Supplemental Material: Probability Models for Open Set Recognition

A Results for Multi-class Open Set Recognition Accuracy

The main article presents F-measure plots, but reviewers may be interested in accuracy plots for comparison. Accuracy is a common choice for evaluating binary decision classifiers. However, it is not always the best choice because it tends to underemphasize the distinction between correct positive and negative classification. Precisely defined, accuracy refers to the correctly classified samples (true positives TP and true negatives TN) out of all of the classification decisions (TP, TN, false positives FP, and false negatives FN).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

For open set accuracy, we consider a correct response to be either the correct class label or "rejection" if the test sample is from a class not used in training. Fig. 1 depicts results for OLETTER, while Fig. 2 depicts results for OMNIST. Comparing these plots with Figs. 3 and 4 in the article, the same trends are evident[1].

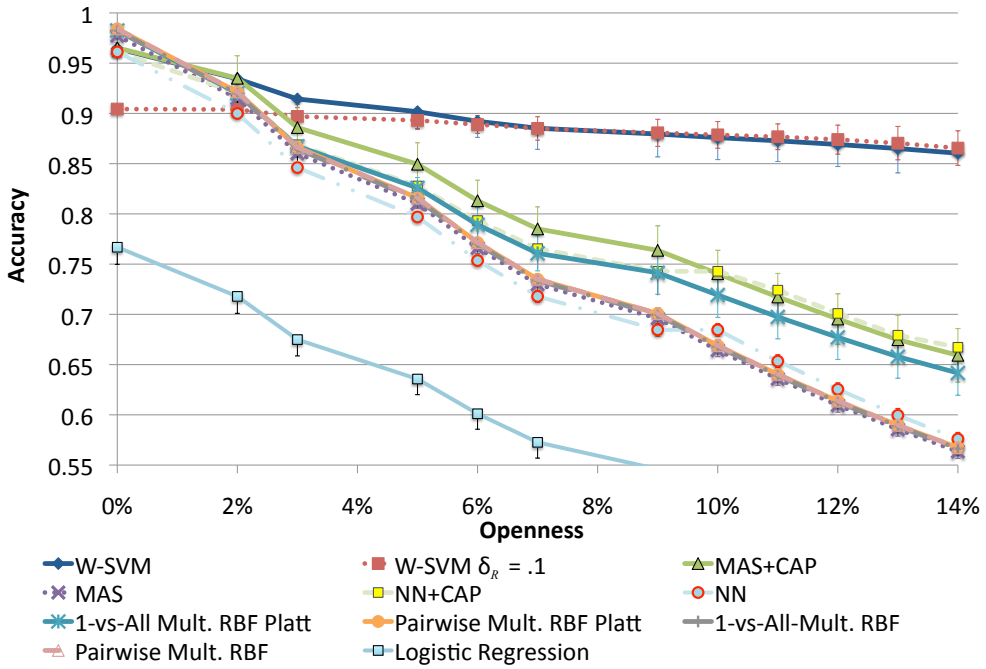

Figure 1. Accuracy for open set multi-class recognition on OLETTER. Common multi-class SVMs and pairwise SVM with Platt probabilities and a rejection option degrade quickly (1-vs-All Multi. RBF, Pairwise Mult. RBF, and MAS are all comparable and visually overlap). The MAS algorithm with a CAP model, Nearest Neighbor algorithm with a CAP model, and the 1-vs-All SVM with Platt probabilities and a rejection option do a bit better, but still degrade much more than the W-SVM. Error bars reflect standard deviation.

[1]For a detailed discussion of accuracy in the context of detection, please see Sec. 5 of our article "Towards Open Set Recognition," IEEE T-PAMI, vol. 36, no. 7, pp.1757–1772.

