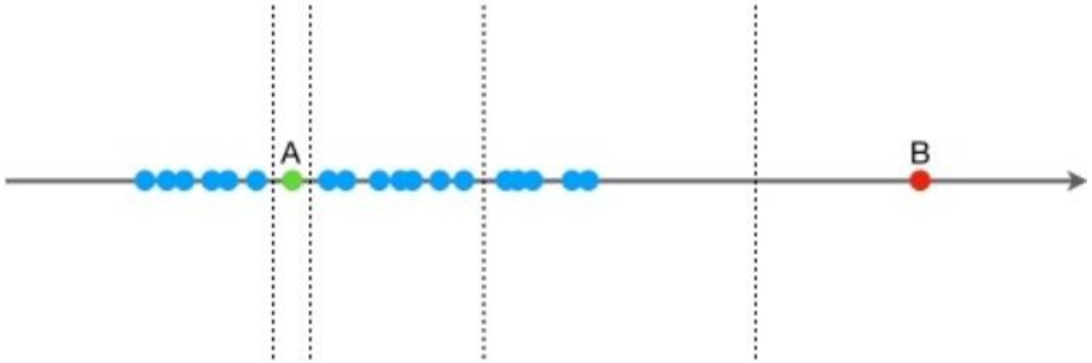
class distributions based on known class labels are recorded **in each K or B region**. **Each region with class distribution** acts as a **classifier** that outputs the majority class as the classification result for a test instance

3. Deployment in a Data Stream

An instance in the data stream is given a class prediction by SENCForest if it falls into K or B region; otherwise, it is identified as an instance from an emerging new class and placed in a buffer of size s.

4. Model Update.

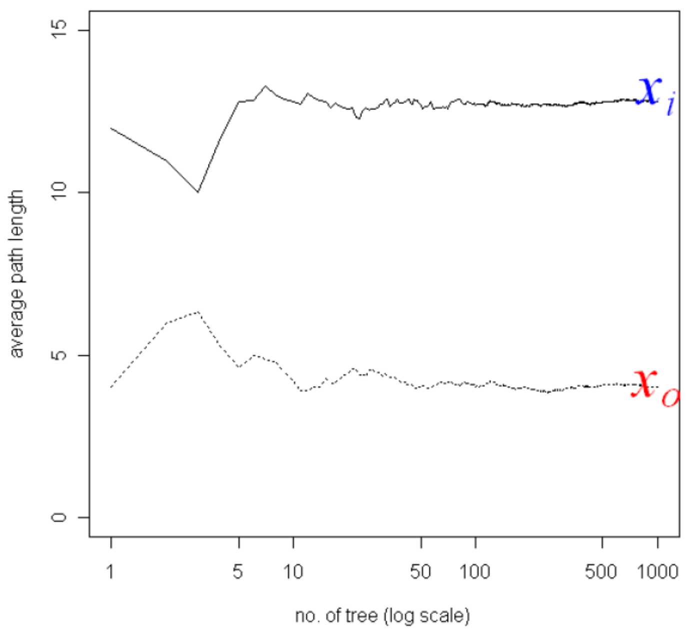
SENCForest: Training Process

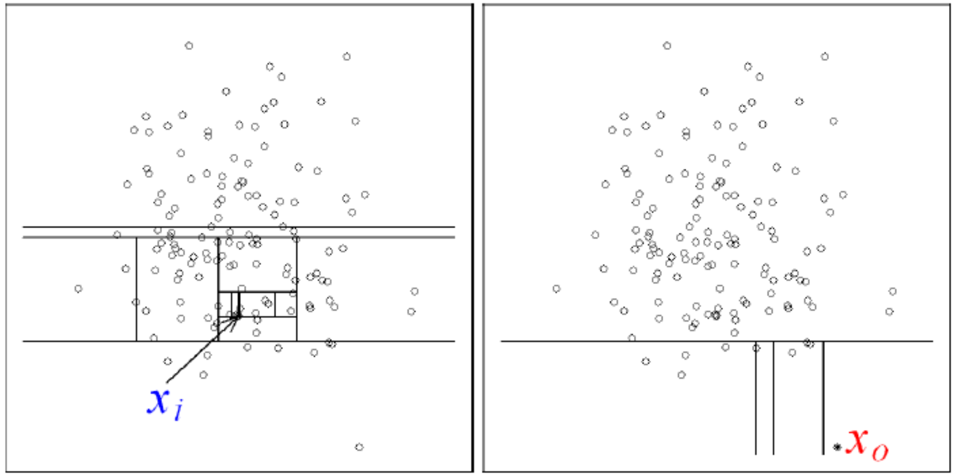


Algorithm 2. *SENCTree*

Input: X - input data, $MinSize$-minimum internal node size
Output: *SENCTree*

if $|X| < MinSize$ **then**
return LeafNode$\{|X|, F[\cdot], c, r\}$, as defined in Section 5.2.
else
  let Q be a list of attributes in X
  randomly select an attribute $q \in Q$
  randomly select a split point p from max and min values of attribute q in X
  $X_L \leftarrow filter(X, q \leq p)$
  $X_R \leftarrow filter(X, q > p)$
  return inNode{Left $\leftarrow SENCTree(X_L)$,
    Right $\leftarrow SENCTree(X_R)$,
    SplittAtt $\leftarrow q$,
    SplittValue $\leftarrow p$ },
end if





