Incorporating Privileged Information to Unsupervised Anomaly Detection

Shubhranshu Shekhar(✉) and Leman Akoglu

Heinz College of Information Systems and Public Policy,
Carnegie Mellon University, Pittsburgh, USA
{shubhras,lakoglu}@andrew.cmu.edu

Abstract. We introduce a new unsupervised anomaly detection ensemble called SPI which can harness *privileged* information—data available only for training examples but not for (future) test examples. Our ideas build on the Learning Using Privileged Information (LUPI) paradigm pioneered by Vapnik et al. [17,19], which we extend to unsupervised learning and in particular to anomaly detection. SPI (for Spotting anomalies with Privileged Information) constructs a number of frames/fragments of knowledge (i.e., density estimates) in the privileged space and *transfers* them to the anomaly scoring space through "imitation" functions that use only the partial information available for test examples. Our generalization of the LUPI paradigm to unsupervised anomaly detection shepherds the field in several key directions, including (*i*) *domain-knowledge-augmented detection* using expert annotations as PI, (*ii*) *fast detection* using computationally-demanding data as PI, and (*iii*) *early detection* using "historical future" data as PI. Through extensive experiments on simulated and real datasets, we show that augmenting privileged information to anomaly detection significantly improves detection performance. We also demonstrate the promise of SPI under all three settings (*i*–*iii*); with PI capturing expert knowledge, computationally-expensive features, and future data on three real world detection tasks. Code related to this paper is available at: http://www.andrew.cmu.edu/user/shubhras/SPI.

1 Introduction

Outlier detection in point-cloud data has been studied extensively [1]. In this work we consider a unique setting with a much sparser literature: the problem of augmenting privileged information into unsupervised anomaly detection. Simply put, privileged information (PI) is *additional* data/knowledge/information that is available only at the learning/model building phase for (subset of) training examples, which however is unavailable for (future) test examples.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-10925-7_6) contains supplementary material, which is available to authorized users.

M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 87–104, 2019.
https://doi.org/10.1007/978-3-030-10925-7_6

The LUPI Framework. Learning Using Privileged Information (LUPI) has been pioneered by Vapnik et al. first in the context of SVMs [17,19] (PI-incorporated SVM is named SVM+), later generalized to neural networks [18]. The setup involves an Intelligent (or non-trivial) Teacher at learning phase, who provides the Student with *privileged* information (like explanations, metaphors, etc.), denoted $\boldsymbol{x}_i^*$, about each training example $\boldsymbol{x}_i$, $i = 1 \ldots n$. The key point in this paradigm is that *privileged information is not available at the test phase* (when Student operates without guidance of Teacher). Therefore, the goal is to build models (in our case, detectors) that can leverage/incorporate such additional information but *yet, not depend on the availability of PI at test time.*

Example: The additional information $\boldsymbol{x}_i^*$'s belong to space X^* which is, generally speaking, different from space X. In other words, the feature spaces of vectors $\boldsymbol{x}_i^*$'s and $\boldsymbol{x}_i$'s do not overlap. As an example, consider the task of identifying cancerous biopsy images. Here the images are in pixel space X. Suppose that there is an Intelligent Teacher that can recognize patterns in such images relevant to cancer. Looking at a biopsy image, Teacher can provide a description like "Aggressive proliferation of A-cells into B-cells" or "Absence of any dynamic". Note that such descriptions are in a specialized language space X^*, different from pixel space X. Further, they would be available only for a set of examples and not when the model is to operate autonomously in the future.

LUPI's Advantages: LUPI has been shown to (*i*) improve rate of convergence for learning, i.e., require asymptotically fewer examples to learn [19], as well as (*ii*) improve accuracy, when one can learn a model in space X^* that is not much worse than the best model in space X (i.e., PI is intelligent/non-trivial) [18]. Motivated by these advantages, LUPI has been applied to a number of problems from action recognition [13] to risk modeling [14] (expanded in Sect. 5). However, the focus of all such work has mainly been on *supervised* learning.

LUPI for Anomaly Detection. The only (perhaps straightforward) extension of LUPI to unsupervised anomaly detection has been introduced recently, generalizing SVM+ to the One-Class SVM (namely OC-SVM+) [2] for malware and bot detection. The issue is that OC-SVM is not a reliable detector since it assumes that normal points can be separated from origin in a single hyperball—experiments on numerous benchmark datasets with ground truth by Emmott et al. that compared popular anomaly detection algorithms find that OC-SVM ranks at the bottom (Table 1, pg. 4 [6]; also see our results in Sect. 4). We note that the top performer in [6] is the Isolation Forest (iForest) algorithm [11], an ensemble of randomized trees.

Our Contributions: Motivated by LUPI's potential value to learning and the scarcity in the literature of its generalization to anomaly detection, we propose a new technique called SPI (pronounced 'spy'), for Spotting anomalies with Privileged Information. Our work *bridges the gap (for the first time) between LUPI and unsupervised ensemble based anomaly detection* that is considered state-of-the-art [6]. We summarize our main contributions as follows.

- **Study of LUPI for anomaly detection:** We analyze how LUPI can benefit anomaly detection, not only when PI is truly unavailable at test time (as in traditional setup) but also when PI is strategically and willingly avoided at test time. We argue that data/information that incurs overhead on resources ($$$/storage/battery/etc.), timeliness, or vulnerability, if designated as PI, can enable resource-frugal, early, and preventive detection (expanded in Sect. 2).
- **PI-incorporated detection algorithm:** We show how to incorporate PI into ensemble based detectors and propose SPI, which constructs frames/fragments of knowledge (specifically, density estimates) in the privileged space (X^*) and *transfers* them to the anomaly scoring space (X) through "imitation" functions that use only the partial information available for test examples. To the best of our knowledge, ours is the first attempt to leveraging PI for improving the state-of-the-art *ensemble methods* for anomaly detection within an *unsupervised* LUPI framework. Moreover, while SPI augments PI within the tree-ensemble detector iForest [11], our solution can easily be applied to any other ensemble based detector (Sect. 3).
- **Applications:** Besides extensive simulation experiments, we employ SPI on three real-world case studies where PI respectively captures (*i*) expert knowledge, (*ii*) computationally-expensive features, and (*iii*) "historical future" data, which demonstrate the benefits that PI can unlock for anomaly detection in terms of accuracy, speed, and detection latency (Sect. 4).

Reproducibility: Implementation of SPI and real world datasets used in experiments are open-sourced at http://www.andrew.cmu.edu/user/shubhras/SPI.

2 Motivation: How Can LUPI Benefit Anomaly Detection?

The implications of the LUPI paradigm for anomaly detection is particularly exciting. Here, we discuss a number of detection scenarios and demonstrate that LUPI unlocks advantages for anomaly detection problems in multiple aspects.

In the original LUPI framework [19], privileged information (hereafter PI) is defined as data that is available *only* at training stage for training examples but *unavailable at test time* for test examples. Several anomaly detection scenarios admit this definition directly. Interestingly, PI can also be specified as *strategically "unavailable"* for anomaly detection. That is, one can willingly avoid using certain data at test time (while incorporating such data into detection models at train phase[1]) in order to achieve resource efficiency, speed, and robustness. We organize detection scenarios into two with PI as (truly) Unavailable vs. Strategic, and elaborate with examples below. Table 1 gives a summary.

