Gaussian Mixture Models for Anomaly Detection

Karan Gandhi   Anurag Singh   Arjun Dikshit   Abhinav Khot

CS 328 : Introduction to Data Science

April 23, 2025

Outline

Outline

Problem Statement

Our main aim in this project is to improve Deep Autoencoding Gaussian Mixture Model (DAGMM) for anomaly detection in an unsupervised setting.

Introduction

Anomaly Detection in the unsupervised or semi - supervised setting is highly sought after in recent years for applications particularly in the medical industry.

Introduction

Anomaly Detection in the unsupervised or semi - supervised setting is highly sought after in recent years for applications particularly in the medical industry.
Since it is not feasible to directly use clustering in higher dimensions due to the curse of dimensionality, there exist two main paradigms:

Introduction

Anomaly Detection in the unsupervised or semi - supervised setting is highly sought after in recent years for applications particularly in the medical industry.
Since it is not feasible to directly use clustering in higher dimensions due to the curse of dimensionality, there exist two main paradigms:

- Compression to lower dimensions and then performing clustering.

Introduction

Anomaly Detection in the unsupervised or semi - supervised setting is highly sought after in recent years for applications particularly in the medical industry.
Since it is not feasible to directly use clustering in higher dimensions due to the curse of dimensionality, there exist two main paradigms:

- Compression to lower dimensions and then performing clustering.
- Training an autoencoder to compress data points to a lower-dimensional representation and classifying points which have high reconstruction error as anomalies.

What are the problems with the existing Paradigms?

- The first method has two separate processes which are independent from each other. Compression into lower dimensions may lose important information needed to classify anomalies.

What are the problems with the existing Paradigms?

- The first method has two separate processes which are independent from each other. Compression into lower dimensions may lose important information needed to classify anomalies.
- It has also been found that anomalies can also be reconstructed well with little error.

Outline

Deep Autoencoding Gaussian Mixture Models

Addresses the main challenges in both the paradigms while using each others pros. Use an autoencoder to lean the lower-dimensional representations and uses another neural network to estimate the parameters of the Gaussians.

Deep Autoencoding Gaussian Mixture Models

Addresses the main challenges in both the paradigms while using each others pros. Use an autoencoder to lean the lower-dimensional representations and uses another neural network to estimate the parameters of the Gaussians.

The main advantage is that the network is **end-to-end trainable**.

Architecture

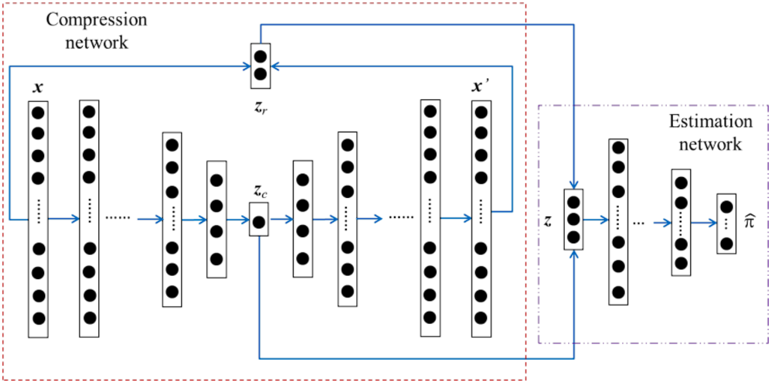

Figure: Overview of the DAGMM architecture

$$\mathbf{p} = MLP(\mathbf{z};\theta_m), \quad \hat{\gamma} = \text{softmax}(\mathbf{p})$$

Training Details

Loss Function:

$$J(\theta_e, \theta_d, \theta_m) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{x}'_i)}_{\text{reconstruction loss}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N} E(\mathbf{z}_i)}_{\text{sample energy}} + \underbrace{\lambda_2 P(\hat{\boldsymbol{\Sigma}})}_{\text{penalise degeneracy of COV}} .$$

GMM parameter updates:

$$\hat{\phi}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N}, \quad \hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}\mathbf{z}_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}, \quad \hat{\boldsymbol{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik}(\mathbf{z}_i - \hat{\mu}_k)(\mathbf{z}_i - \hat{\mu}_k)^T}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

Training Details

- Train only on **normal data**

Training Details

- Train only on **normal data**
- During Testing, classify the data points with sample energy in the **top x percentile** to be anomalous.