(a) Isolating x_i (b) Isolating x_o

Algorithm 1. Build *SENCForest*

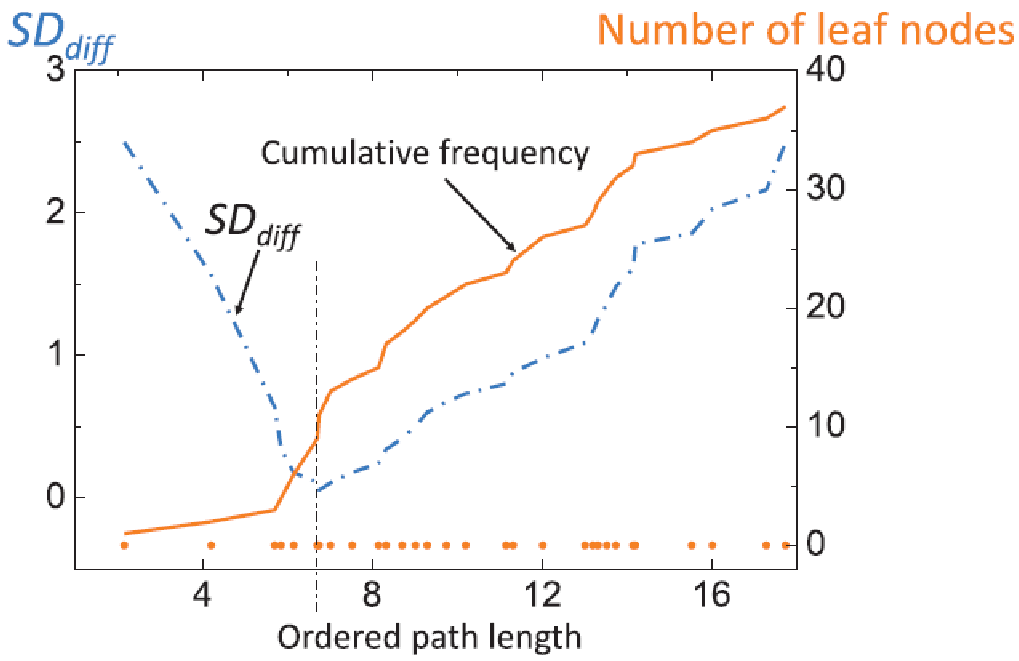
Input: D - input data, z - number of trees, ψ - subsample size.
Output: *SENCForest*

```
initialize: SENCForest ← {}
for i = 1, ..., z do
    X_i ← sample(D, ψ)
        SENCForest ← SENCForest ∪ SENCTree(X_i)
end for
return SENCForest
```

Determine the Path Length Threshold

$$\hat{\tau} = \arg\min_{\tau} |\sigma(L^r) - \sigma(L^l)|$$

$$SD_{diff} = |\sigma(L^r) - \sigma(L^l)|$$



Construct “Outlying” Anomaly Subregions

Ball B is constructed using all training instances in every region A of a tree

Produce a Classifier from a Detector

record class distribution $F[j]$ in each K or B region using the training subsample

$F[j]$ the number of class j instances in a region.

Deployment in Data Stream

$$y \in \{b_1, \dots, b_m, NewClass\}$$

$$\underset{j \in \{b_1, \dots, b_m\}}{\arg\max} \; F[j]$$

Algorithm 3. Deploying *SENCForest* in Data Stream

Input: *SENCForest*, $\mathcal{B}$ - buffer of size s

Output: y - class label for each x in a data stream

while not end of data stream **do**
  for each x **do**
    $y \leftarrow SENCForest(x)$
    if $y = NewClass$ **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{x\}$
      if $|\mathcal{B}| \geq s$ **then**
        Update (*SENCForest*, $\mathcal{B}$)
        $\mathcal{B} \leftarrow$ NULL
        $m \leftarrow m + 1$
      end if
    end if
    Output $y \in \{b_1, \dots, b_m, NewClass\}$.
  end for
end while

Model Update 略

Prediction Using Multiple SENCForests

$$p_i = \frac{\text{Number of } SENCTrees \text{ predicting } y_i}{\text{Total number of } SENCTrees}$$

Algorithm 5. Final Prediction from E $SENCForests$

Input: x-an instance in the data stream
Output: $y_\imath$ - class label for x

for $i = 1, \ldots, E$ **do**
  $\langle y_i, p_i \rangle \leftarrow SENCForest_i(x)$
end for
if $\forall_i \quad y_i = NewClass$ **then**
  $y_\imath = NewClass$
else
  $L \leftarrow \{i \in \{1, \ldots, E\} \mid y_i \neq NewClass\}$
  $\imath \leftarrow \arg\max_{i \in L} \; p_i$
end if
Output $y_\imath$

Semi-Supervised Learning with Graphs PhD thesis 2005

Active learning via transductive experimental design ICML 2006

Manifold Regularized Experimental Design for Active Learning TIP 2017

Beyond the Point Cloud: from Transductive to Semi-supervised Learning ICML 2005

Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples

JMLR 2006