[1] Note that training phase in anomaly detection does not involve the use of any labels.

Table 1. Types of data used in anomaly detection with various overhead on resources ($$$, storage, battery, etc.), timeliness, and/or risk, *if used as privileged information can enable resource-frugal, early, as well as preventive detection.*

Properties vs. type of privileged info	Unavailable vs. Strategic	Need Resources	Cause Delay	Incur Risk
1. "historical future" data	U	n/a	n/a	n/a
2. after-the-fact data	U	n/a	n/a	n/a
3. advanced technical data	U	n/a	n/a	n/a
4. restricted-access data	U, S	✓		
5. expert knowledge	U, S	✓	✓	
6. compute-heavy data	S	✓	✓	
7. unsafe-to-collect data	S		✓	✓
8. easy-target-to-tamper data	S			✓

Unavailable PI: This setting includes typical scenarios, where PI is (truly) unknown for test examples.

1. *"historical future" data*: When training an anomaly detection model with offline/historical data that is over time (e.g., temporal features), one may use values both before *and after* time t while creating an example for each t. Such data is PI; not available when the model is deployed to operate in real-time.

2. *after-the-fact data*: In malware detection, the goal is to detect before it gets hold of and harms the system. One may have historical data for some (training) examples from past exposures, including measurements of system variables (number of disk/port read/writes, CPU usage, etc.). Such after-the-exposure measurements can be incorporated as PI.

3. *advanced technical data*: This includes scenarios where some (training) examples are well-understood but those to be detected are simply unknown. For example, the expected behavior of various types of apps on a system may be common domain knowledge that can be converted to PI, but such knowledge may not (yet) be available for new-coming apps.

Strategic PI: Strategic scenarios involve PI that can in principle be acquired but is willingly avoided at test time to achieve gains in resources, time, or risk.

4. *restricted-access data*: One may want to build models that do not assume access to private data or intellectual property at test time, such as source code (for apps or executables), *even if* they could be acquired through resources. Such information can also be truly unavailable, e.g. encrypted within the software.

5. *expert knowledge*: Annotations about some training examples may be available from experts, which are truly unavailable at test time. One could also strategically choose to avoid expert involvement at test time, which (a) may be costly to obtain and/or (b) cause significant delay, especially for real-time detection.

6. *compute-heavy data*: One may strategically choose not to rely on features that are computationally expensive to obtain, especially in real-time detection, but rather use such data as PI (which can be extracted offline at training phase). Such features not only cause delay but also require compute resources (which e.g., may drain batteries in detecting malware apps on cellphones).

7. *unsafe-to-collect data*: This involves cases where collecting PI at test time is unsafe/dangerous. For example, the slower a drone moves to capture high-resolution (privileged) images for surveillance, not only it causes delay but more importantly, the more susceptible it becomes to be taken down.

8. *easy-target-to-tamper data*: Finally, one may want to avoid relying on features that are easy for adversaries to tamper with. Examples to those features include self-reported data (like age, location, etc.). Such data may be available reliably for some training examples and can be used as PI.

In short, by strategically designating PI one can achieve resource, timeliness, and robustness gains for various anomaly detection tasks. Designating features that need resources as PI → allow resource-frugal ("lazy") detection; features that cause delay as PI → allow early/speedy detection; and designating features that incur vulnerability as PI → allow preventive and more robust detection.

In this subsection, we laid out a long list of scenarios that make LUPI-based learning particularly attractive for anomaly detection. In our experiments (Sect. 4) we demonstrate its premise for scenarios 1., 5. and 6. above using three real world datasets, while leaving others as what we believe interesting future investigations.

3 Privileged Info-Augmented Anomaly Detection

The Learning Setting. Formally, the input for the anomaly detection model at learning phase are tuples of the form

$$\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{x}_1^*), (\boldsymbol{x}_2, \boldsymbol{x}_2^*), \ldots, (\boldsymbol{x}_n, \boldsymbol{x}_n^*)\},$$

where $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^d) \in X$ and $\boldsymbol{x}_i^* = (x_i^{*1}, \ldots, x_i^{*p}) \in X^*$. Note that this is an unsupervised learning setting where label information, i.e., y_i's are not available. The privileged information is represented as a feature vector $\boldsymbol{x}^* \in \mathbb{R}^p$ that is in space X^*, which is *additional to and different from* the feature space X in which the primary information is represented as a feature vector $\boldsymbol{x} \in \mathbb{R}^d$.

The important distinction from the traditional anomaly detection setting is that the input to the (trained) detector at testing phase are feature vectors

$$\{\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}\}.$$

That is, the (future) test examples do not carry any privileged information. The anomaly detection model is to score the incoming/test examples and make decisions solely based on the primary features $\boldsymbol{x} \in X$.

In this text, we refer to space X^* as the *privileged space* and to X as the *decision space*. Here, a key assumption is that the information in the privileged space

is intelligent/nontrivial, that is, it allows to create models $f^*(\boldsymbol{x}^*)$ that detect anomalies with vectors $\boldsymbol{x}^*$ corresponding to vectors $\boldsymbol{x}$ with higher accuracy than models $f(\boldsymbol{x})$. As a result, the main question that arises which we address in this work is: "how can one use the knowledge of the information in space X^* to improve the performance of the desired model $f(\boldsymbol{x})$ in space X?".

In what follows, we present a first-cut attempt to the problem that is a natural knowledge transfer between the two feature spaces (called FT for feature transfer). We then lay out the shortcomings of such an attempt, and present our proposed solution SPI. We compare to FT (and other baselines) in experiments.

3.1 First Attempt: Incorporating PI by Transfer of Features

A natural attempt to learning under privileged information that is unavailable for test examples is to treat the task as a *missing data problem*. Then, typical techniques for data imputation can be employed where missing (privileged) features are replaced with their predictions from the available (primary) features.

In this scheme, one simply maps vectors $\boldsymbol{x} \in X$ into vectors $\boldsymbol{x}^* \in X^*$ and then builds a detector model in the transformed space. The goal is to find the transformation of vectors $\boldsymbol{x} = (x^1, \ldots, x^d)$ into vectors $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_p(\boldsymbol{x}))$ that minimizes the expected risk given as

$$R(\boldsymbol{\phi}) = \sum_{j=1}^{p} \min_{\phi_j} \int (x^{*j} - \phi_j(\boldsymbol{x}))^2 p(x^{*j}, \boldsymbol{x}) dx^{*j} d\boldsymbol{x}, \tag{1}$$

where $p(x^{*j}, \boldsymbol{x})$ is the joint probability of coordinate x^{*j} and vector $\boldsymbol{x}$, and functions $\phi_j(\boldsymbol{x})$ are defined by p regressors.

