Paper Summary
CADE: Detecting and Explaining Concept Drift Samples for Security Applications

Sarthak Sarthak

June 2024

Problem Statement

The paper addresses the challenge of concept drift in machine learning models used for security applications, where the distribution of testing data deviates from the training data over time.

Motivation

- Learning-based models work under a “closed-world” assumption, expecting the testing data distribution to roughly match that of the training data
- Testing data distribution often shifts from original training data, leading to model failure in critical tasks such as detecting malware or network intrusions
- Retraining outdated models often needs labeling a large number of new samples and is difficult to decide when should we retrain
- Detection of drifting samples by checking prediction confidence or by computing non conformity measures lose effectiveness when sample (from a new class) is sparse with high dimensionality

Key Idea

To detect drifting samples by mapping data into a low-dimensional space using contrastive learning. Then identify and rank drifting samples based on their distance from existing classes and provide explanations by highlighting the features responsible for the drift. The mapping function is learned by contrasting samples to enlarge the distances between samples of different classes, while reducing the distance between samples in the same class.

Framework & Methodology

- Implemented using the Keras package with TensorFlow as the backend
- The encoder is a Multi-Layer Perceptron (MLP)
- Each hidden layer uses the ReLU activation function
- Adam optimizer is employed for faster convergence and training is conducted for 250 epochs
- An autoencoder consists of encoder f parameterised by θ and a decoder h parameterised by ϕ
- Encoder maps x to lower dimensional representation $z = f(x)$, Decoder reconstructs input from representation as $\hat{x} = h(z)$
- **Loss Function :**
  - **Reconstruction Loss :** Mean squared error between the original input x and it's reconstruction $\hat{x}$

    $$E_x[||x - \hat{x}||_2^2]$$

  - **Contrastive Loss :**

    $$E_{x_i,x_j}[(1 - y_{ij})d_{ij}^2 + y_{ij}(m - d_{ij})_+^2]$$

    * $y_{ij} = 1$ if samples are from different class and 0 if they're from same class
    * d_{ij} is the Euclidean distance between the latent space representations between $z_i = f(x_i; \theta)$ and $z_j = f(x_j; \theta)$
    * $(m - d_{ij})_+$ is the hinge loss term that enforces a margin m for dissimilar pairs, contributing to the loss only when the distance is within this margin
  - **Overall loss** $= \min_{\theta,\phi} E_x[||x-\hat{x}||_2^2] + \lambda E_{x_i,x_j}[(1-y_{ij})d_{ij}^2 + y_{ij}(m-d_{ij})_+^2]$
- A sample is flagged as out-of-distribution if its distance in latent space to all class centroids exceeds a predefined threshold (MAD threshold)
- Outputs a mask indicating feature importance, pinpointing the ones most relevant to identifying drifting samples

Contributions

- Introduced an effective method to detect drifting samples based on contrastive representation learning, complementing existing supervised learning based security applications to combat concept drift

- Illustrated the limitation of supervised explanation methods in explaining outlier samples and introduced a distance-based explanation method for this context
- Extensively evaluated the proposed methods with two applications (Android malware and network intrusion detection). Tested CADE with a security company's dataset to prove that it is effective in Real-World environment
- Released the code of CADE to support future research

Strengths

- Reduces the need for frequent retraining and extensive labeling, making it practical for real-time applications
- Provides understandable explanations for detected drifts
- Shows strong performance across multiple datasets, highlighting versatility

Weaknesses & Limitations

- While being effective in tested scenarios, its performance across other types of security applications or unforeseen drifts remain uncertain
- Integration of autoencoders would lead to computational complexity, this coupled with the task of fine tuning CADE would limit deployment in certain environments