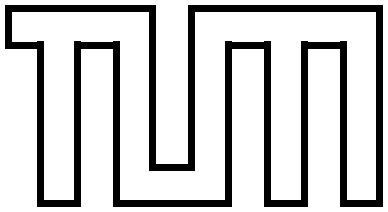
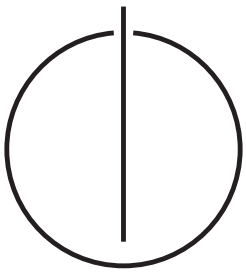


Detecting Distribution Shifts in Machine Learning

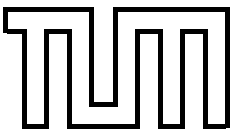


Stephan Rabanser

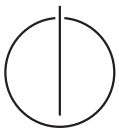
Department of Informatics

Technical University of Munich

Master's Thesis in Informatics



Technical University of Munich
Department of Informatics



Detecting Distribution Shifts in Machine Learning

Erkennung von Verteilungsveränderungen im Maschinellen Lernen

A

Master's Thesis in Informatics

submitted by

Stephan Rabanser

on

July 15th 2019.

Supervisor
Prof. Dr. Stephan Günnemann
Technical University of Munich
Department of Informatics
Munich, Germany

Advisor
Prof. Zachary C. Lipton, PhD
Carnegie Mellon University
Machine Learning Department
Pittsburgh, PA

Declaration
I confirm that this master's thesis is my own work and I have documented all sources and material used.

Erklärung
Ich versichere, dass ich diese Masterarbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Munich, July 15th 2019

Stephan Rabanser

Acknowledgements

First and foremost, I would like to express my deep gratitude to my thesis supervisor **Prof. Dr. Stephan Günnemann** (Technical University of Munich, Department of Informatics) and my thesis advisor **Prof. Zachary Chase Lipton, PhD** (Carnegie Mellon University, Tepper School of Business and Machine Learning Department) for their patient guidance, enthusiastic encouragement, useful critiques, and relentless dedication for/to this work. In particular, I want to thank Prof. Lipton for making a long-held dream come true by inviting me to conduct most of the work concerning this thesis as a Visiting Research Scholar at Carnegie Mellon University.

Moreover, I wish to thank **Prof. Aaditya Ramdas, PhD** (Carnegie Mellon University, Department of Statistics and Data Science) for his ongoing support throughout the course of this work concerning multiple hypothesis testing and **Dr. Stephan Haug** (Technical University of Munich, Department of Mathematics) for his inputs in the early stages of this work on general statistical hypothesis testing.

In addition, I would like to thank my colleagues **Lukas Prantl**, **Tammo Rukat**, and **all lab members** from both Prof. Günnemann's and Prof. Lipton's teams for our fruitful discussions about this work. Further, I also wish to acknowledge the important role of inputs provided by **conference attendees and anonymous reviewers** to whom I presented my work in the last couple of months.

Special thanks go to **my parents** for their ongoing financial support throughout my graduate studies, especially for assisting me in my wish to conduct the biggest chunk of my thesis work as an abroad stay in the United States and for funding my attendance at Neural Information Processing Systems (NeurIPS) 2018 and at the International Conference on Learning Representations (ICLR) 2019.

Abstract

We might hope that when faced with unexpected inputs, well-designed software systems would fire off warnings. Machine learning (ML) systems, however, which depend strongly on properties of their inputs (e.g. the i.i.d. assumption), tend to fail silently as thorough input validation is often insufficiently implemented in operationalized ML pipelines.

As part of this thesis we explore the problem of building ML systems that fail loudly, investigating methods for detecting dataset shift, identifying exemplars that most typify the shift, and quantifying shift malignancy. We focus on several datasets and various perturbations to both covariates and label distributions with varying magnitudes and fractions of data affected.

Interestingly, we show that across the dataset shifts that we explore, a two-sample-testing-based approach, using pre-trained classifiers for dimensionality reduction, performs best. Moreover, we demonstrate that domain-discriminating approaches tend to be helpful for characterizing shifts qualitatively and determining if they are harmful.