Here, one could construct approximations to functions $\phi_j(\boldsymbol{x})$, $j = \{1, \ldots, p\}$ by solving p regression estimation problems based on the training examples

$$(\boldsymbol{x}_1, x_1^{*j}), \ldots, (\boldsymbol{x}_n, x_n^{*j}), \quad j = 1, \ldots, p,$$

where $\boldsymbol{x}_i$'s are input to each regression ϕ_j and the jth coordinate of the corresponding vector $\boldsymbol{x}_i^*$, i.e. x_i^{*j}'s are treated as the output, by minimizing the regularized empirical loss functional

$$R(\phi_j) = \min_{\phi_j} \sum_{i=1}^{n} (x_i^{*j} - \phi_j(\boldsymbol{x}_i))^2 + \lambda_j \text{penalty}(\phi_j), \quad j = 1, \ldots, p. \tag{2}$$

Having estimated the transfer functions $\hat{\phi}_j$'s (using linear or non-linear regression techniques), one can then learn any desired anomaly detector $f(\hat{\boldsymbol{\phi}}(\boldsymbol{x}))$ using the training examples, which concludes the learning phase. Note that the detector does not require access to privileged features $\boldsymbol{x}^*$ and can be employed solely on primary features $\boldsymbol{x}$ of the test examples $i = n+1, \ldots, m$.

3.2 Proposed SPI: Incorporating PI by Transfer of Decisions

Treating PI as missing data and predicting $\boldsymbol{x}^*$ from $\boldsymbol{x}$ could be a difficult task, when privileged features are complex and high dimensional (i.e., p is large). Provided $f^*(\boldsymbol{x}^*)$ is an accurate detection model, a more direct goal would be to *mimic its decisions*—the scores that f^* assigns to the training examples. Mapping *data* between two spaces, as compared to *decisions*, would be attempting to solve a more general problem, that is likely harder and unnecessarily wasteful.

The general idea behind transferring decisions/knowledge (instead of data) is to identify a small number of elements in the privileged space X^* that well-approximate the function $f^*(\boldsymbol{x}^*)$, and then try to transfer them to the decision space—through the approximation of those elements in space X. This is the knowledge transfer mechanism in LUPI by Vapnik and Izmailov [17]. They illustrated this mechanism for the (supervised) SVM classifier. We generalize this concept to unsupervised anomaly detection.

The knowledge transfer mechanism uses three building blocks of knowledge representation in AI, as listed in Table 2. We first review this concept for SVMs, followed by our proposed SPI. While SPI is clearly different in terms of the task it is addressing as well as in its approach, as we will show, it is inspired by and builds on the same fundamental mechanism.

Table 2. Three building blocks of knowledge representation in artificial intelligence, in context of SVM-LUPI for classification [17] and SPI for anomaly detection [this paper].

	SVM-LUPI	SPI (Proposed)
1. Fundamental elements of knowledge	Support vectors	Isolation trees
2. Frames (fragments) of the knowledge	Kernel functions	Tree anomaly scores
3. Structural connections of the frames	Weighted sum	Weighted sum (by L2R)

Knowledge Transfer for SVM: The *fundamental elements* of knowledge in the SVM classifier are the support vectors. In this scheme, one constructs two SVMs; one in X space and another in X^* space. Without loss of generality, let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ be the support vectors of SVM solution in space X and $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_{t^*}^*$ be the support vectors of SVM solution in space X^*, where t and t^* are the respective number of support vectors.

The decision rule f^* in space X^* (which one aims to mimic) has the form

$$f^*(\boldsymbol{x}^*) = \sum_{k=1}^{t^*} y_k \alpha_k^* K^*(\boldsymbol{x}_k^*, \boldsymbol{x}^*) + b^*, \tag{3}$$

where $K^*(\boldsymbol{x}_k^*, \boldsymbol{x}^*)$ is the kernel function of similarity between support vector $\boldsymbol{x}_k^*$ and vector $\boldsymbol{x}^* \in X^*$, also referred as the *frames* (or *fragments*) of knowledge. Equation (3) depicts the *structural connection* of these fragments, which is a weighted sum with learned weights α_k^*'s.

The goal is to approximate each fragment of knowledge $K^*(\boldsymbol{x}_k^*, \boldsymbol{x}^*)$, $k = 1, \ldots, t^*$ in X^* using the fragments of knowledge in X; i.e., the t kernel functions $K(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_t, \boldsymbol{x})$ of the SVM trained in X. To this end, one maps t-dimensional vectors $\boldsymbol{z} = (K(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, K(\boldsymbol{x}_t, \boldsymbol{x})) \in Z$ into t^*-dimensional vectors $\boldsymbol{z}^* = (K^*(\boldsymbol{x}_1^*, \boldsymbol{x}^*), \ldots, K^*(\boldsymbol{x}_{t^*}^*, \boldsymbol{x}^*)) \in Z^*$ through t^* regression estimation problems. That is, the goal is to find regressors $\phi_1(\boldsymbol{z}), \ldots, \phi_{t^*}(\boldsymbol{z})$ in X such that

$$\phi_k(\boldsymbol{z}_i) \approx K^*(\boldsymbol{x}_k^*, \boldsymbol{x}_i^*), \quad k = 1, \ldots, t^* \tag{4}$$

for all training examples $i = 1, \ldots, n$. For each $k = 1, \ldots, t^*$, one can construct the approximation to function ϕ_k by training a regression on the data

$$\{(\boldsymbol{z}_1, K^*(\boldsymbol{x}_k^*, \boldsymbol{x}_1^*)), \ \ldots, \ (\boldsymbol{z}_n, K^*(\boldsymbol{x}_k^*, \boldsymbol{x}_n^*))\}, \ k = 1, \ldots, t^*,$$

where we regress vectors $\boldsymbol{z}_i$'s onto scalar output $K^*(\boldsymbol{x}_k^*, \boldsymbol{x}_i^*)$'s to obtain $\hat{\phi}_k$.

For the prediction of a test example $\boldsymbol{x}$, one can then replace each $K^*(\boldsymbol{x}_k^*, \boldsymbol{x}^*)$ in Eq. (3) (which requires privileged features $\boldsymbol{x}^*$) with $\hat{\phi}_k(\boldsymbol{z})$ (which mimics it, using only the primary features $\boldsymbol{x}$—to be exact, by first transforming $\boldsymbol{x}$ into $\boldsymbol{z}$ through the frames $K(\boldsymbol{x}_j, \boldsymbol{x}), j = 1, \ldots, t$ in the X space).

Knowledge Transfer for SPI**:** In contrast to mapping of features from space X to space X^*, knowledge transfer of decisions maps space Z to Z^* in which fragments of knowledge are represented. Next, we show how to generalize these ideas to anomaly detection with no label supervision. Figure 1 shows an overview.

To this end, we utilize a state-of-the-art ensemble technique for anomaly detection, called Isolation Forest [11] (hereafters IF, for short), which builds a set of extremely randomized trees. In essence, each tree approximates density in a random feature subspace and anomalousness of a point is quantified by the sum of such partial estimates across all trees.

In this setting, one can think of the individual trees in the ensemble to constitute the *fundamental elements* and the partial density estimates (i.e., individual anomaly scores from trees) to constitute the *fragments* of knowledge, where the structural connection of the fragments is achieved by an unweighted sum.

