eInfochips
An Arrow Company

ARROW
Five Years Out

Extreme value machine: An algorithm for open set classification

www.einfochips.com

Abstract

Open set classification is a problem of identifying/classifying unknown classes. Unknown classes are images used neither in training nor in testing. With recent advent of Deep learning based techniques, all the conventional classification techniques are replaced with deep learning algorithms. However, Deep Learning (DL) based algorithm does not perform well in open set classification.

In this section, we will discuss about open set classification and difficulties in using DL approaches. We will also see the formulation of Extreme value machine algorithm [1], state of art algorithm used for open set classification problem. Python implementation of Extreme value machine (EVM) algorithm is available open source [2], but it is difficult to use python implementation in android applications. eInfochips recreated this algorithm on Java platform to use it in a face recognition android application.

Introduction

Open set classification is the problem of identifying/labelling classes as unknown, which were not used while training/teaching the DL algorithms. In our application, a class is a collection of images of faces, from a single person with variations in face angles. For example, a DL model trained to recognize faces of people X, Y, Z and the model gives an accuracy of 99%. Which indicates, given any of the images of X, Y, Z faces, the model will be able to predict the person 99 times out of 100.

However, what happens when a face of a new person (say A's face) is given as input? As shown in figure 1, the deep learning algorithm would classify the new face as one among the three classes trained. This shows the problem in closed set classification. The model works only with the trained dataset and does not classify the new class as 'Unknown'. EVM model helps to overcome these problems by classifying new faces as 'Unknown'. At eInfochips, a face recognition system (POC) was developed on the Snapdragon 845 board, which is an android-based application. As the available EVM model is python based, the model is recreated in Java to suit the platform.

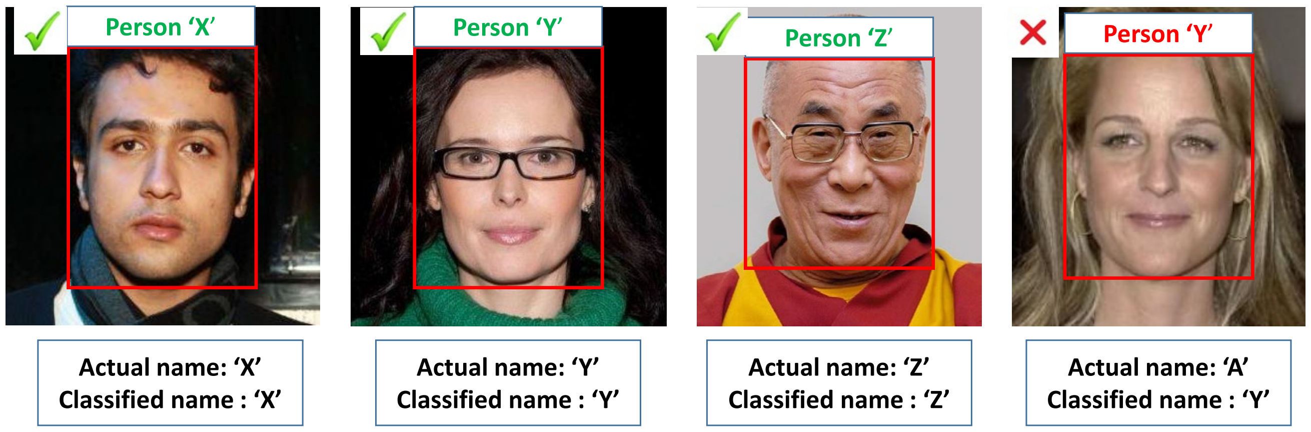


Figure 1: Shows faces of class X, Y, Z & A. Faces X, Y, Z, were classified correctly with deep learning algorithm. New face 'A' was wrongly classified as 'Y', indicates the problem in closed set classification

Known space vs Unknown space

While training the DL model, the model extracts features from individual faces. These features are multidimensional and it is difficult to visualize. Let us assume that the extracted features are one-dimensional and see how it appears in the feature space. Figure 2 shows the feature space map for 'known' and 'unknown' classes. The coloured circles are the features of the training classes X, Y and Z.

During the training process, the DL model would enclose these features inside a known space. However, there is infinite unknown space outside the known space, which is not considered while training the model. In order to overcome this problem, we use the algorithm called Extreme value machine (EVM). EVM is based on statistical Extreme value theory, and is the first classifier to be able to perform nonlinear kernel-free variable bandwidth incremental learning. While training the EVM model, along with the known space, the unknown space is also taken in to consideration. On giving A's face as an input (figure 1), the model will place it on the unknown space and classify it as 'Unknown' class. This model plays a critical role in applications to identify intruders, criminal identification, etc. Let us understand the basic theorem's involved in this algorithm, the entire design pipeline of our application and how this algorithm implemented inside the application.

