Project Presentation

Calvin Smith
Harald Westling
Konrad Pohl

MVE385/MSA520
Gothenburg University and Chalmers University
Gothenburg, Sweden 2021

Background

Problem introduction

The problem is known as an **open set recognition problem** for neural networks.

Data from traffic $\rightarrow$
NN 1 (or NN 2) $\rightarrow$
10-dim feature vector/observation $\rightarrow$
Classification between in-distribution and out-of-distribution.

Splitting the data

We are considering three different data points:

- **Inliers**: Correctly classified observations from the in-distribution set.
- **Missclassified**: Missclassified observations from the in-distribution set.
- **Out-of-distribution (OOD)**: Observations from the out-of-distributions set.

The task is to separate the **inliers** from the joint set of **missclassified** and **OOD**. Observations from this joint set will henceforth be referred to as **outliers**.

Problem description

Main task:

- Separate inliers from outliers.

secondary tasks:

- Separate between the 3 classes: Inliers, Missclassified, OOD.
- Separate between points from NN1 and NN2

Theory/Methods

DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) is an unsupervised clustering algorithm. The algorithm has three main advantages:

1. No need to specify the number of clusters to be found.
2. Can find clusters of any shape.
3. Can detect outliers.

The main idea behind the algorithm is to identify dense regions of the data, by assigning a point to a cluster if it is ”close” to many other points from that cluster. In other words, the algorithm uses the geometry of the feature space to perform clustering.

DBSCAN

- Two main parameters **eps** and **minPts**.
- **eps** is defined as the radius of the neighbourhood around a point **x**.
- Two points are neighbours if the distance between them is less than or equal to **eps**.
- **minPts** is the minimum number of points required to define a cluster.
- These parameters have to be chosen before running the algorithm.
- Highly specific to the data you are working with.

Based on **eps** and **minPts** observations are classified in three different categories:

- **Core point**: A point is classified as a core point if there are at least **minPts** number of points within its radius **eps**.
- **Border point**: A point is classified as a border point if it is within the radius of a core point but there are less than **minPts** within its own radius **eps**.
- **Outlier**: A point is classified as an outlier if it is neither a core point nor a border points.

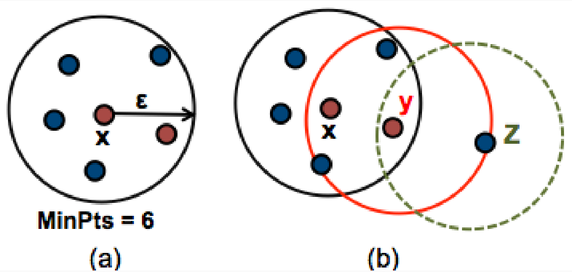

Figure 1: DBSCAN Clustering.

- (a) **x** is a **core point** since there are six points (including x) within its radius **eps**.
- (b) **x** is a **core point**, **y** is a **border point** and **z** is an **outlier**.

Algorithm:

1. Choose **eps** and **minPts**.
2. Randomly choose starting point. Determine its neighbourhood using **eps**. If there are at least **minPts** nr of points within its neighbourhood the clustering formation begins. Otherwise it is marked as an outlier. When clustering formation begins, all the points within the neighbourhood of the initial point become part of a cluster, and if these points are also core points then their neighbourhood points are added to the cluster, and so on.
3. Randomly choose a new starting point that has not yet been visited and repeat step 2.
4. Continue until all points have been visited.

Mixture Discriminant Analysis (MDA)

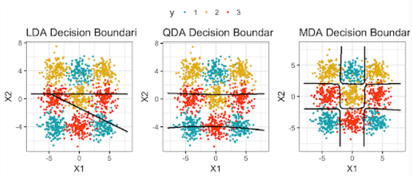

Figure 2: MDA visualized incomparison to LDA and QDA. As presented in the big data course.

For class k (the colors), with R_k subclasses (=3):

$$P(X|G=k) = \sum_{r=1}^{R_k} \pi_{kr}\phi(X;\mu_{kr},\Sigma)$$

•The EM-algoritm finds the boundaries.

•Σ is the same, for all subclusters.

•Different amounts of subclusters are possible for each class.

- Unsupervised algorithm
- Anomaly detection
- Recursively partitions the data

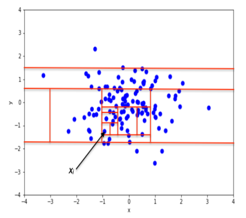

Figure 3: The recursive partitioning in isolation forest

Area Under Receiver Operating Characteristic curve (AUROC)

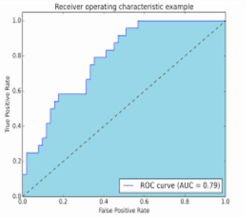

Figure 4: Example of AUROC

- One of the most common evaluation metrics.
- The ROC plots the Fpr vs the Tpr
  - $Tpr = \frac{TP}{TP+FN}$
  - $Fpr = \frac{FP}{FP+TN}$
- Requires a vector of bias for belonging to the binary classes for each element in the dataset.
- Calculate ROC curve using one point.