Similar to the scheme with SVMs, we construct two IFs; one in X space and another in X^* space. Let $\mathcal{T} = T_1, \ldots, T_t$ denote the trees in the ensemble in X and $\mathcal{T}^* = T_1^*, \ldots, T_{t^*}^*$ the trees in the ensemble in X^*, where t and t^* are the respective number of trees (prespecified by the user, typically a few 100s). Further, let $S^*(T_k^*, \boldsymbol{x}^*)$ denote the anomaly score estimated by tree T_k^* for a given $\boldsymbol{x}^*$ (the lower the more anomalous; refer to [11] for details of the scoring). $S(T_k, \boldsymbol{x})$ is defined similarly. Then, the anomaly score s^* for a point $\boldsymbol{x}^*$ in space X^* (which we aim to mimic) is written as

$$s^*(\boldsymbol{x}^*) = \sum_{k=1}^{t^*} S^*(T_k^*, \boldsymbol{x}^*), \tag{5}$$

which is analogous to Eq. (3). To mimic/approximate each fragment of knowledge $S^*(T_k^*, \boldsymbol{x}^*)$, $k = 1, \ldots, t^*$ in X^* using the fragments of knowledge in X; i.e., the

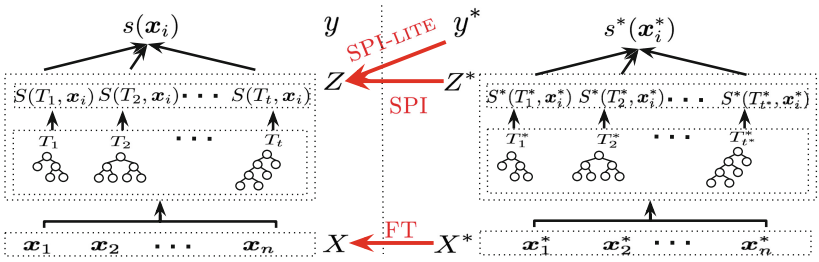


Fig. 1. Anomaly detection with PI illustrated. FT maps *data* between spaces (Sect. 3.1) whereas SPI (and "light" version SPI-LITE) mimic *decisions* (Sect. 3.2).

t scores for $\boldsymbol{x}$: $S(T_1,\boldsymbol{x}),\ldots,S(T_t,\boldsymbol{x})$ of the IF trained in X, we estimate t^* regressors $\phi_1(\boldsymbol{z}),\ldots,\phi_{t^*}(\boldsymbol{z})$ in X such that

$$\phi_k(\boldsymbol{z}_i) \approx S^*(T_k^*,\boldsymbol{x}_i^*), \quad k=1,\ldots,t^* \tag{6}$$

for all training examples $i=1,\ldots,n$, where $\boldsymbol{z}_i = (S(T_1,\boldsymbol{x}_i),\ldots,S(T_t,\boldsymbol{x}_i))$. Simply put, each $\hat{\phi}_k$ is an approximate mapping of all the t scores from the ensemble $\mathcal{T}$ in X to an individual score (fragment of knowledge) by tree T_k^* of the ensemble $\mathcal{T}^*$ in X^*. In practice, we learn a mapping from the leaves rather than the trees of $\mathcal{T}$ for a more granular mapping. Specifically, we construct vectors $\boldsymbol{z}_i = (\boldsymbol{z}'_{i1},\ldots,\boldsymbol{z}'_{it})$ where each $\boldsymbol{z}'_{ik}$ is a size ℓ_k vector in which the value at index $\text{leaf}(T_k,\boldsymbol{x}_i)$ is set to $S(T_k,\boldsymbol{x}_i)$ and other entries to zero. Here, ℓ_k denotes the number of leaves in tree T_k and $\text{leaf}(\cdot)$ returns the index of the leaf that $\boldsymbol{x}_i$ falls into in the corresponding tree (note that $\boldsymbol{x}_i$ belongs to exactly one leaf of any tree, since the trees partition the feature space).

SPI-lite: A "light" version. We note that instead of mimicking each individual fragment of knowledge $S^*(T_k^*,\boldsymbol{x}^*)$'s, one could also directly mimic the "final decision" $s^*(\boldsymbol{x}^*)$. To this end, we also introduce SPI-LITE, which estimates a *single* regressor $\phi(\boldsymbol{z}_i) \approx s^*(\boldsymbol{x}_i^*)$ for $i=1,\ldots,n$ (also see Fig. 1). We compare SPI and SPI-LITE empirically in Sect. 4.

Learning to Rank (L2R) Like in X^*: An important challenge in learning to accurately mimic the scores s^*'s in Eq. (5) is to make sure that the regressors ϕ_k's are very accurate in their approximations in Eq. (6). Even then, it is hard to guarantee that the final ranking of points by $\sum_{k=1}^{t^*}\hat{\phi}_k(\boldsymbol{z}_i)$ would reflect their ranking by $s^*(\boldsymbol{x}_i^*)$. Our ultimate goal, after all, is to *mimic the ranking* of the ensemble in X^* space since anomaly detection is a ranking problem at its heart.

To this end, we set up an additional pairwise learning to rank objective as follows. Let us denote by $\boldsymbol{\phi}_i = (\hat{\phi}_1(\boldsymbol{z}_i),\ldots,\hat{\phi}_{t^*}(\boldsymbol{z}_i))$ the t^*-dimensional vector of estimated knowledge fragments for each training example i. For each pair of training examples, we create a tuple of the form $((\boldsymbol{\phi}_i,\boldsymbol{\phi}_j),p_{ij}^*)$ where

$$p_{ij}^* = P(s_i^* < s_j^*) = \sigma(-(s_i^* - s_j^*)), \tag{7}$$

Algorithm 1. SPI-TRAIN: Incorporating PI to Unsupervised Anomaly Detector

Input: training examples $\{(\boldsymbol{x}_1, \boldsymbol{x}_1^*), \ldots, (\boldsymbol{x}_n, \boldsymbol{x}_n^*)\}$
Output: detection model (ensemble-of-trees) $\mathcal{T}$ in X space; regressors $\hat{\phi}_k$'s, $k = 1, \ldots, t^*$; $\boldsymbol{\beta}$ (or $\boldsymbol{\gamma}$ for kernelized L2R)
1: Learn t^* isolation trees $\mathcal{T}^* = \{T_1^*, \ldots, T_{t^*}^*\}$ on $\boldsymbol{x}_i^*$'s $i = 1, \ldots, n$
2: Learn t isolation trees $\mathcal{T} = \{T_1, \ldots, T_t\}$ on $\boldsymbol{x}_i$'s $i = 1, \ldots, n$
3: Construct leaf score vectors $\boldsymbol{z}_i$'s, $i = 1, \ldots, n$, based on $\mathcal{T}$
4: **for each** $k = 1, \ldots, t^*$ **do**
5: Learn regressor $\hat{\phi}_k$ of $\boldsymbol{z}_i$'s onto $S^*(T_k^*, \boldsymbol{x}_i^*)$'s
6: Obtain $\boldsymbol{\beta}$ by optimizing C in (9) (or $\boldsymbol{\gamma}$ for kernelized C_ψ)
7: **end for**