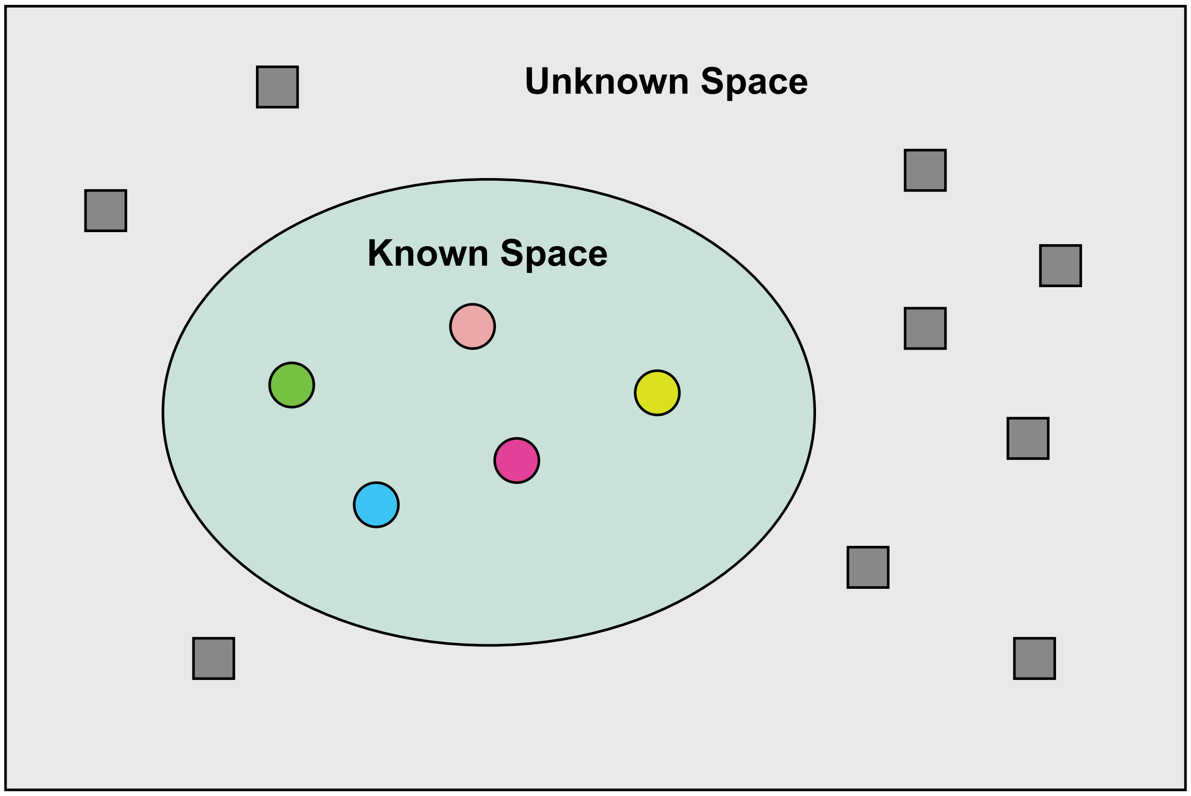


Figure 2: Representative image shows the closed known and the infinite open unknown feature space

Facial recognition pipeline

As we are going to use the EVM algorithm for facial recognition system, let us understand the steps involved in facial recognition. To develop facial recognition system, three different model are used as listed below,

- Face detection – MTCNN (Multi-cascade convolutional neural network) [3]
- Feature extraction – Facenet [4]
- Feature classification – EVM (Extreme value machines) [1]

Face detection/MTCNN model identifies the location of the faces, which are cropped and further passed to facenet model to extract unique features from each face. These unique features (Embeddings) as shown in figure 3, are the input vectors for the EVM model. EVM creates a boundary for each class, using these vectors. Before going in to the details of EVM model, let us understand the working of a simple model (classifier).