Zusammenfassung

In der Informatik gilt es als allgemeine Praxis, dass gut konzipierte Softwaresysteme im Falle von unerwarteten Eingaben Warnungen ausgeben. Systeme, die auf Algorithmen des Maschinellen Lernens (ML) basieren, sind jedoch stark von Eigenschaften der Eingabedaten abhängig (wie unter anderem der i.i.d.-Annahme) und scheitern daher oft im Stillen. Dies liegt insbesondere meist daran, dass eine gründliche Überprüfung der Eingaben in die ML Pipeline oft nur unzureichend implementiert ist.

Im Rahmen dieser Masterarbeit befassen wir uns mit dem Problem von Verteilungsveränderungen im Maschinellen Lernen. Insbesondere beschäftigen wir uns mit der Erkennung von Verteilungsveränderungen, deren Charakterisierung mittels der Identifizierung von für die Veränderung typischen Exemplaren, sowie der Frage in welchem Umfang die Verteilungsveränderung die Performance des ML Algorithmus beeinflusst. Dabei untersuchen wir mehrere Datensätze mit unterschiedlichsten Abwandlungen sowohl der Daten- als auch der Merkmalsverteilungen in verschiedenen Intensitäten und prozentual betroffenen Datenpunkten.

Wir zeigen, dass über sämtliche untersuchten Perturbationen hinweg ein Zwei-Stichproben-Test-Verfahren basierend auf Repräsentationen eines vor-trainierten Klassifikators die besten Ergebnisse erzielt. Außerdem legen unsere Ergebnisse nahe, dass Domain-diskriminierende Verfahren hilfreich sind, um Verteilungsveränderungen qualitativ zu charakterisieren und um festzustellen ob diese schädlich sind.

Table of contents

List of figures

List of tables

Nomenclature

Probability distributions

χ^2	Chi-Squared distribution
$\mathcal{N}$	Gaussian distribution
p	Source distribution
q	Target distribution
Bin	Binomial distribution

Functions

$\beta(\cdot)$	Power function
$\kappa(\cdot, \cdot)$	Kernel function
$\mathbb{I}$	Indicator function
$\phi(\cdot)$	Nonlinear activation function
$\varphi(\cdot)$	Feature map
$f(\cdot)$	Label classifier

Integer variables

C	Number of total classes
D	Data dimensionality
K	Reduced data dimensionality
N	Sample size

N_{te} Testing sample size

N_{tr} Training sample size

N_{val} Validation sample size

Matrices and vectors

$\hat{\boldsymbol{X}}$ Dimensionality-reduced data matrix

$\hat{\boldsymbol{y}}'$ Estimated testing label vector

$\boldsymbol{\mu}$ Mean vector

$\boldsymbol{\Sigma}$ Covariance matrix

$\tilde{\boldsymbol{X}}$ Zero-centered data matrix

$\boldsymbol{b}$ Bias vector

$\boldsymbol{h}$ Hidden vector

$\boldsymbol{i}$ Input vector

$\boldsymbol{K}$ Gramm matrix

$\boldsymbol{o}$ Output vector

$\boldsymbol{R}$ Transformation matrix for dimensionality reduction

$\boldsymbol{w}$ Weight vector

$\boldsymbol{X}$ Training data matrix

$\boldsymbol{X}'$ Testing data matrix

$\boldsymbol{y}$ Training label vector

$\boldsymbol{y}'$ True testing label vector

Other Symbols

α Significance level

δ Shift intensity

σ^2 Variance

H_0	Null hypothesis
H_A	Alternative hypothesis
I, J	Total count for running indices
i, j	Running indices
S	Statistical test
X^2	Chi-Squared statistic
Z	Kolmogorov-Smirnov statistic

Sets and spaces

$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathcal{B}$	Borel set
$\mathcal{D}$	Dataset
$\mathcal{F}$	Reproducing kernel Hilbert space
$\mathcal{H}$	Latent/hidden space
$\mathcal{M}$	Statistical model
$\mathcal{P}$	Power set
$\mathcal{S}$	σ-algebra
$\mathcal{X}$	Sample space
$\mathcal{Y}$	Label space
Θ	Parameter space
A	Acceptance region
R	Rejection region

Chapter 1

Introduction

Owing to breakthroughs on a variety of practical supervised learning problems, software systems employing deep neural networks are now applied widely in industry, powering the vision systems in social networks (Stone et al., 2008) and self-driving cars (Bojarski et al., 2016), providing assistance to radiologists (Lakhani and Sundaram, 2017), underpinning recommendation engines used by online retailers and digital content providers (Cheng et al., 2016; Covington et al., 2016), enabling the best-performing commercial speech recognition software (Graves et al., 2013; Hinton et al., 2012), and automating translation between languages (Sutskever et al., 2014). In each of these systems, predictive models are integrated into conventional human-interacting software systems, which leverage their predictions to drive consequential real-world decisions.