Algorithm 2. SPI-TEST: PI-Augmented Unsupervised Anomaly Detection

Input: test examples $\{\boldsymbol{x}_{n+1}, \ldots, \boldsymbol{x}_{n+m}\}$; $\mathcal{T}$, $\hat{\phi}_k$'s $k = 1, \ldots, t^*$, $\boldsymbol{\beta}$ (or $\boldsymbol{\gamma}$ if kernelized)
Output: estimated anomaly scores $\{s_{n+1}, \ldots, s_{n+m}\}$ for all test examples
1: **for each** test example $\boldsymbol{x}_e$, $e = n+1, \ldots, n+m$ **do**
2: Construct leaf score vector $\boldsymbol{z}_e = (\boldsymbol{z}'_{e1}, \ldots, \boldsymbol{z}'_{et})$ where entry in each $\boldsymbol{z}'_{ek}$ for index leaf$(T_k, \boldsymbol{x}_e)$ is set to $S(T_k, \boldsymbol{x}_e)$ and to 0 o.w., for $k = 1, \ldots, t$
3: Construct $\boldsymbol{\phi}_e = (\hat{\phi}_1(\boldsymbol{z}_e), \ldots, \hat{\phi}_{t^*}(\boldsymbol{z}_e))$
4: Estimate anomaly score as $s_e = \boldsymbol{\beta}\boldsymbol{\phi}_e^T$ (or $s_e = \sum_{l=1}^n \gamma_l K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_e)$ if kernelized)
5: **end for**

which is the probability that i is ranked ahead of j by anomalousness in X^* space (recall that lower s^* is more anomalous), where $\sigma(v) = 1/(1+e^{-v})$ is the sigmoid function. Notice that the larger the gap between the anomaly scores of i and j, the larger this probability gets (i.e., more surely i ranks above j).

Given the training pair tuples above, our goal of learning-to-rank is to estimate $\boldsymbol{\beta} \in \mathbb{R}^{t^*}$, such that

$$p_{ij} = \sigma(\Delta_{ij}) = \sigma(\boldsymbol{\beta}\boldsymbol{\phi}_i^T - \boldsymbol{\beta}\boldsymbol{\phi}_j^T) = \sigma(-\hat{s_i^*} + \hat{s_j^*}) \approx p_{ij}^*, \quad \forall i, j \in \{1, \ldots, n\}. \quad (8)$$

We then utilize the cross entropy as our cost function over all (i, j) pairs, as

$$\min_{\boldsymbol{\beta}} \; C = \sum_{(i,j)} -p_{ij}^* \log(p_{ij}) - (1 - p_{ij}^*) \log(1 - p_{ij}) = \sum_{(i,j)} -p_{ij}^* \Delta_{ij} + \log(1 + e^{\Delta_{ij}}) \quad (9)$$

where p_{ij}^*'s are given as input to the learning as specified in Eq. (7) and p_{ij} is denoted in Eq. (8) and is parameterized by $\boldsymbol{\beta}$ that is to be estimated.

The objective function in (9) is convex and can be solved via a gradient-based optimization, where $\frac{\mathrm{d}C}{\mathrm{d}\boldsymbol{\beta}} = \sum_{(i,j)} (p_{ij} - p_{ij}^*)(\boldsymbol{\phi}_i - \boldsymbol{\phi}_j)$ (details omitted for brevity). More importantly, in case the linear mapping $s_i^* \approx \boldsymbol{\beta}\boldsymbol{\phi}_i^T$ is not sufficiently accurate to capture the desired pairwise rankings, the objective can be *kernelized* to learn a *non-linear* mapping that is likely more accurate. The idea is to write $\boldsymbol{\beta}_\psi = \sum_{l=1}^n \gamma_l \psi(\boldsymbol{\phi}_l)$ (in the transformed space) as a weighted linear combination of (transformed) training examples, for feature transformation function $\psi(\cdot)$ and

parameter vector $\boldsymbol{\gamma} \in \mathbb{R}^n$ to be estimated. Then, Δ_{ij} in objective (9) in the transformed space can be written as

$$\Delta_{ij} = \sum_{l=1}^{n} \gamma_l [\psi(\boldsymbol{\phi}_l)\psi(\boldsymbol{\phi}_i)^T - \psi(\boldsymbol{\phi}_l)\psi(\boldsymbol{\phi}_j)^T] = \sum_{l=1}^{n} \gamma_l [K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_i) - K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_j)]. \quad (10)$$

The kernelized objective, denoted C_ψ, can also be solved through gradient-based optimization where we can show partial derivatives (w.r.t. each γ_l) to be equal to $\frac{\partial C_\psi}{\partial \gamma_l} = \sum_{(i,j)} (p_{ij} - p^*_{ij})[K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_i) - K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_j)]$. Given the estimated γ_l's, prediction of score is done by $\sum_{l=1}^{n} \gamma_l K(\boldsymbol{\phi}_l, \boldsymbol{\phi}_e)$ for any (test) example e.

The SPI **Algorithm:** We outline the steps of SPI for both training and testing (i.e., detection) in Algorithms 1 and 2, respectively. Note that the test-time detection no longer relies on the availability of privileged features for the test examples, but yet be able to leverage/incorporate them through its training.

4 Experiments

We design experiments to evaluate our methods in two different settings:

1. **Benchmark Evaluation:** We show the effectiveness of augmenting PI (see Table 3) on 17 publicly available benchmark datasets.[2]
2. **Real-world Use Cases:** We conduct experiments on LingSpam[3] and BotOrNot[4] datasets to show that (*i*) domain-expert knowledge as PI improves spam detection, (*ii*) compute-expensive PI enables fast detection at test time, and (*iii*) "historical future" PI allows early detection of bots.

Baselines. We compare both SPI and SPI-LITE to the following baselines:

1. IF(X-only): Isolation Forest [11] serves as a simple baseline that operates solely in decision space X. PI is not used neither for modeling nor detection.
2. OC-SVM+ (PI-incorporated): OC+ for short, is an extension of (unsupervised) One-Class SVM that incorporates PI as introduced in [2].
3. FT(PI-incorporated): This is the direct feature transfer method that incorporates PI by learning a mapping $X \rightarrow X^*$ as we introduced in Sect. 3.1.

* IF* (X^*-only): IF that operates in X^* space. We report performance by IF* only for reference, since PI is unavailable at test time.

4.1 Benchmark Evaluation

The benchmark datasets do not have an explicit PI representation. Therefore, in our experiments we introduce PI as explained below.

[2] http://agents.fel.cvut.cz/stegodata/Loda.zip.
[3] http://csmining.org/index.php/ling-spam-datasets.html.
[4] https://botometer.iuni.iu.edu/bot-repository/datasets/caverlee-2011/caverlee-2011.zip.

Table 3. Mean Average Precision (MAP) on benchmark datasets (avg'ed over 5 runs) for $\gamma = 0.7$. Numbers in parentheses indicate rank of each algorithm on each dataset. IF* (for reference only) reports MAP in the X^* space.