Brief overview

We make the following modifications to the DAGMM architecture

1. Loss Function Modification

Brief overview

We make the following modifications to the DAGMM architecture

1. Loss Function Modification
2. Inference Criteria Modification

Brief overview

We make the following modifications to the DAGMM architecture

1. Loss Function Modification
2. Inference Criteria Modification
3. Laplacian Mixture Models

Brief overview

We make the following modifications to the DAGMM architecture

1. Loss Function Modification
2. Inference Criteria Modification
3. Laplacian Mixture Models
4. VAE and VQ-VAE Architecture instead of Autoencoders

Brief overview

We make the following modifications to the DAGMM architecture

1. Loss Function Modification
2. Inference Criteria Modification
3. Laplacian Mixture Models
4. VAE and VQ-VAE Architecture instead of Autoencoders
5. Estimating the correct value of threshold, and using unlabeled data

Outline

Modifications in the Architecture

- In the current DAGMM, we are using autoencoders for encoding the data to lower dimensional representations.

Modifications in the Architecture

- In the current DAGMM, we are using autoencoders for encoding the data to lower dimensional representations.
- So, we replaced the autoencoder with Variational Autoencoder (VAE) and Vector Quantized Variational Autoencoder (VQ-VAE).

VAE

Variational Autoencoders (VAEs) are a type of autoencoder that learns a probability distribution over the data. Instead of mapping each input to a single point, VAEs map it to a range of possible values (mean and variance). This helps in learning more relevant lower dimensional representation.

VAE

Variational Autoencoders (VAEs) are a type of autoencoder that learns a probability distribution over the data. Instead of mapping each input to a single point, VAEs map it to a range of possible values (mean and variance). This helps in learning more relevant lower dimensional representation.

DAGMM–VAE Loss:

$$\mathcal{L}_{\text{VAE}} = \frac{1}{N} \sum_{i=1}^{N} \Big[-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_i)} [\log p_{\theta_d}(\mathbf{x}_i \mid \mathbf{z})] + KL\big(q_\phi(\mathbf{z} \mid \mathbf{x}_i) \,\|\, p(\mathbf{z})\big) \Big]$$

$$J(\phi, \theta_d, \theta_m) = \underbrace{\mathcal{L}_{\text{VAE}}}_{\substack{\text{recon.+} \\ \text{KL reg.}}} + \underbrace{\frac{\lambda_1}{N} \sum_{i=1}^{N} E(\mathbf{z}_i)}_{\text{sample energy}} + \underbrace{\lambda_2 \, P(\{\Sigma_k\})}_{\text{covariance penalty}}$$

VAE Results

Percentile Threshold	Precision	Recall	F-score
10	0.9686	0.4776	0.6398
20	0.9642	0.9461	0.9551
30	0.7767	0.9684	0.8620
40	0.6526	0.9846	0.7849
50	0.5616	0.9916	0.7171
60	0.4930	0.9975	0.6599

Table: Performance metrics while using VAE **(ROC AUC score: 98.28)**.

VQ-VAE in DAGMM

Vector Quantized-VAEs (VQ-VAEs) use a fixed set of learned discrete codes (like a dictionary) to represent inputs. Each input is mapped to the closest code in this dictionary, making the representation compact and discrete.

VQ-VAE in DAGMM

Vector Quantized-VAEs (VQ-VAEs) use a fixed set of learned discrete codes (like a dictionary) to represent inputs. Each input is mapped to the closest code in this dictionary, making the representation compact and discrete. By using VQ-VAE, the latent $\mathbf{z}_i = e_{k_i}$ takes only K possible values, so the GMM fits a mixture over discrete clusters. Outliers—which map to rarely used or poorly reconstructed codes—produce high sample energy and stand out better.

VQ-VAE in DAGMM

Vector Quantized-VAEs (VQ-VAEs) use a fixed set of learned discrete codes (like a dictionary) to represent inputs. Each input is mapped to the closest code in this dictionary, making the representation compact and discrete. By using VQ-VAE, the latent $\mathbf{z}_i = e_{k_i}$ takes only K possible values, so the GMM fits a mixture over discrete clusters. Outliers—which map to rarely used or poorly reconstructed codes—produce high sample energy and stand out better.