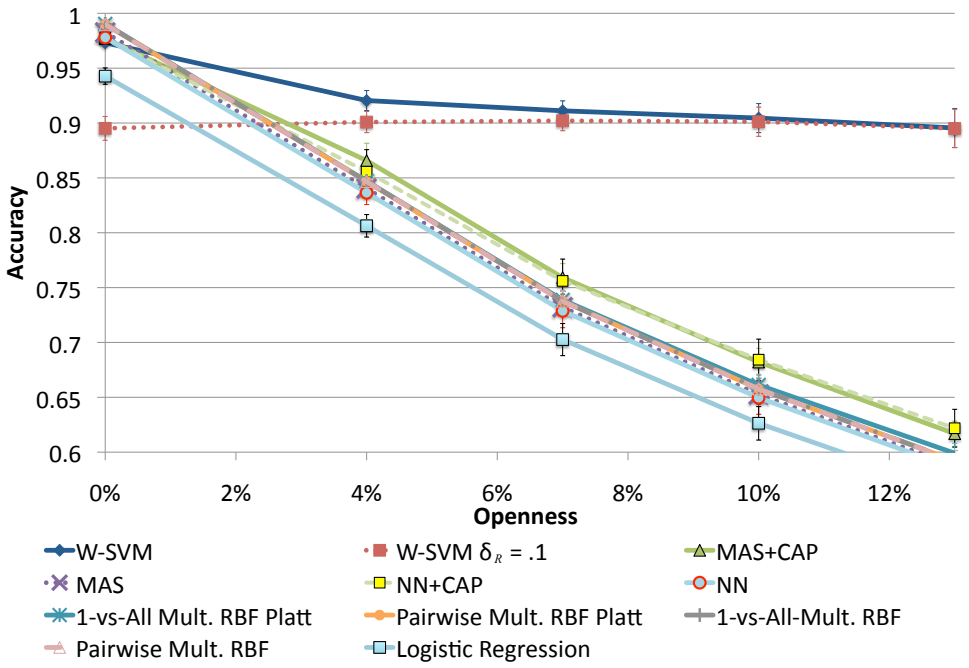

Figure 2. Accuracy for open set multi-class recognition on OMNIST. W-SVM again maintains high F-measure scores as the problem grows to be more open, but common multi-class SVMs and existing thresholded probability estimators degrade quickly. Nearest neighbor with a CAP model and MAS with a CAP model are again better than their baselines. All algorithms except the W-SVM, NN+CAP, MAS+CAP, and Logistic Regression are comparable and visually overlap. Error bars reflect standard deviation.

B Algorithmic Pseudocode for W-SVM

Alg. 1 provides a precise description of the W-SVM EVT probability modeling for each of the k classes considered in multi-class recognition. Each class y_i in the multi-class problem is represented individually as the combination of a one-class RBF SVM and a 1-vs-All binary RBF SVM. It assumes that all of the individual one-class SVMs and 1-vs-All binary SVMs have been trained *a priori*. We use (and refer to) functions in the open source libMR library, which is available from http://metarecognition.com.

We estimate the tail size q_i (number of extrema) based on the number of support vectors for the associated SVMs, using 0.5 times the number of support vectors for our detection task, which has very high openness and hence needs strong rejection, and 1.5 times the number of support vectors for our multi-class recognition task, which is more permissive at a lower level of openness.

For the binary SVMs, parameters are estimated for both the non-match (where we fit a Reverse Weibull) and the match (where we fit a Weibull) distributions. Note that in doing so, the match data for one class will be considered non-match data for another. The sets of decision scores d_i^-, d_i^+ and d_i^o are collected from the non-match and the match scores for class y_i. We use libMR to compute the parameters. We note the match scores are effectively transformed in the library, so that larger values are the points closer to the margin.

In steps 5, 6 and 7, we call off to the library function FitSVM to estimate the key Reverse Weibull or Weibull parameters. The first parameter is the set of scores, the second is the class label, the third indicates if the class of interest should have positive scores, the fourth is a relatively self-evident enum indicating the type of model, and the fifth is the tail size to use in fitting.

Alg. 2 gives a description of the W-SVM probability estimation for a new test sample. The function computes ι, P_η and P_ψ for each class. Given these values one can use Eq. 9 from the paper to make a final decision. In terms of computational costs, the EVT fittings and probability estimations are very fast, taking only milliseconds for each sample. Compared to using Platt-style Sigmoids and cross-validation for parameter estimation, the W-SVM is much faster.

Source code for the W-SVM (provided as a patch to the popular LIBSVM package) will be released after this article has been published.

Algorithm 1 W-SVM Multi-Class Model Fitting

Require:
Classes y_i, i=1 $\rightarrow k$;
Pre-trained 1-vs-All binary SVM for each class y_i, with score functions f_i and support vectors α_i^-, α_i^+;
Pre-trained one-class SVM for each class y_i, with score functions f_i^o and support vectors α_i^o;
Labeled training data X_i;
Tail size multiplier $\theta = 0.5$ for detection, or $\theta = 1.5$ for multi-class recognition
1-vs-all binary SVM scores $s_{i,j} = f_i(x_j), x_j \in X_i$;
One-class SVM scores $o_{i,j} = f_i^o(x_j), x_j \in X_i$;
Meta-Recognition object MR from libMR, providing MLE-based Weibull fitting and score calibration.