Datasets	$p+d$	n	IF	OC+	FT	SPI-LITE	SPI	IF*
breast-cancer	30	357	0.1279 (4)	0.0935 (6)	0.0974 (5)	0.4574 (3)	**0.5746** (2)	0.6773 (1)
ionosphere	33	225	0.0519 (4)	**0.2914** (1)	0.0590 (3)	0.0512 (5)	0.0470 (6)	0.0905 (2)
letter-recognition	617	4197	0.0889 (6)	0.1473 (4)	0.0908 (5)	0.3799 (3)	**0.6413** (2)	0.9662 (1)
multiple-features	649	1200	0.1609 (5)	0.1271 (6)	0.2044 (4)	0.6589 (3)	**0.8548** (2)	1.0000 (1)
wall-following-robot	24	2923	0.1946 (5)	0.2172 (4)	0.1848 (6)	0.4331 (3)	**0.5987** (2)	0.7538 (1)
cardiotocography	27	1831	0.2669 (5)	0.6107 (4)	0.2552 (6)	0.6609 (3)	**0.6946** (2)	0.8081 (1)
isolet	617	4497	0.1533 (5)	0.1561 (4)	0.1303 (6)	0.5084 (3)	**0.7124** (2)	0.9691 (1)
libras	90	216	0.1368 (5)	0.4479 (4)	0.0585 (6)	0.5175 (3)	**0.6806** (2)	1.0000 (1)
parkinsons	22	147	0.0701 (6)	0.0964 (4)	0.0714 (5)	0.1556 (3)	**0.1976** (1)	0.1778 (2)
statlog-satimage	36	3594	0.2108 (6)	0.5347 (5)	0.5804 (4)	0.9167 (3)	**0.9480** (2)	0.9942 (1)
gisette	4971	3500	0.1231 (4)	0.0814 (6)	0.0977 (5)	0.5593 (3)	**0.8769** (2)	0.9997 (1)
waveform-1	21	3304	0.1322 (4)	0.1481 (3)	0.0841 (6)	0.1234 (5)	**0.1556** (2)	0.4877 (1)
madelon	500	1300	0.7562 (5)	0.1167 (6)	**0.9973** (2)	0.9233 (4)	0.9925 (3)	1.0000 (1)
synthetic-control	60	400	0.3207 (6)	0.7889 (4)	0.6870 (5)	0.8103 (3)	**0.8539** (2)	0.9889 (1)
waveform-2	21	3304	0.1271 (5)	**0.2828** (2)	0.1014 (6)	0.1778 (3)	0.1772 (4)	0.2944 (1)
statlog-vehicle	18	629	0.1137 (6)	0.3146 (5)	0.6326 (4)	0.6561 (3)	**0.7336** (2)	1.0000 (1)
statlog-segment	18	1320	0.1250 (6)	0.2323 (4)	0.1868 (5)	0.3304 (3)	**0.3875** (2)	0.7399 (1)
(Average Rank)			(5.11)	(4.23)	(4.88)	(3.29)	(2.35)	(1.11)

Generating Privileged Representation. For each dataset, we introduce PI by perturbing normal observations. We designate a small random fraction (= 0.1) of n normal data points as anomalies. Then, we randomly select a subset of p attributes and add zero-mean Gaussian noise to the designated anomalies along the selected subset of attributes with matching variances of the selected features. The p selected features represent PI since anomalies stand-out in this subspace due to added noise, while the rest of the d attributes represent X space. Using normal observations allows us to control for features that could be used as PI. Thus we discard the actual anomalies from these datasets where PI is unknown.

We construct 4 versions per dataset with varying fraction γ of perturbed features (PI) retained in X^* space. In particular, each set has γp features in X^*, and $(1-\gamma)p + d$ features in X for $\gamma \in \{0.9, 0.7, 0.5, 0.3\}$.

Results. We report the results on perturbed datasets with $\gamma = 0.7$[5] as fraction of features retained in space X^*. Table 3 reports mean Average Precision (area under the precision-recall curve) against 17 datasets for different methods. The results are averaged across 5 independent runs on stratified train-test splits.

[5] The results with $\gamma \in \{0.9, 0.5, 0.3\}$ are similar and reported in the supplementary material available at http://www.andrew.cmu.edu/user/shubhras/SPI.

Our SPI outperforms competition in detection performance in most of the datasets. To compare the methods statistically, we use the non-parametric Friedman test [5] based on the average ranks. Table 3 reports the ranks (in parentheses) on each dataset as well as the average ranks. With p-value $= 2.16 \times 10^{-11}$, we reject the null hypothesis that all the methods are equivalent using Friedman test. We proceed with Nemenyi post-hoc test to compare the algorithms pairwise and to find out the ones that differ significantly. The test identifies performance of two algorithms to be significantly different if their average ranks differ by at least the "critical difference" (CD). In our case, comparing 6 methods on 17 datasets at significance level $\alpha = 0.05$, CD $= 1.82$.

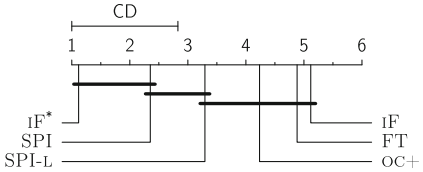
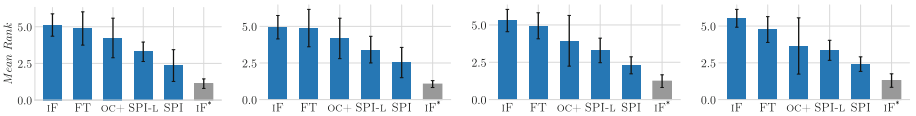


Fig. 2. Average rank of algorithms (w.r.t. MAP) and comparison by the Nemenyi test. Groups of methods not significantly different (at p-val $= 0.05$) are connected with horizontal lines. CD depicts critical distance required to reject equivalence. Note that SPI is significantly better than the baselines.

Results of the post-hoc test are summarized through a graphical representation in Fig. 2. We find that SPI is significantly better than all the baselines. We also notice that SPI has no significant difference from IF* which uses PI at test time, demonstrating its effectiveness in augmenting PI. While all the baselines are comparable to SPI-LITE, its average rank is better (also see last row in Table 3), followed by other PI-incorporated detectors, and lastly IF with no PI.

Average Precision (AP) is a widely-accepted metric to quantify overall performance of ranking methods like anomaly detectors. We also report average rank of the algorithms against other popular metrics including AUC of ROC curve, NDCG@10 and PRECISION@10 in Fig. 3. Notice that the results are consistent across measures, SPI and SPI-LITE performing among the best.



(a) MAP (b) AUC (c) NDCG@10 (d) PRECISION@10

Fig. 3. SPI and SPI-LITE outperform competition w.r.t. different evaluation metrics. Average rank (bars) across benchmark datasets. IF* shown for reference.

4.2 Real-World Use Cases

Data Description. LingSpam dataset (see footnote 3) consists of 2412 non-spam and 481 spam email messages from a linguistics mailing-list. We evaluate two use cases (1) domain-expert knowledge as PI and (2) compute-expensive PI on LingSpam.