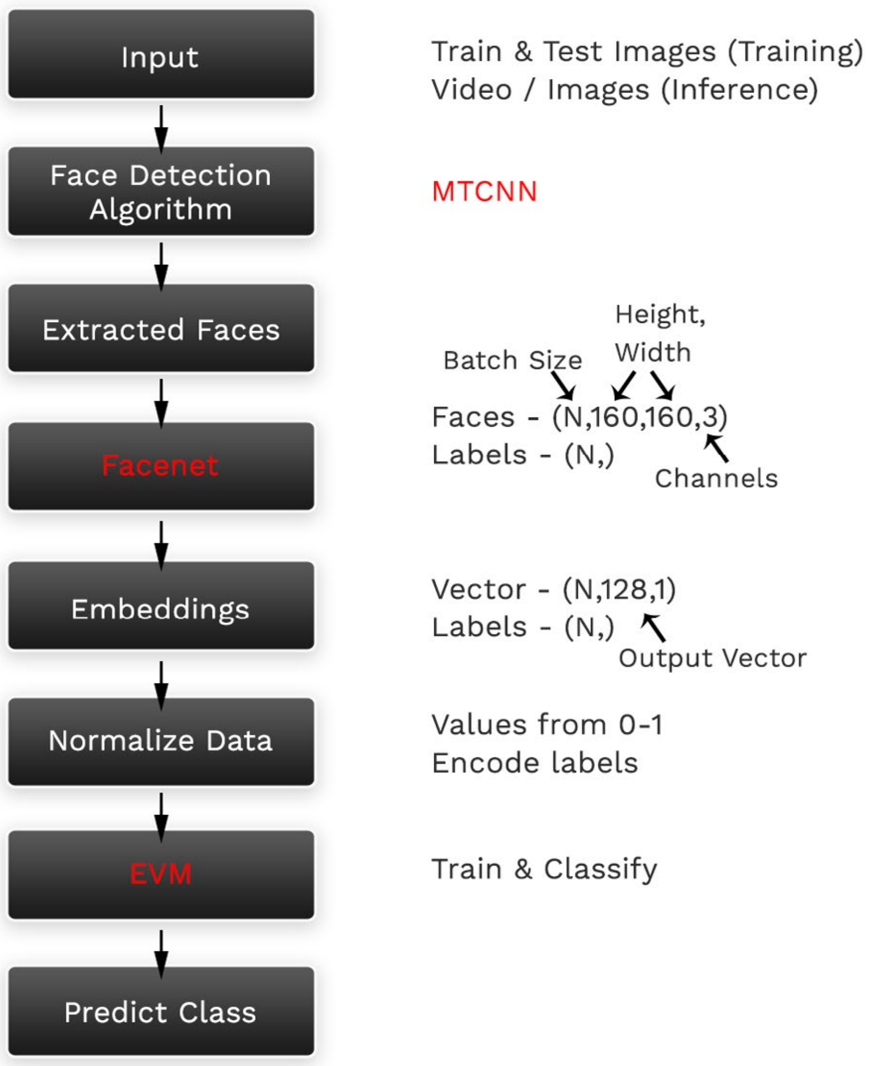


Figure 3: Flow chart of the face recognition pipeline

Simple classifier

Figure 4, shows a 2-dimensional feature space containing the features of classes X, Y and Z. Each red star represent the features from a single image/face of class X, green diamond represent features of class Y and the blue square represent the features of class Z. As shown in figure 4, the distance between two of the features of class X is D1 and distance between class X feature and the class Y feature is D2. For a given set of data, we can state the distance D1 is smaller than D2 as given in Equation 1.

$$D1 << D2 \qquad \text{Equation (1)}$$

We could clearly observe that the features representing class X (red stars) are close to each other. Therefore, the distance calculated between the red stars (D1) will be smaller than the distance between red star and green diamond (D2). Defining a threshold to this distances, let us say, **α** = 0.5*D2, any new feature vector 'A' can be compared with all the 3 classes (red arrows in figure 4). If the calculated distance is greater than the threshold **α**, then it can be defined as 'unknown' class. Else, the closest class to the feature vector 'A' is labelled.

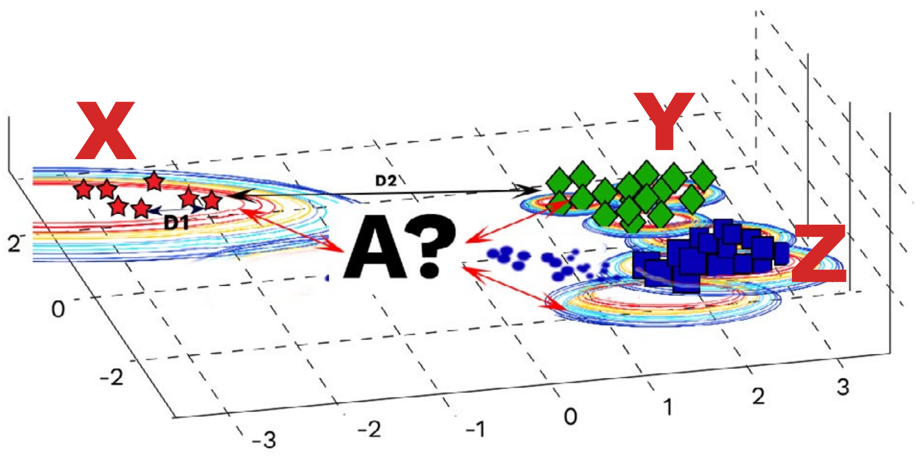


Figure 4: Sample classes X, Y and Z placed in feature space

The above technique is a simple method to classify 'unknown' and 'known' classes. However, it has drawbacks as listed below,

1. There can be thousands of samples in each class. Calculating the distances between all these samples inside and outside the class is computationally expensive.
2. Storing the features of all classes consumes memory
3. The algorithm will be slow if the data size is high.

EVM model helps to overcome these problems. To understand EVM model better, it is necessary to understand about extreme values and the distributions to model extreme values.

Statistical Extreme values

Statistically, an extreme value or extremum, are the values that occur along the tail side of the distribution. Assuming the features of the class ‘X’ follows a uniform distribution, the extremum occur at the tail of the distribution as shown in figure 5. It is either the smallest or the largest values of the given distribution function. Modelling these extreme values in a distribution function, helps us to find the probability of inclusion.