for $i = 1 \rightarrow k$ **do**
1. Let $q_i^+ = \theta \times |\alpha_i^+|$, $q_i^- = \theta \times |\alpha_i^-|$, $q_i^o = \theta \times |\alpha_i^o|$
2. Let d_i^- be the set of binary non-match scores: $s_{i,j} = f_i(x_j)$ when x_j does not belong to y_i
3. Let d_i^+ be the set of binary match scores: $s_{i,j} = f_i(x_j)$ when x_j belongs to y_i
4. Let d_i^o be the set of one-class scores: $o_{i,j} = f_i^o(x_j)$ when x_j belongs to y_i
5. Let $[\nu_{\psi,i}, \lambda_{\psi,i}, \kappa_{\psi,i}]$ = MR.fitSVM(d_i^-, y_i, false, MetaRecognition::complement_model, q_i^+)
6. Let $[\nu_{\eta,i}, \lambda_{\eta,i}, \kappa_{\eta,i}]$ = MR.fitSVM(d_i^+, y_i, true, MetaRecognition::positive_model, q_i^-)
7. Let $[\nu_{o,i}, \lambda_{o,i}, \kappa_{o,i}]$ = MR.fitSVM(d_i^o, y_i, false, MetaRecognition::positive_model, q_i^o)

end for
return $\mathcal{W} = [\nu_{\eta,i}, \lambda_{\eta,i}, \kappa_{\eta,i}, \nu_{\psi,i}, \lambda_{\psi,i}, \kappa_{\psi,i}, \nu_{o,i}, \lambda_{o,i}, \kappa_{o,i}]$

Algorithm 2 W-SVM Multi-Class Probability Estimation

Require:
Classes y_i, i=1 $\rightarrow k$;
Pre-trained 1-vs-All binary SVM for each class y_i, with score function f_i;
Pre-trained one-class SVM for each class y_i, with score functions f_i^o;
Meta-Recognition object MR from libMR and parameter set $\mathcal{W}$.

function W-SVM-PREDICT(test sample x)
for $i = 1 \rightarrow k$ **do**
$\iota_i = 0$
if $x > \nu_{o,i}$ **then**
$P_{o,i} =$ MR.W_score$(f_i^o(x); \nu_{o,i}, \kappa_{o,i}, \lambda_{o,i}) = \left(1 - e^{-\left(\frac{-f_i^o(x)-\nu_{o,i}}{\lambda_{o,i}}\right)^{\kappa_{o,i}}}\right)$
if $P_{o,i} > 0.001$ **then** $\iota_i = 1$;
end if
else $P_{o,i} = 0$
end if
if $x > \nu_{\eta,i}$ **then**
$P_{\eta,i} =$ MR.W_score$(f_i(x); \nu_{\eta,i}, \kappa_{\eta,i}, \lambda_{\eta,i}) = \left(1 - e^{-\left(\frac{-f_i(x)-\nu_{\eta,i}}{\lambda_{\eta,i}}\right)^{\kappa_{\eta,i}}}\right)$
else $P_{\eta,i} = 0$
end if
if $-x > \nu_{\psi,i}$ **then**
$P_{\psi,i} =$ MR.W_score$(f_i(x); \nu_{\psi,i}, \kappa_{\psi,i}, \lambda_{\psi,i}) \left(e^{-\left(\frac{f_i(x)-\nu_{\psi,i}}{\lambda_{\psi,i}}\right)^{\kappa_{\psi,i}}}\right)$
else $P_{\psi,i} = 0$
end if
end for
return $[P_{\eta,1}, P_{\psi,1}, \iota_1 \ldots, P_{\eta,k}, P_{\psi,k}, \iota_k]$
end function

C Step-by-Step Examples for W-SVM Training and Testing

In order to give the reader a more intuitive look at the operation of the W-SVM algorithm, we include a series of illustrations for three examples: training, testing when the input comes from a known class, and testing when the input comes from an unknown class. These illustrations correspond to the description of the W-SVM algorithm provided in Sec. IV of the main article, and the pseudocode found above in this supplemental material.