BotOrNot dataset (see footnote 4) is collected from Twitter during December 30, 2009 to August 2, 2010. It contains 22,223 content polluters (bots) and 19,276 legitimate users, along with their number of followings over time and tweets. For our experiments, we select accounts with age less than 10 days (for early detection task) at the beginning of dataset collection. The subset contains 901 legitimate (human) accounts and 4535 bots. We create 10 sets containing all the legitimate and a random 10% sample of the bots. We evaluate use case (3) "historical future" as PI and report the results averaged over these sets.

Case 1: Domain-Expert Knowledge as PI for Email Spam Detection.
X* **space:** The Linguistic Inquiry and Word Count (LIWC) software[6] is a widely used text analysis tool in social sciences. It uses a manually-curated keyword dictionary to categorize text into 90 psycholinguistic classes. Construction of LIWC dictionary relies exclusively on human experts which is a slow and evolving process. For the LingSpam dataset, we use the percentage of word counts in each class (assigned by LIWC software) as the privileged features.

***X* space:** The bag-of-word model is widely used as feature representation in text analysis. As such, we use the term frequencies for our email corpus as the primary features.

Figure 4 shows the detection performance[7] of algorithms in ROC curves (averaged over 15 independent runs on stratified train-test splits). We find that IF, which does not leverage PI but operates solely in X space, is significantly worse than most PI-incorporated methods. OC-SVM+ is nearly as poor as IF despite using PI—this is potentially due to OC-SVM being a poor anomaly detector in the first place, as shown in [6] and as we argued in Sect. 1. All knowledge transfer methods, SPI, SPI-LITE, and FT, perform similarly on this case study, and are as good as IF*, directly using X^*.

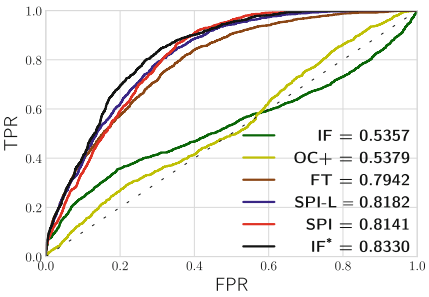


Fig. 4. Detection performance on Case 1: using expert knowledge as PI. Legend depicts the AUC values. PI-incorporated detectors (except OC-SVM+) outperform non-PI IF and achieve similar performance to IF*.

Case 2: Compute-Expensive Features as PI for Email Spam Detection.
X* **space:** Beyond bag-of-words, one can use syntactic features to capture *stylistic* differences between spam and non-spam emails. To this end, we extract features from the parse trees of emails using the StanfordParser[8]. The parser

[6] https://liwc.wpengine.com/.
[7] See supplementary material quantifying the performance of methods against other ranking metrics.
[8] https://nlp.stanford.edu/software/lex-parser.shtml.

provides the taxonomy (tree) of Part-of-Speech (PoS) tags for each sentence, based on which we construct (*i*) PoS bi-gram frequencies, and (*ii*) quantitative features (width, height, and horizontal/vertical imbalance) of the parse tree.

On average, StanfordParser requires 66 s[9] to parse and extract features from a single raw email in LingSpam. Since the features are computationally demanding, we incorporate those as PI to facilitate faster detection at test time.

***X* space:** We use the term frequencies as the primary features as in Case 1.

Figure 5(a) shows the detection performance (see footnote 7) of methods in terms of AUC under ROC. We find that IF* using (privileged) syntactic features achieves lower AUC of ∼0.65 as compared to ∼0.83 using (privileged) LIWC features in Case 1. Accordingly, all methods perform relatively lower, suggesting that the syntactic features are less informative of spam than psycholinguistic ones. Nonetheless, we observe that the performance ordering remains consistent, where IF ranks at the bottom and SPI and SPI-LITE get closest to IF*.

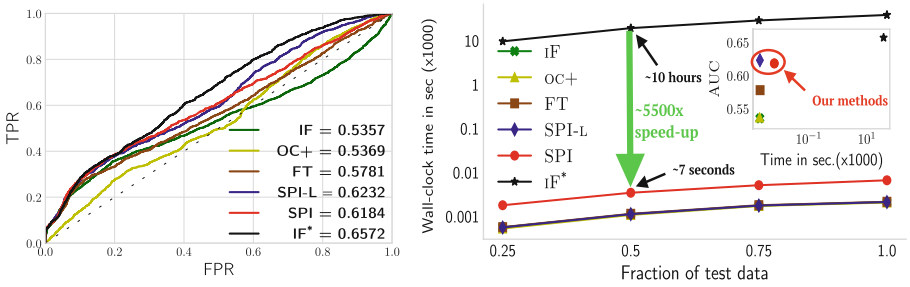


Fig. 5. Comparison of detectors on Case 2: using computationally-expensive features as PI. (a) detection performance, legend depicts AUC values; and (b) wall-clock time required (in seconds, note the logarithmic scale) vs. test data size [inset plot on top right: AUC vs. time (methods depicted with symbols)].

Figure 5(b) shows the comparison of wall-clock time required by each detector to compute the anomaly scores at test time for varying fraction of test data. On average, SPI achieves 5500× speed-up over IF* that employs the parser at test time. This is a considerable improvement of response time for comparable accuracy. Also notice the inset plot showing the AUC vs. total test time, where our proposed SPI and SPI-LITE are closest to the ideal point at the top left.

Case 3: "Historical Future" as PI for Twitter Bot Detection.
We use temporal data from the activity and network evolution of an account to capture behavioral differences between a human and a bot. We construct temporal features including volume, rate-of-change, and lag-autocorrelations of the number of followings. We also extract temporal features from text such as count of tweets, links, hash-tags and mentions.

[9] Using a single thread on 2.2 GHz Intel Core i7 CPU with 8 cores and 16 GB RAM.

$\boldsymbol{X}^*$ **space**: All the temporal features within f_t days in the future (relative to detection at time t) constitute privileged features. Such future values would not be available at any test time point but can be found in historical data.

$\boldsymbol{X}$ **space**: Temporal features within h_t days in the past as well as static user features (from screen name and profile description) constitute primary features.

Figure 6(a) reports the detection performance of algorithms in terms of ROC curves (averaged over 10 sets) at time $t = 2$ days after the data collection started; for $h_t = 2$, $f_t = 7$.[10] The findings are similar to other cases: SPI and SPI-LITE outperform the competing methods in terms of AUC and OC-SVM+ performs similar to non-PI IF; demonstrating that knowledge transfer based methods are more suitable for real-world use cases.

Figure 6(b) compares the detection performance of SPI and IF over time; for detection at $t = \{0, 1, 2, 3, 4\}$. As time passes, historical data grows as $h_t = \{0, 1, 2, 3, 4\}$ where "historical future" data is fixed at $f_t = 7$ for PI-incorporated methods. Notice that at time $t = 1$, SPI achieves similar detection performance to IF's performance at $t = 2$ that uses more historical data of 2 days. As such, SPI enables 24 h early detection as compared to non-PI IF for the same accuracy. Notice that with the increase in historical data, the performances of both methods improve, as expected. At the same time, that of SPI improves faster, ultimately reaching a higher saturation level, specifically ∼7% higher relative to IF. Moreover, SPI gets close to IF*'s level in just around 3 days.