DAGMM–VQ-VAE Loss:

$$\mathcal{L}_{\text{VQ}} = \frac{1}{N}\sum_{i=1}^{N}\Big[\underbrace{\|\mathbf{x}_i - \mathbf{x}_i'\|^2}_{\text{reconstruction}} + \underbrace{\|\,\text{sg}[z_e(\mathbf{x}_i)] - e_{k_i}\|^2}_{\text{codebook update}} + \underbrace{\beta\|\,z_e(\mathbf{x}_i) - \text{sg}[e_{k_i}]\|^2}_{\text{commitment}}\Big]$$

$$J = \mathcal{L}_{\text{VQ}} \; + \; \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N} E(e_{k_i})}_{\text{sample energy}} \; + \; \underbrace{\lambda_2\, P(\{\Sigma_k\})}_{\text{covariance penalty}}$$

VQ-VAE Results

Percentile Threshold	Precision	Recall	F-score
10	0.9720	0.4804	0.6430
20	0.9663	0.9499	0.9581
30	0.7869	0.9887	0.8763
40	0.6552	0.9893	0.7883
50	0.5759	1.0000	0.7309
60	0.5336	1.0000	0.6959

Table: Performance metrics while using VQ-VAE **(ROC AUC score: 98.93)**.

Modifications in the Mixture Models

- In the current setup we use Gaussian Mixture Models, we replaced GMM with Laplacian Mixture Models and observed how they perform.

Laplacian Mixture Models

- A Laplacian Mixture Model (LMM) assumes each data point x_i comes from one of K Laplace components:

$$p(x_i) \;=\; \sum_{k=1}^{K} \pi_k \, \mathrm{Laplace}(x_i \mid \mu_k, b_k),$$

where

$$\mathrm{Laplace}(x \mid \mu, b) = \frac{1}{2b} \exp\Bigl(-\frac{|x-\mu|}{b}\Bigr), \quad \sum_{k=1}^{K} \pi_k = 1.$$

- Parameter update for LMMs:

$$\mathbf{p} = MLP(\mathbf{z}; \theta_m), \quad \gamma = \text{softmax}(\mathbf{p})$$

$$\pi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}, \quad \mu_k \leftarrow \frac{\sum_i \gamma_{ik} \, x_i}{\sum_i \gamma_{ik}}, \quad b_k \leftarrow \frac{\sum_i \gamma_{ik} \, |x_i - \mu_k|}{\sum_i \gamma_{ik}}$$

LMM Results

Percentile Threshold	Precision	Recall	F-score
10	0.9786	0.4868	0.6502
20	0.8532	0.7554	0.8014
30	0.6700	0.7680	0.7156
40	0.6183	0.9086	0.7358
50	0.5653	0.9326	0.7039
60	0.5217	0.9327	0.6691

Table: Performance metrics for LMM (constant ROC AUC = 88.91%).

Modify the loss function

- Current Setup only uses normal data for training.

Modify the loss function

- Current Setup only uses normal data for training.
- We include anomalous points and encourage the model to produce high sample energy for them.

$$J_1(\theta_e, \theta_d, \theta_m) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{x}_i')}_{\text{reconstruction loss}} + \underbrace{\frac{\lambda_1}{N}\sum_{i=1}^{N}\left(\underbrace{E(\mathbf{z}_i)}_{\text{normal data}} \quad OR \quad \underbrace{\frac{1}{E(\mathbf{z}_i)}}_{\text{anomalous data}} \right)}_{\text{sample energy}}$$

$$+ \quad \underbrace{\lambda_2 P(\hat{\mathbf{\Sigma}})}_{\text{penalise degeneracy of COV}} \quad .$$

Modify the Loss Function - Results

Percentile Threshold	Precision	Recall	F-score
10	0.5820	0.2977	0.3939
20	0.5072	0.5171	0.5121
30	0.3694	0.5641	0.4465
40	0.4207	0.8556	0.5641
50	0.3806	0.9573	0.5446
60	0.3807	0.9576	0.5448

Table: Model trained on 200 epochs with different percentile thresholds for anomaly classification. **ROC AUC score: 81.80**.

Modify the Inference Criterion

- Currently only sample energy is being used during inference.

Modify the Inference Criterion

- Currently only sample energy is being used during inference.
- In inference, it could be the case the during compression, anomalous data may get compressed to look similar to normal data and hence get lower sample energy.

Modify the Inference Criterion

- Currently only sample energy is being used during inference.
- In inference, it could be the case the during compression, anomalous data may get compressed to look similar to normal data and hence get lower sample energy.
- So we include even reconstruction loss during inference.

Results

Percentile Threshold	Precision	Recall	F-score
10	0.9464	0.4557	0.6152
20	0.9505	0.9200	0.9350
30	0.7919	0.9995	0.8837
40	0.6590	0.9996	0.7943
50	0.5952	0.9999	0.7462
60	0.5952	0.9999	0.7462

Table: Performance metrics using reconstruction loss + sample energy as inference criterion **(ROC AUC score: 98.62)**.