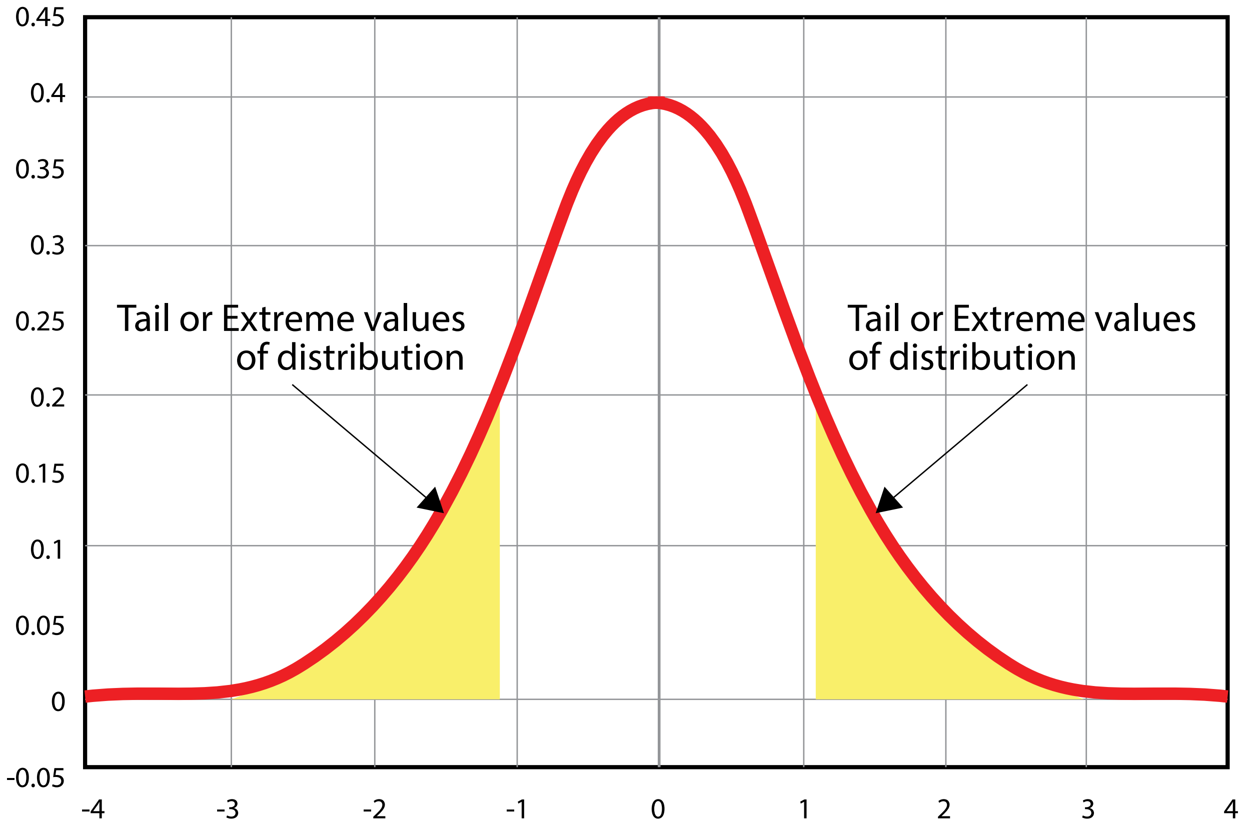


Figure 5: Distribution of feature values of class ‘X’ showing the extreme values

EVM model picks significant features to represent a particular class. To select these features, the Euclidean distance is calculated between all the features. Minimum and maximum values of the distance used to model a distribution function. As, minimum or maximum of distance values are considered as extreme values, Weibull and reversed Weibull distributions are well suited to model these extreme values [5]. Using the Probability density function (PDF) and the cumulative distribution function (CDF) of the distribution model, the probability of inclusion of the new input feature vector is calculated. Using this probability, the class name (‘known’- X, Y, Z or ‘unknown’) assigned. Let us see the details of the theorem used for modelling these distributions.

Extreme value theorem

Extreme value theorem (EVT) is used to model extreme values. In the above section, we have clearly seen about the extreme values. EVT indicates that given a well-behaved overall distribution of values (e.g., a distribution that is continuous and has an inverse), the distribution of the maximum or minimum values can assume only limited forms [1]. Fisher-Tippett Theorem is the first extreme value theorem. Let's understand the appropriate form of the theorem,

Fisher-Tippett Theorem:

Let $(S_1, S_2, \ldots . S_n)$ *be a sequence of identically individual samples.* $M\boldsymbol{ax}\,\{\boldsymbol{S_1}, \boldsymbol{S_2} . . \boldsymbol{S_n}\} = \boldsymbol{M_n}$.