The reliable functioning of software depends crucially on tests, such as unit tests, input validations, and deployment verifications. Many classic software bugs can be caught when software is compiled, e.g. that a function receives input of the wrong type, while other problems are detected only at run-time, triggering warnings or exceptions. In the worst case, if the errors are never caught, software may behave incorrectly without alerting anyone to the problem.

Unfortunately, software systems based on machine learning are notoriously hard to test and maintain (Sculley et al., 2014). Despite their power, modern machine learning models are brittle. Seemingly subtle changes in the data distribution can destroy the performance of otherwise state-of-the-art classifiers, a phenomenon exemplified by adversarial examples (Szegedy et al., 2013; Zügner et al., 2018). When decisions are made under uncertainty, even shifts in the label distribution can significantly compromise accuracy (Lipton et al., 2018; Zhang et al., 2013).

Unfortunately, in practice, ML pipelines rarely inspect incoming data for signs of distribution shift. Moreover, best practices for detecting shift in high-dimensional real-

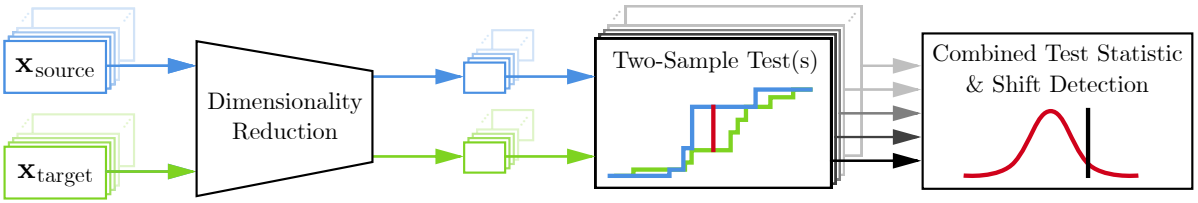


Fig. 1.1 Our pipeline for detecting dataset shift. Source and target data is fed through a dimensionality reduction process and subsequently analyzed through statistical hypothesis testing. We consider various choices for how to represent the data and how to perform two-sample tests.

world data have not yet been established[1]. The first indications that something has gone awry might come when customers complain.

In this thesis, we investigate methods for detecting and characterizing distribution shift, with the hope of removing a critical stumbling block obstructing the safe and responsible deployment of machine learning in high-stakes applications. Faced with distribution shift, our goals are three-fold: (i) detect when distribution shift occurs from as few examples as possible; (ii) characterize the shift, e.g. by identifying those samples from the test set that appear over-represented in the target data; and (iii) provide some guidance on whether the shift is harmful or not. As part of this thesis we principally focus on goal (i) and explore preliminary approaches to (ii) and (iii).

Although a couple of related work have already established shift detection and correction schemes (see Chapter 2), many of these methods require specific assumptions on the underlying properties of the shift, producing wrong predictions whenever these specific conditions are not met. Furthermore, some of these properties are either unknown or intractable to estimate in many real-life settings. To that end, we propose a general purpose detection method that does not impose a structured (distributional) assumption on how the shift occurs.

We investigate shift detection through the lens of statistical two-sample testing where we wish to test the equivalence of the *source* distribution p (from which training data is sampled) and *target* distribution q (from which real-world data is sampled). For simple univariate distributions, such hypothesis testing is a mature science. For example, given draws from two different Bernoulli random variables, well-established procedures govern whether to reject the null hypothesis that their means are the same. On the other hand for color photographs, with high-dimensional continuously-valued inputs, best

[1] TensorFlow's data validation tools compare only summary statistics of source vs target data: https://tensorflow.org/tfx/data_validation/get_started#checking_data_skew_and_drift

practices for detecting shift have not yet been established. While off-the-shelf methods for kernel-based multivariate two-sample tests are appealing, they scale badly with dataset size and their statistical power is known to decay badly with high ambient dimension (Ramdas et al., 2015).