Implementation

Splitting the data

We are considering three different data points:

- **Inliers**: Correctly classified observations from the in-distribution set.
- **Missclassified**: Missclassified observations from the in-distribution set.
- **Out-of-distribution (OOD)**: Observations from the out-of-distributions set.

The task is to separate the **inliers** from the joint set of **missclassified** and **OOD**. Observations from this joint set will henceforth be referred to as **outliers**.

Splitting the data

- Training, validation and testing sets
- validation and testing sets in 2 versions: Large (largest possible) and Small (5% outliers compared to training).
- The proportions were handled as:
  - Inliers: constant over all 10 classes.
  - Missclassified: constant over all 10 classes (20%)
  - OOD: The proportions were obtained from the OOD-data and kept in the 3 subsets train, valid, testing.
- **50-50 between Inliers and Outliers** for getting valid AUROC-scores.

General strategy

Each method was implemented on a subset of the data:

1. for(`i` in 1:10) {
2. `data` = subset(`data`, `class==i`)
3. apply method
   }

In contrast to applying the method directly.

DBSCAN Implementation

- The DBSCAN algorithm was implemented in R using the **dbscan**-package.
- The main challenge is to determine the optimal values of **eps** and **minPts**.
- This was done using a grid search over a grid of possible values for the parameters.

DBSCAN Implementation: Choosing eps and minPts

Using the **training** and **validation** sets. For each combination of **eps** and **minPts**:

1. Apply the DBSCAN algorithm on the **training** set, obtaining a clustering object.
2. Using the clustering object, make predictions on the **validation** set. Only interested in distinguishing between **inliers** and **outliers**. All observations belonging to a cluster are grouped together. The rest are classified as **outliers**.
3. Binary classification problem, 0 (outlier) and 1 (non-outlier). Based on the predictions, calculate the AUROC-score.
4. Choose the combination of **eps** and **minPts** that produces the highest AUROC-score.

DBSCAN Implementation

- This method has been applied for each class in our data ($k = 0, 1, ..., 9$).
- One AUROC-score for each class.
- The optimal **eps** and **minPts** that gives the highest AUROC-score is stored and used for the final testing of DBSCAN's performance on the **test** set.

DBSCAN Implementation

To further analyze the performance of DBSCAN with respect to the "robustness" of the method against outliers, three different **training**-sets have been constructed:

1. The full training set X_{full}: Contains 12000 observations with 50% in-distribution points (80% Inliers, 20% Missclassified) and 50% out-of-distribution.
2. Subset of full training set $X_{in,miss}$: Only containing Inliers and Missclassified. 6000 observations (80/20).
3. Subset of full training set X_{in}: Only containing Inliers. 4768 observations.

DBSCAN Implementation

So for each training set:

1. Apply DBSCAN-algorithm.
2. Predict on the small and large validation sets.
3. Obtain best AUROC-score and optimal **eps** and **minPts**.
4. Apply DBSCAN on the training set again with the optimal parameters, predict on the small and large test sets.
5. Final result. Per class AUROC-score and confusion matrix.

MDA

- Implement in `R` with "mda::mda()".
- Requires data from all classes to work.
- Does not output AUROC, so ROC and AUROC are calculated manually.

- Implemented using Python Scikit-learn implementation
- The algorithm outputs a bias vector between 1 and -1 where lower value means higher probability of the element being an anomaly.
- Found best possible cutoff value from the training set, using a grid search of potential values, according to the manually calculated AUROC value.
- Use the stored cutoff value to get prediction on the validation/test datasets.

Results

DBSCAN - train X_{full}

Classes	0	1	2	3	4	5	6	7	8	9	AVG
X_{large}	0.794	0.845	0.756	0.723	0.728	0.735	0.753	0.810	0.830	0.804	**0.78**
X_{small}	0.675	0.850	0.675	0.625	0.650	0.725	0.850	0.850	0.850	0.775	**0.75**

Table 1: AUROC values when training on X_{full} and predicting on X_{large}, X_{small}.

		Actual 0	Actual 1
Pred	0	1280	376
Pred	1	336	1209

Table 2: acc = 0.77

		Actual 0	Actual 1
Pred	0	149	48
Pred	1	51	152

Table 3: acc = 0.75

DBSCAN - train data X_{full} test data X_{large}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	57	32	54	90	32	61	26	33	17	13	415
correct_miss	57	32	54	90	32	61	26	33	17	13	415
acc_miss	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
nr_ood	61	44	159	168	147	120	201	69	98	134	1201
correct_ood	33	30	128	99	86	75	170	48	88	108	865
acc_ood	0.54	0.68	0.81	0.59	0.59	0.62	0.85	0.70	**0.90**	0.81	**0.72**

DBSCAN - train data X_{full}, test data X_{small}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	9	8	3	10	1	4	1	9	3	0	48.00
correct_miss	9	8	3	10	1	4	1	9	3	0	48
acc_miss	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		**1.00**
nr_ood	11	12	17	10	19	16	19	11	17	20	152
correct_ood	2	8	6	7	11	13	17	7	14	16	101
acc_ood	0.18	0.67	0.35	0.70	0.58	0.81	**0.89**	0.64	0.82	0.80	**0.66**

DBSCAN - train $X_{in,miss}$

Classes	0	1	2	3	4	5	6	7	8	9	AVG
X_{large}	0.792	0.829	0.744	0.723	0.728	0.741	0.769	0.815	0.851	0.8	**0.78**
X_{small}	0.675	0.850	0.7	0.625	0.7	0.725	0.850	0.825	0.850	0.775	**0.75**

Table 4: AUROC values when training on $X_{in,miss}$ and predicting on X_{large}, X_{small}.