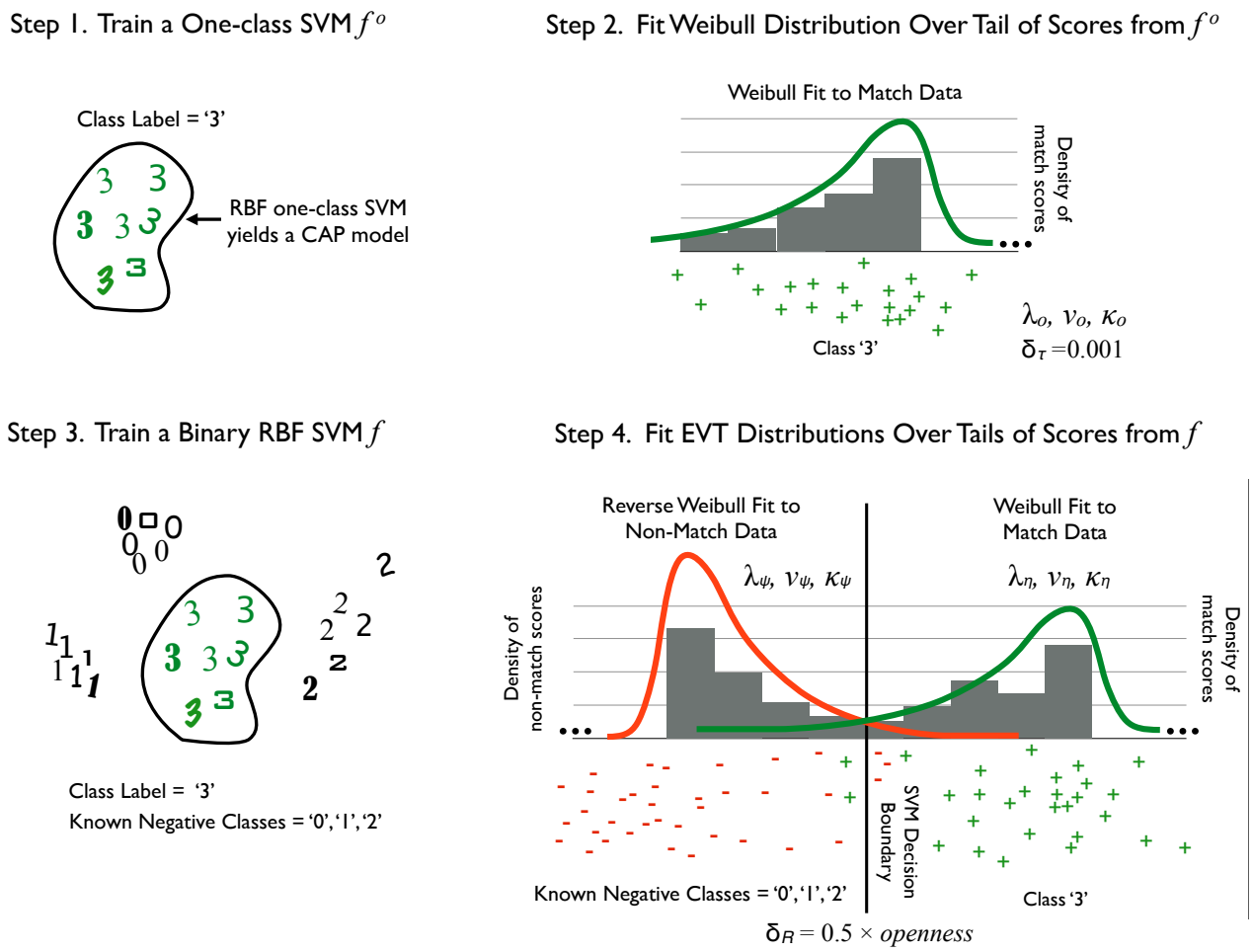

Figure 3. W-SVM Training. For simplicity, we show the procedure for a single class (the same steps are repeated for k known classes during training). In steps 1 and 2, we train a one-class SVM f^o and fit a Weibull distribution to the corresponding tail of scores from that SVM for samples of data known at training time. This gives us a Weibull model of the data with parameters $\lambda_o, \nu_o, \kappa_o$. We set the one-class probability threshold δ_τ to 0.001. In steps 3 and 4, we train a binary SVM f and fit two EVT distributions to the corresponding scores from that SVM for samples of data known at training time. A Reverse Weibull distribution is fit to the tail of the non-match data (scores for known classes other than '3'), giving us a model with parameters $\lambda_\eta, \nu_\eta, \kappa_\eta$. A Weibull distribution is fit to the tail of the match data (scores for class '3'), giving us a model with parameters $\lambda_\psi, \nu_\psi, \kappa_\psi$. We set the rejection threshold δ_R for the overall W-SVM to $0.5 \times openness$ (see Eq. 10 in the paper for a definition of $openness$). The collection of SVM models, EVT distribution parameters, and thresholds constitute the W-SVM.