Then a sequence of pairs of real numbers (a_n, b_n) *exists such that each* $a_n > 0$,

$$\left(\lim_{x\to\infty}\left(\frac{M_n - b_n}{b_n}\right)^1 > X\right) = F(x) \qquad \text{Equation (2)}$$

Then if F *is a non degenerate function, it belongs to one of the three extreme value distribution functions: the Gumbel, Frechet or Reversed weibull distributions (Equation 2)*

Margin distribution theorem:

Given a positive sample X_i *and sufficiently many negative sampels* X_j *drawn from well defined class distributions, yielding pairwise margin estimates* m_{ij}. *Assume a continuous non degenerate margin distribution exists. Then the distribution for the minimal values of the margin distance for* X_i *is given by weibull distribution.*

Weibull density function:

Given the conditions for the margin distribution theorem, the probability that x' *is included in the the boundary estimated by* x_i *is given by equation* 3.

$$A = exp^{-\left(\frac{||x_i - x'||}{\lambda_i}\right).^{\kappa_i}} \qquad \text{Equation (3)}$$

where $|x_i - x'|$ *is the distance of* x' *from sample* x_i.

κ_i, λ_i, *are the weibull shape and scale parameters respectively obtained from fitting the smallest* m_{ij}

Distance function:

Distance between two points P, Q *is calculated using the euclidean distance formula given in equation* 4

$$d(p,q)^2 = \sum_{k=0}^{n}(q_k - p_k).^2 \qquad \text{Equation (4)}$$

d *is the distance function,* p *and* q *are the feature vectors.*

Extreme Value Machine formulation

Margin Weibull computation

Equation 3, gives the Weibull distribution function for fitting the extreme values. In our case, the distance between the features of different classes are the extreme values. Our objective is to fit these distance values, obtain the shape (K), and scale (λ) parameters from equation 3. Using these values, the probability of inclusion for the new feature vector obtained. Figure 6 shows a simple representative Weibull distribution function for different K and λ values. Let's discuss the detailed step by step procedure of the EVM formulation. Assuming there are three classes to train X,Y and Z each having 3 samples.

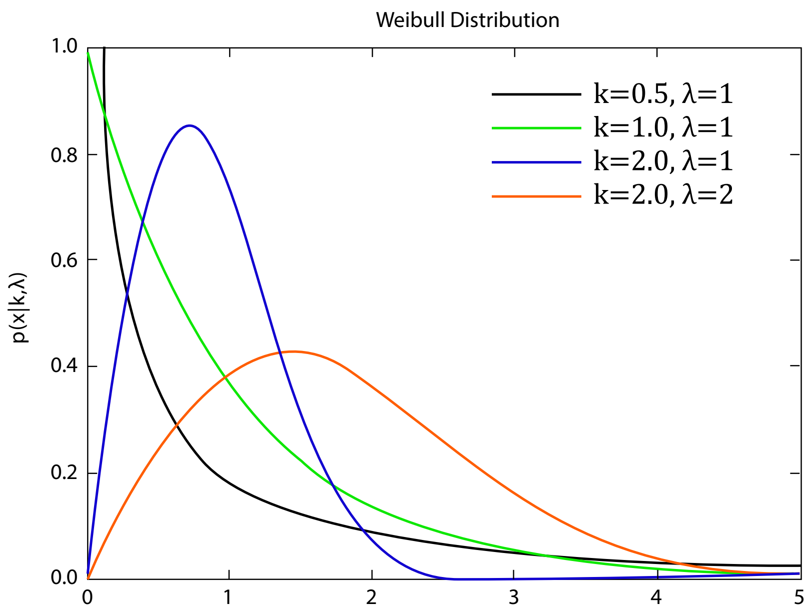


Figure 6: Weibull distribution for different K and λ values [5]