		Actual 0	Actual 1
Pred	0	1265	351
Pred	1	351	1235

Table 5: acc = 0.78

		Actual 0	Actual 1
Pred	0	154	51
Pred	1	46	149

Table 6: acc = 0.757

train data $X_{in,miss}$, test data X_{large}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	57	32	54	90	32	61	26	33	17	13	415
correct_miss	57	32	54	90	32	61	26	33	17	13	415
acc_miss	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
nr_ood	61	44	159	168	147	120	201	69	98	134	1201
correct_ood	30	26	129	110	92	76	142	54	82	109	850
acc_ood	0.49	0.59	0.81	0.65	0.63	0.63	0.71	0.78	**0.84**	0.81	**0.71**

DBSCAN - train data $X_{in,miss}$, test data X_{small}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	9	8	3	10	1	4	1	9	3	0	48
correct_miss	9	8	3	9	1	4	1	8	3	0	46
acc_miss	1.00	1.00	1.00	0.90	1.00	1.00	1.00	0.89	1.00		**0.96**
nr_ood	11	12	17	10	19	16	19	11	17	20	152
correct_ood	2	10	7	8	13	13	17	8	14	16	108
acc_ood	0.18	0.83	0.41	0.80	0.68	0.81	**0.89**	0.73	0.82	0.80	**0.71**

DBSCAN - train X_{in}

Classes	0	1	2	3	4	5	6	7	8	9	AVG
X_{large}	0.789	0.828	0.741	0.743	0.734	0.735	0.769	0.815	0.851	0.79	**0.78**
X_{small}	0.675	0.825	0.7	0.650	0.700	0.725	0.850	0.850	0.850	0.775	**0.76**

Table 7: AUROC values when training on X_{in} and predicting on X_{large}, X_{small}.

		Actual 0	Actual 1
Pred	0	1171	327
	1	445	1258

Table 8: acc = 0.76

		Actual 0	Actual 1
Pred	0	155	51
	1	45	149

Table 9: acc = 0.76

DBSCAN - train data X_{in}, test data X_{large}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	57	32	54	90	32	61	26	33	17	13	415
correct_miss	57	32	54	90	32	61	26	33	17	13	415
acc_miss	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.00**
nr_ood	61	44	159	168	147	120	201	69	98	134	1201
correct_ood	35	26	116	101	87	62	142	54	82	109	814
acc_ood	0.57	0.59	0.73	0.60	0.59	0.52	0.71	0.78	**0.84**	0.81	**0.68**

DBSCAN - train data X_{in}, test data X_{small}

	0	1	2	3	4	5	6	7	8	9	TOT
nr_miss	9	8	3	10	1	4	1	9	3	0	48
correct_miss	9	8	3	10	1	4	1	9	3	0	48
acc_miss	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		**1.00**
nr_ood	11	12	17	10	19	16	19	11	17	20	152
correct_ood	2	9	7	8	13	13	17	8	14	16	107
acc_ood	0.18	0.75	0.41	0.80	0.68	0.81	**0.89**	0.73	0.82	0.80	**0.70**

	Main task	Inliers-Missclassified
X_test	0.742	0.991
small_test	0.777	0.998

Table 10: AUROC values for the 4 tested settings.

Isolation forest

Classes	AUROC	Manual AUROC
0	0.88	0.79
1	0.95	0.86
2	0.81	0.71
3	0.82	0.71
4	0.83	0.73
5	0.81	0.73
6	0.84	0.71
7	0.93	0.84
8	0.86	0.73
9	0.82	0.7
Avg value	0.86	0.75

Table 11: AUROC values using the isolation forest algorithm.

Conclusions

DBSCAN

- Quite similar results. Robust against variations in training data.
- Training with no outliers yields slightly better results. This is promising.
- The Neural Network is trained on a specific set of objects, so a method that doesn't need examples of outliers is good.
- Overall very good at separating **Inliers** from **Missclassified** but harder to separate **Inliers** from **OOD**.
- However, it seems like for some classes, the ability to separate **Inliers** from **OOD** is very good.

DBSCAN

- The implementation when choosing **eps** and **minPts** can be improved.
- Different choice of grid yields different results. Intuition based.
- Several combinations of the parameters yield the same optimal AUROC value. But can only choose one combination on the test data –> not necessarily optimal for the test data.

MDA

- Suitable for separation between Inliers and missclassified.
- Easy to implement.

Isolation forest

- Suitable for separation with few outliers.