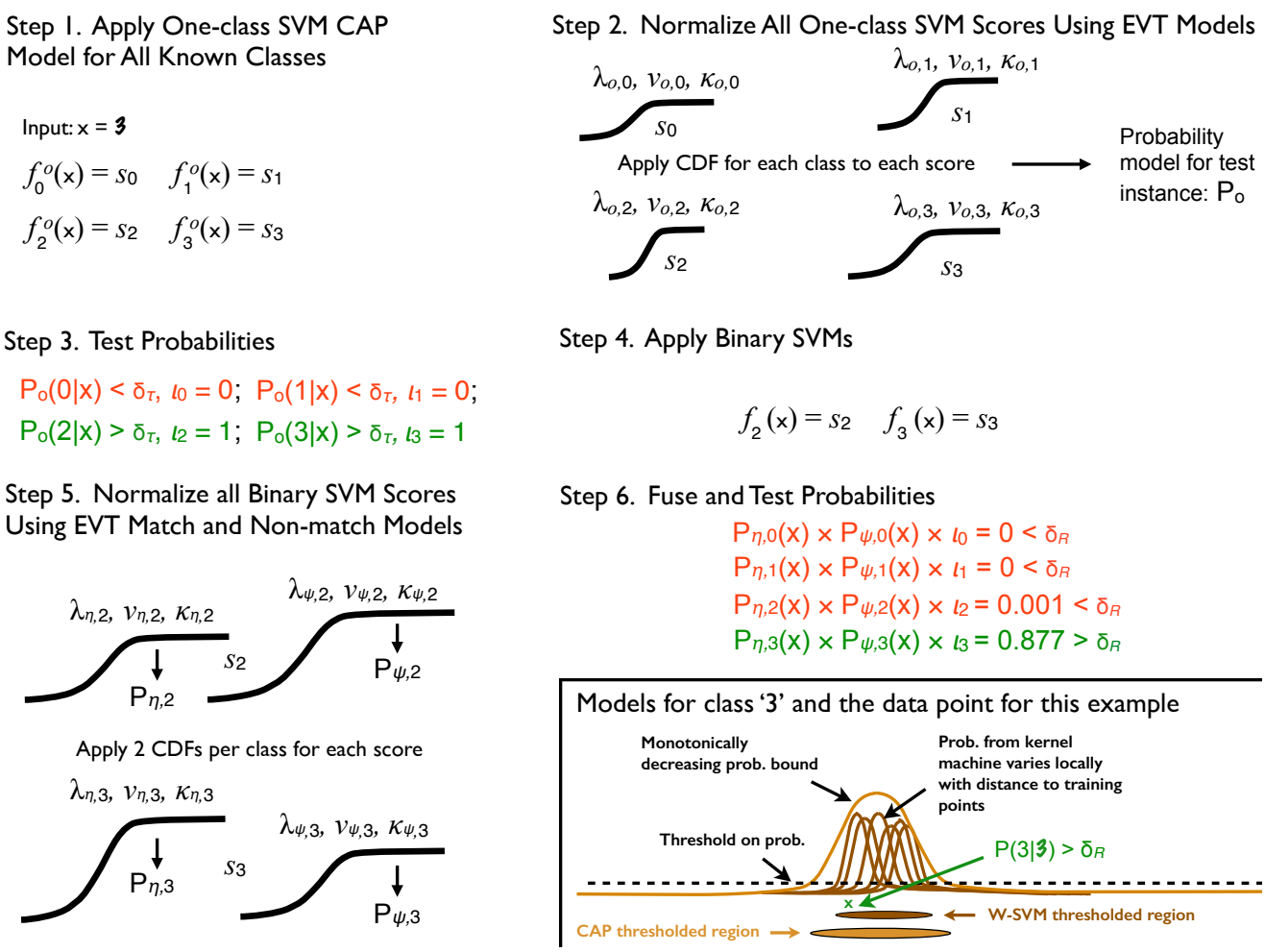

Figure 4. W-SVM testing when the input comes from a known class ('3'). Classes marked in red indicate rejection, and those marked in green indicate acceptance. The first stage of the algorithm (steps 1 – 3) applies the one-class SVM CAP models from each class to the input data, normalizes the resulting scores to probabilities, and then tests each probability against the threshold δ_τ. The indicator variable ι_y is set to 1 if the probability exceeds δ_τ, and 0 otherwise (thus eliminating those classes as possibilities later in step 6). The second stage of the algorithm (steps 4 – 6) applies the binary SVMs from each class to the same input data, normalizes the resulting scores to two probability estimates per class (one for the match model and another for the non-match model), and then fuses and tests the probability estimates. Any fused probability exceeding the W-SVM rejection threshold δ_R indicates acceptance for a class. Note for brevity, we only show the process for classes '2' and '3' in steps 4 and 5 (short circuit evaluation is possible using the indicator variables ι_0 