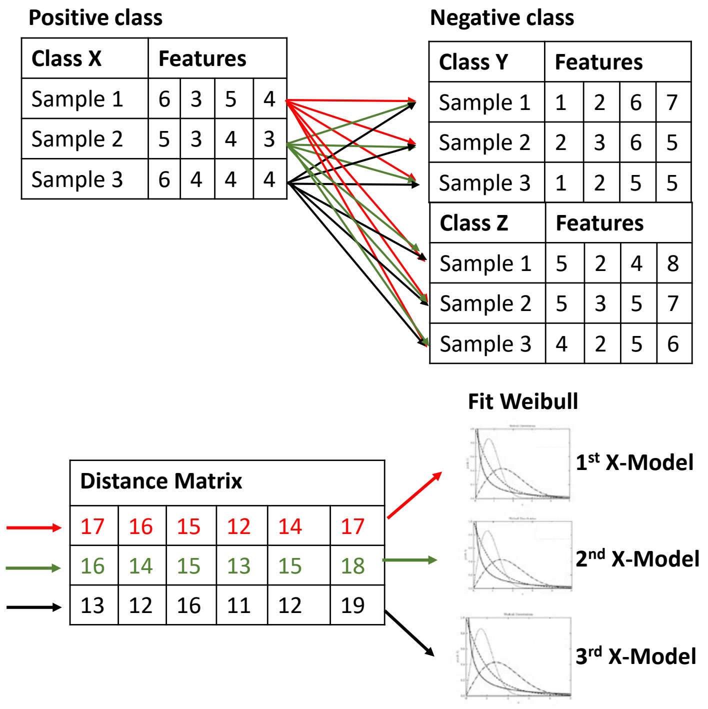
Iteration 1

Initially one class is picked as a positive class and the rest of the classes are categorized as negative classes. As shown in figure 7, class X, Y and Z has 3 sample feature vectors, each of size [1 x 4]. Initially class X is picked as positive class and the distance between sample 1 of class X and sample 1, sample 2 and sample 3 of class Y and class Z are computed using equation 4 (shown in red arrows, figure 7). That is, the distance between sample 1 of positive class and all the samples of negative classes are computed and tabulated as shown in distance matrix (1st row, shown in red). The calculated distance matrix of size [1 x 6] is modelled using the Weibull distribution function (Equation 3) and the K and λ values are calculated. Then the distance between sample 2 of positive class and sample 1, sample 2 and sample 3 of negative class is computed and modelled.

Similarly, all the samples of positive classes are used to calculate distance matrix against all the samples of negative classes. Each row of the distance matrix is used to fit the distribution function, yielding 3 distribution models. Since class X is used as a positive class, the models are named as 1st, 2nd and 3rd X-models.

Iteration 2

In the second iteration, the positive class is replaced with class Y and the negative class with class Z and class X as shown in figure 8. All the steps explained in iteration 1 is followed again in iteration 2, yielding 3 distribution models (Y-models). The iterations are repeated until all the classes are used atleast once as a positive class. In our example case, we will have 3 iterations and 9 margin weibull models in total.



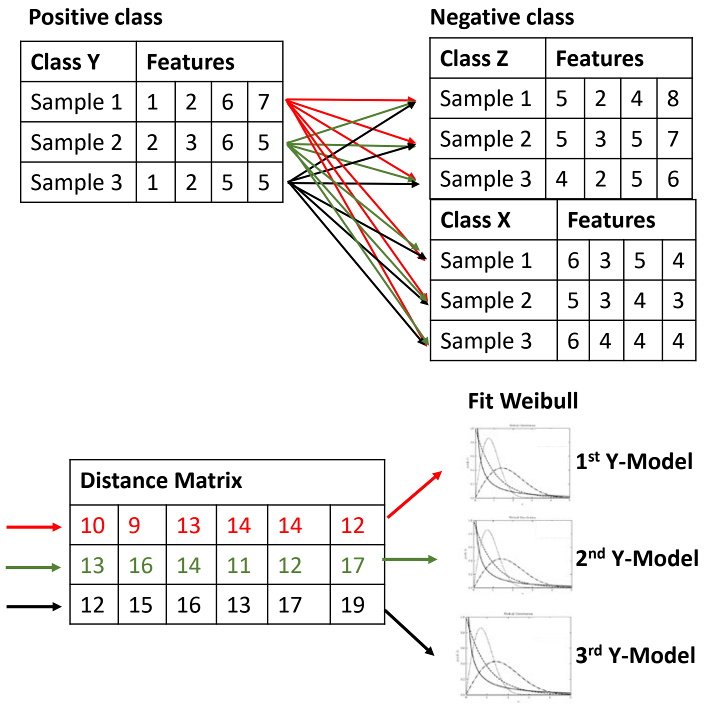
Class X	Features			
Sample 1	6	3	5	4
Sample 2	5	3	4	3
Sample 3	6	4	4	4

Class Y	Features			
Sample 1	1	2	6	7
Sample 2	2	3	6	5
Sample 3	1	2	5	5

Class Z	Features			
Sample 1	5	2	4	8
Sample 2	5	3	5	7
Sample 3	4	2	5	6

Distance Matrix					
17	16	15	12	14	17
16	14	15	13	15	18
13	12	16	11	12	19

Figure 7: Distance computation between class X as positive against class Y and Z as negative class, fitting with density function.



Class Y	Features			
Sample 1	1	2	6	7
Sample 2	2	3	6	5
Sample 3	1	2	5	5

Class Z	Features			
Sample 1	5	2	4	8
Sample 2	5	3	5	7
Sample 3	4	2	5	6

Class X	Features			
Sample 1	6	3	5	4
Sample 2	5	3	4	3
Sample 3	6	4	4	4

Distance Matrix					
10	9	13	14	14	12
13	16	14	11	12	17
12	15	16	13	17	19

Figure 8: Distance computation between class Y as positive against class Z and X as negative class, fitting with density function.

Probability matrix computation

Iteration 1

In the previous section, we have gone through the steps to compute margin Weibull's (9 models). Now let us see how to compute probability matrix. The distance between all the samples of a single positive class computed independently as shown in figure 9. For example, class X is the first positive class with 3 samples. The distance between all the 3 samples are computed. This yields a distance matrix with all the diagonal elements as zero and the elements above and below the diagonals are either zero or close to zero. We try to find the distances between all the samples inside the class. Each value of first row of the distance matrix ([0, 2, 5]) is substituted in the 1st X-model obtained from the previous section. This yields the probability of inclusion for each of the distance values. Similarly, each of the 2nd row and the 3rd row of the distance matrix is substituted on the 2nd and 3rd X-model respectively. This yields a probability matrix with diagonal elements having dominant probability values.