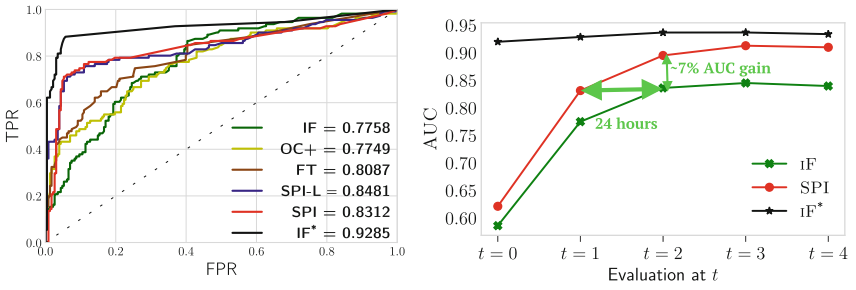


Fig. 6. Comparison of detectors on Case 3: using "historical future" data as PI. (a) SPI outperforms competition in performance and is closest to IF*'s; (b) SPI achieves same detection performance as IF 24 h earlier, and gets close to IF* in 3 days of history.

5 Related Work

We review the history of LUPI, follow up and related work on learning with side/hidden information, as well as LUPI-based anomaly detection.

[10] Same conclusions can be drawn for $f_t \in \{1, 3, 5, 7\}$ (see supplementary material).

Learning Under Privileged Information: The LUPI paradigm is introduced by Vapnik and Vashist [19] as the SVM+ method, where, Teacher provides Student not only with (training) examples but also explanations, comparisons, metaphors, etc. which accelerate the learning process. Roughly speaking, PI adjusts Student's concept of similarity between training examples and reduces the amount of data required for learning. Lapin et al. [10] showed that learning with PI is a particular instance of importance weighting in SVMs. Another such mechanism was introduced more recently by Vapnik and Izmailov [17], where knowledge is transferred from the space of PI to the space where the decision function is built. The general idea is to specify a small number of fundamental concepts of knowledge in the privileged space and then try to transfer them; i.e., construct additional features in decision space via e.g., regression techniques in decision space. Importantly, the knowledge transfer mechanism is not restricted to SVMs, but generalizes, e.g. to neural networks [18].

LUPI has been applied to a number of different settings including clustering [7,12], metric learning [8], learning to rank [15], malware and bot detection [2,3], risk modeling [14], as well as recognizing objects [16], actions and events [13].

Learning with Side/Hidden Information: Several other work, particularly in computer vision [4,20], propose methods to learn with data that is unavailable at test time referred as side and hidden information (e.g., text descriptions or tags for general images, facial expression annotations for face images, etc.). In addition, Jonschkowski et al. [9] describe various patterns of learning with side information. All of these work focus on supervised learning problems.

LUPI-Based Anomaly Detection: With the exception of One-Class SVM (OC-SVM+) [2], which is a direct extension of Vapnik's (supervised) SVM+, the LUPI framework has been utilized only for supervised learning problems. While anomaly detection has been studied extensively [1], we are unaware of any work other than [2] leveraging privileged information for unsupervised anomaly detection. Motivated by this along with the premises of the LUPI paradigm, we are the first to design a new technique that ties LUPI with unsupervised tree-based ensemble methods, which are considered state-of-the-art for anomaly detection.

6 Conclusion

We introduced SPI, a new ensemble approach that leverages privileged information (data available only for training examples) for unsupervised anomaly detection. Our work builds on the LUPI paradigm, and to the best of our knowledge, is the first attempt to incorporating PI to improve the state-of-the-art ensemble detectors. We validated the effectiveness of our method on both benchmark datasets as well as three real-world case studies. We showed that SPI and SPI-LITE consistently outperform the baselines. Our case studies leveraged a variety of privileged information—"historical future", complex features, expert knowledge—and verified that SPI can unlock multiple benefits for anomaly detection in terms of detection latency, speed, as well as accuracy.

Acknowledgements. This research is sponsored by NSF CAREER 1452425 and IIS 1408287. Any conclusions expressed in this material are of the authors and do not necessarily reflect the views, expressed or implied, of the funding parties.

References

1. Aggarwal, C.C.: Outlier analysis. Data Mining, pp. 237–263. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14142-8_8
2. Burnaev, E., Smolyakov, D.: One-class SVM with privileged information and its application to malware detection. In: ICDM Workshops, pp. 273–280 (2016)
3. Celik, Z.B., McDaniel, P., Izmailov, R., Papernot, N., Swami, A.: Extending detection with forensic information. arXiv:1603.09638 (2016)
4. Chen, J., Liu, X., Lyu, S.: Boosting with side information. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012. LNCS, vol. 7724, pp. 563–577. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37331-2_43
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
6. Emmott, A., Das, S., Dietterich, T., Fern, A., Wong, W.-K.: Systematic construction of anomaly detection benchmarks from real data. In: KDD ODD (2013)
7. Feyereisl, J., Aickelin, U.: Privileged information for data clustering. Inf. Sci. **194**, 4–23 (2012)
8. Fouad, S., Tino, P., Raychaudhury, S., Schneider, P.: Incorporating privileged information through metric learning. IEEE Neural Netw. Learn. Syst. **24**(7), 1086–1098 (2013)
9. Jonschkowski, R., Höfer, S., Brock, O.: Patterns for learning with side information. arXiv:1511.06429 (2015)
10. Lapin, M., Hein, M., Schiele, B.: Learning using privileged information: SVM+ and weighted SVM. Neural Netw. **53**, 95–108 (2014)
11. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: ICDM (2008)
12. Marcacini, R.M., Domingues, M.A., Hruschka, E.R., Rezende, S.O.: Privileged information for hierarchical document clustering: a metric learning approach. In: ICPR, pp. 3636–3641 (2014)
13. Niu, L., Li, W., Xu, D.: Exploiting privileged information from web data for action and event recognition. Int. J. Comput. Vis. **118**(2), 130–150 (2016)
14. Ribeiro, B., Silva, C., Chen, N., Vieira, A., das Neves, J.C.: Enhanced default risk models with SVM+. Expert Syst. Appl. **39**(11), 10140–10152 (2012)
15. Sharmanska, V., Quadrianto, N., Lampert, C.H.: Learning to rank using privileged information. In: ICCV, pp. 825–832 (2013)
16. Sharmanska, V., Quadrianto, N., Lampert, C.H.: Learning to transfer privileged information. arXiv:1410.0389 (2014)
17. Vapnik, V., Izmailov, R.: Learning with intelligent teacher: similarity control and knowledge transfer. In: Gammerman, A., Vovk, V., Papadopoulos, H. (eds.) SLDS 2015. LNCS (LNAI), vol. 9047, pp. 3–32. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17091-6_1
18. Vapnik, V., Izmailov, R.: Knowledge transfer in SVM and neural networks. Ann. Math. Artif. Intell. **81**(1–2), 3–19 (2017)
19. Vapnik, V., Vashist, A.: A new learning paradigm: learning using privileged information. Neural Netw. **22**(5–6), 544–557 (2009)
20. Wang, Z., Ji, Q.: Classifier learning with hidden information. In: CVPR (2015)