and ι_1; an indicator variable with a value of 0 always results in a fused probability of 0). The bottom panel on the right shows where the data point for this example falls with respect to the CAP thresholded region (defined by the one-class SVM) and the overall W-SVM thresholded region for the class '3'.

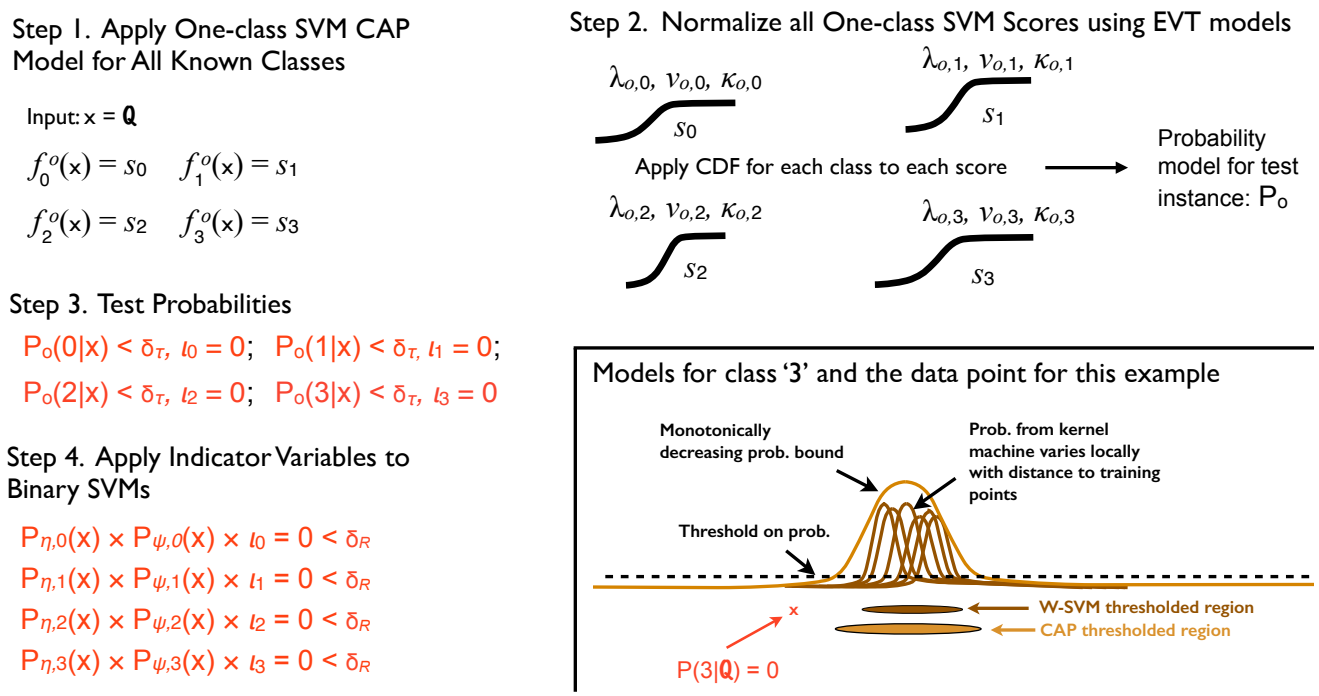

Figure 5. W-SVM testing when the input comes from an unknown class ('Q'). Classes marked in red indicate rejection. The procedure follows the same process depicted in Fig. 4. In this example, we show what happens when a data point falls outside the CAP thresholded region (*i.e.* exists in open space) for each known class. The CAP models for the knowns classes do not produce probabilities that exceed the threshold δ_τ, thus all of the indicator variables for the known classes ($\iota_0, \iota_1, \iota_2, \iota_3$) are set to 0, resulting in rejection for each class. The bottom panel on the right shows where the data point for this example falls with respect to the CAP thresholded region (defined by the one-class SVM) and the overall W-SVM thresholded region for the class '3'.