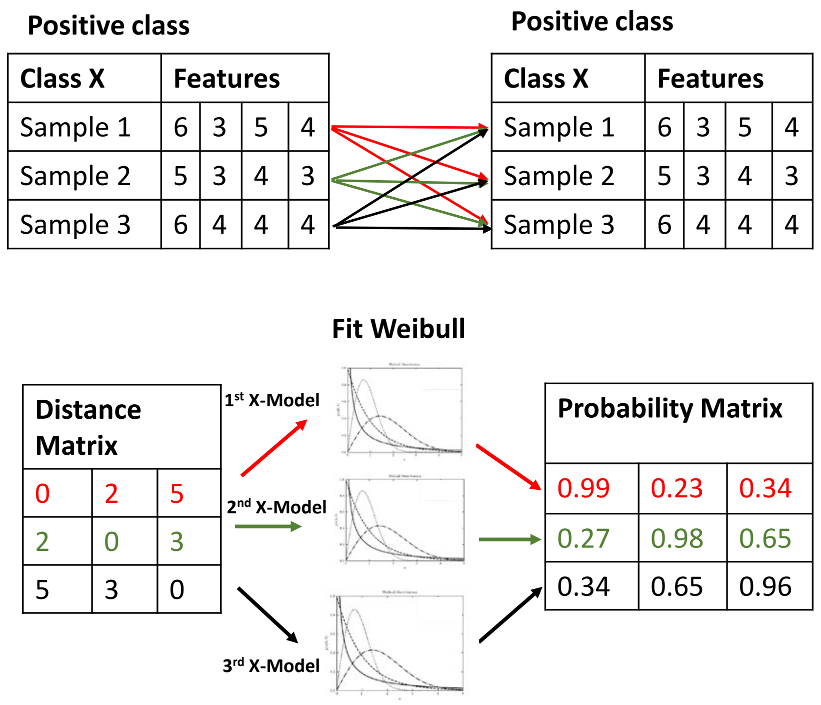


Figure 9: Probability matrix computation

Iteration 2

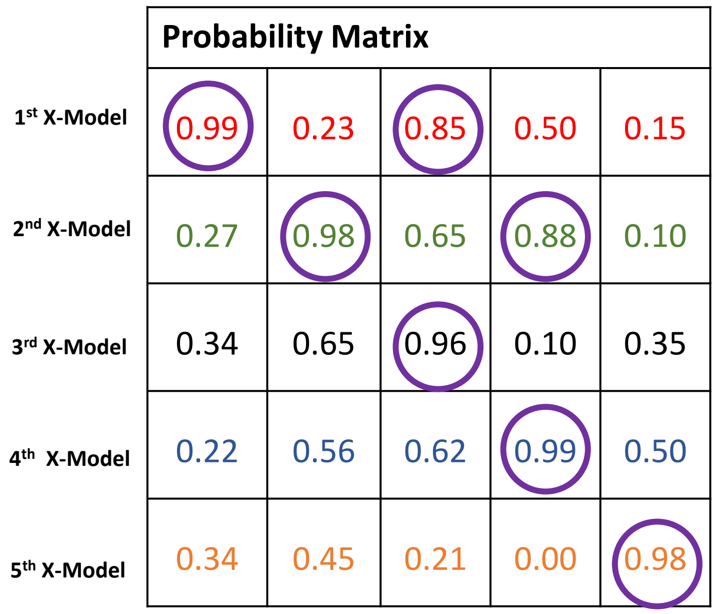
In the second iteration, the positive class is replaced with class Y and we use Y-models for probability computation. All the above steps are followed to calculate the distance and probability matrix for the class Y and class Z. This process helps to obtain 3 probability matrices one from each class.

Extreme vectors and model selection

Example 1:

Extreme vectors are the significant vectors used to select a particular model from a class. From the probability matrices obtained from the above method, we select the extreme vectors. For this process, a cover threshold value is given as input hyper parameter to the model. Let us assume that the cover threshold value is 0.97 for our examples. So, from the probability matrix, all the indices of values above 0.97 are extracted. By looking at the probability matrix in figure 9, index [1,1] and [2,2] values are above 0.97 and the rest are below the threshold. Since [1,1] and [2,2] (extreme vectors) corresponds to 1st and 2nd X-model, we store these models and reject the 3rd X-model.

Example 2:



	Probability Matrix				
1st X-Model	0.99	0.23	0.85	0.50	0.15
2nd X-Model	0.27	0.98	0.65	0.88	0.10
3rd X-Model	0.34	0.65	0.96	0.10	0.35
4th X-Model	0.22	0.56	0.62	0.99	0.50
5th X-Model	0.34	0.45	0.21	0.00	0.98

Figure 10: Sample Probability matrix for 5 positive classes

Consider a case with 5 positive classes yielding a probability matrix of size 5x5 as shown in figure 10. Let us use a cover threshold value of 0.8 for this case. We could see all the values circled in figure 10 are above the cover threshold value. The row corresponding to the 1st and 2nd X-model has more than one value above the cover threshold, that is the threshold value of index [1,3] and [2,4]. This could be because the 1st sample vector and the 2nd sample vector is close to 3rd sample vector and 4th sample vector respectively. So, we can use 1st X-model to represent both 1st and 3rd X-model and 2nd X-model to represent both 2nd and 4th X-model. By this method, we are using only 2 X-models instead of 4, reducing the space and time required for computation. The remaining 5th X-model which is not yet covered is considered as last model, having a total of 3 X-models representing the positive class X. Similarly, using the cover threshold values, the Y-models and Z-models are filtered and stored. These models are used for final probability calculation.