Estimating the correct value of threshold, and using unlabeled data

- In the original setup, the threshold is a hyperparameter that we were allowed to set.

Estimating the correct value of threshold, and using unlabeled data

- In the original setup, the threshold is a hyperparameter that we were allowed to set.
- Also, we are only using the positive samples for training. In real-life scenarios, especially in medical datasets, we typically have a really small handcrafted dataset of positive samples.

Estimating the correct value of threshold, and using unlabeled data

- In the original setup, the threshold is a hyperparameter that we were allowed to set.
- Also, we are only using the positive samples for training. In real-life scenarios, especially in medical datasets, we typically have a really small handcrafted dataset of positive samples.
- If we only use this dataset to train our model, it will overfit on this data.

Estimating the correct value of threshold, and using unlabeled data

- In the original setup, the threshold is a hyperparameter that we were allowed to set.
- Also, we are only using the positive samples for training. In real-life scenarios, especially in medical datasets, we typically have a really small handcrafted dataset of positive samples.
- If we only use this dataset to train our model, it will overfit on this data.
- Instead, it might be beneficial to include other unlabeled data to train our model.

Estimating the correct value of threshold, and using unlabeled data

- So now we try to modify our setup into the following setup: Given a partially labelled dataset containing only positive. More formally: Our dataset is $\{(x^{(i)}, t^{(i)}, y^{(i)})\}_{i=1}^{m}$, where $t^{(i)} \in \{0,1\}$ is the "true" label, and where

$$y^{(i)} = \begin{cases} 1 & x^{(i)} \text{ is labeled} \\ 0 & \text{otherwise.} \end{cases}$$

All labeled examples are positive, which is to say $p(t^{(i)} = 1 \mid y^{(i)} = 1) = 1$, but unlabeled examples may be positive or negative. Our task is to correctly predict the true labels of the datapoints.

Estimating the correct value of threshold, and using unlabeled data

- We can show that if we train on the entire dataset, $p(t^{(i)} = 1|x^{(i)}) = p(y^{(i)} = 1|x^{(i)})/p(y = 1|t = 1)$, (assuming the number of anomalous points is very less)

Estimating the correct value of threshold, and using unlabeled data

- We can show that if we train on the entire dataset, $p(t^{(i)} = 1|x^{(i)}) = p(y^{(i)} = 1|x^{(i)})/p(y = 1|t = 1)$, (assuming the number of anomalous points is very less)
- We can also show that:

Estimating the correct value of threshold, and using unlabeled data

- We can show that if we train on the entire dataset, $p(t^{(i)} = 1|x^{(i)}) = p(y^{(i)} = 1|x^{(i)})/p(y = 1|t = 1)$, (assuming the number of anomalous points is very less)
- We can also show that:

$$p(y = 1|t = 1) = \mathbb{E}_{v \in V^+}[h(v)]$$

where h(v) is the predicted logits, and V^+ is the subset of positively labelled points.

Estimating the correct value of threshold, and using unlabeled data

- We can show that if we train on the entire dataset, $p(t^{(i)} = 1|x^{(i)}) = p(y^{(i)} = 1|x^{(i)})/p(y = 1|t = 1)$, (assuming the number of anomalous points is very less)
- We can also show that:

$$p(y = 1|t = 1) = \mathbb{E}_{v \in V^+}[h(v)]$$

  where h(v) is the predicted logits, and V^+ is the subset of positively labelled points.
- So we can simply set the threshold to $\frac{\mathbb{E}_{v \in V^+}[h(v)]}{2}$ for predicting the probabilities.
- This method achieves an F-score of 0.8434 (Precision is 0.7411, Recall is 0.9785).

Outline

Summary

Method	Precision	Recall	F-score
DAGMM (Baseline)	0.9297	0.9442	0.9369
DAGMM with VAE	**0.9642**	**0.9461**	**0.9551**
DAGMM with VQVAE	**0.9663**	**0.9499**	**0.9581**
DAGMM with modified inference	**0.9505**	0.9200	0.9350
DAGMM with Laplacian	0.8532	0.7554	0.8014
DAGMM with modified Loss	0.4207	0.8556	0.5641
DAGMM with modified training setup*	0.7411	**0.9785**	0.8434

Table: Performance metrics using across all modifications

References I

Maxim.
Cs229 problem set 1 (autumn 2018), 2018.
Accessed: 2025-03-27.

B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen.
Deep autoencoding gaussian mixture model for unsupervised anomaly detection.
In *International Conference on Learning Representations*, 2018.