New feature probability calculation

From the extreme vector selection process explained, let us assume there are 2 distribution models selected for each class. The given new feature vector is substituted in each of the models and the probability value is extracted as shown in figure 11. Table 1 shows the Weibull distribution models from each class and the corresponding probability of inclusion for the given feature vector. It can be observed that the probability of inclusion reaches highest for Y-model (red circle). Thus, the input few feature can be classified as Class Y. By setting a threshold probability value, say 0.5, if we find all the probability of inclusion to be less than 0.5, then the given new feature belongs to 'unknown' class.

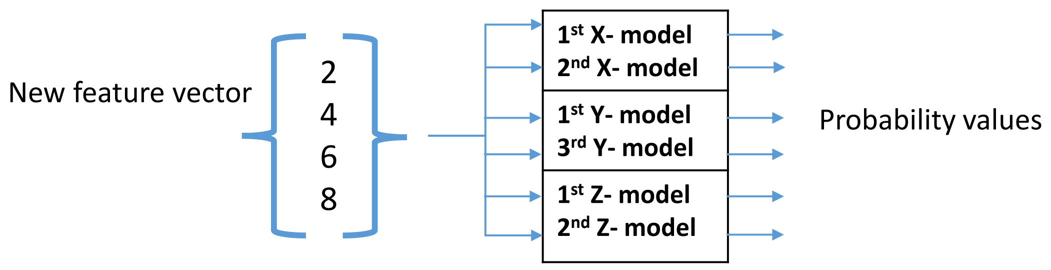


Figure 11: Schematic showing the new feature vector fed to selected models

Extreme vectors	Class	New feature probability
1st X- model 2nd X- model	Class X	0.32 0.52
1st Y- model 3rd Y- model	Class Y	0.99 0.75
1st Z- model 2nd Z- model	Class Z	0.50 0.43

Table 1: Shows the selected extreme vector models and the corresponding classes. The probability of inclusion gives the probability of the given feature.

Applications

EVM model explained in this article plays a critical role in many vision-based applications. Few of the applications where the open set classification is inevitable explained below,

Security industries

For security based applications like biometric authentication, face identification and recognition based security systems used at schools, offices, for unlocking house doors, unlocking phones etc. requires open set classification algorithm. All the above cases may end up in identifying ‘unknown’ people as ‘known’, if open set classification is not used, leads to failure of the system.

Criminal identification

Criminal identification system requires access to cameras installed in public places and needs continuous monitoring of people crossing the camera. Since the video/frame comes across public faces more than criminals, open set classification algorithm helps to reject these faces as ‘Unknown’.

Driver verification

Driver verification system helps to identify the driver while starting the vehicle. In case of ‘unknown’ person trying to start the vehicle multiple times, a notification can be sent to the owner of the vehicle.

Passport verification

Passport verification system matches the photo in the passport with the person holding the passport. If a different person is holding the passport compared to the photo pasted, a notification is sent to the verification officer. If there is a mismatch in the faces, EVM model helps to tag the face as ‘Unknown’.

Conclusion

In this article, we have checked the mathematical formulations used in Extreme value machine and the detailed step by step procedure to design a EVM model. This model is light and robust for open set classification problems. EVM python implementation is readily available but there are no Java implementations available open source. At Einfochips, the python implementation is recreated using Java and implemented in SnapDragon 845 board for face recognition application.

Authors

Somasundaram S., works as a Data scientist at Einfochips, an Arrow Company. He has more than 6 years of experience in various domains like Digital Image processing, Machine learning and Deep learning techniques. He holds a Master of Science degree from Indian Institute of Technology, Madras.

Vaibhav Kulkarni., works as a Engineer at Einfochips, an Arrow Company. He has more than 2 years of experience in Android Application development. He holds a Bachelor of Computer Science degree from SVERI's College of Engineering, Pandharpur.

References:

1. E. M. Rudd, L. P. Jain, W. J. Scheirer and T. E. Boult, "The Extreme Value Machine," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 3, pp. 762-768, 1 March 2018, doi: 10.1109/TPAMI.2017.2707495.
2. https://pypi.org/project/EVM/all
3. K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.
4. F. Schroff, D. Kalenichenko, J. Philbin, "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)", 2015, pp. 815-823.
5. Walter J. Scheirer; Gerard Medioni; Sven Dickinson, Extreme Value Theory-Based Methods for Visual Recognition , Morgan & Claypool, 2017.