One natural approach to ML practitioners might be to train a classifier to distinguish between source and target examples. Given class-balanced holdout samples, we can pronounce the data shifted if our classifier can recognizes the domain with significantly greater than 50% accuracy and simultaneously determine the nature of the shift and its malignancy. Analyzing the simple case where one wishes to test the means of two Gaussians, Ramdas et al. (2016) recently showed that the power of a classification-based two-sample test using Fisher's Linear Discriminant Analysis classifier achieves minimax rate-optimal performance. However, the performance of classifier-based approaches has not been characterized (either theoretically or empirically) for the complex high-dimensional data distributions on which modern machine learning is routinely deployed. Providing this empirical analysis is a key contribution of this work. Note that throughout this thesis, to avoid confusion, we denote any source-vs-target classifier a *domain classifier* and refer to classifiers trained (on source data) for the original classification task as a *label classifier*.

One benefit of the domain-classifier approach is that it reduces dimensionality to a single dimension, learned precisely for the purpose of discriminating between source and target data. However, learning such a classifier from scratch may require large amounts of training data. Adding to the problem, the domain-classifier approach requires partitioning our (scarce) target data, e.g. using half for training and leaving the remainder for two-sample testing. Alternatively we also explore the *black box shift detection (BBSD)* approach due to Lipton et al. (2018), which addresses shift detection under the label shift assumption. They show that if one possesses an off-the-shelf label classifier $f(\cdot)$ with an invertible confusion matrix (verifiable on training data), then detecting that the source distribution p differs from the target distribution q requires only detecting that $p(f(\cdot)) \neq q(f(\cdot))$. This insight enables efficient shift detection, using a pre-trained (label) classifier for dimensionality reduction. Building on these ideas of combining black-box dimensionality reduction with subsequent two-sample testing, we explore a range of dimensionality reduction techniques and compare them under a wide variety of shifts (see Figure 1.1 for an illustration of our general framework). We show (empirically) that BBSD works surprisingly well under a broad set of shifts, even when the label shift assumption is not met.

Chapter 2

Related Work

The problems of detecting and correcting dataset shift relate to extensive prior work in the domain adaptation and anomaly detection literature, as well as in the time-series literature on the problem of change-point detection.

2.1 Early Works

Perhaps the earliest work directly relevant to distribution shift comes from groundbreaking work in epidemiology and public health on propensity scoring and the work of econometricians James Heckman, who studied sample selection bias (Heckman, 1977), and Manski who studied the related problem of estimating choice probabilities from choice-based samples (Manski and Lerman, 1977). These works were not concerned just with detecting dataset shift but with estimating propensity scores to correct for biases in collected data sets. These datasets were concerned with simple survey data and did not address the problems that emerge from working with high-dimensional data. Shortly after, similar methods were developed in the public health literature to estimate propensity scores for correcting for biases in the construction of cohorts for health studies (Rosenbaum and Rubin, 1983).

2.2 Anomaly Detection

Given just one example and the task of estimating whether or not it came from the same distribution that generated the training data, the dataset shift problem simplifies to the widely studied problem of anomaly detection, surveyed thoroughly by Chandola et al. (2009); Markou and Singh (2003). Some popular approaches here include density

estimation (Breunig et al., 2000), margin-based approaches such as the one-class support vector machine (Schölkopf et al., 2000), the tree-based isolation forest method due to Liu et al. (2008) and recently also GANs have proven to be potentially useful for this task (Schlegl et al., 2017) although this line of work remains fairly heuristic. Given simple streams of data arriving in a time-dependent fashion where the signal is piece-wise stationary, with stationary periods separated by abrupt changes, the problem of detecting a dataset shift becomes the classic time series problem of change point detection, with existing methods summarized succinctly in an excellent survey by Truong et al. (2018).

2.3 Domain Adaptation

In the machine learning literature, a number of papers have addressed distribution shift, generally under the umbrella of domain adaptation. In the most familiar setup, the learner has access to labeled training data from the source distribution and unlabeled test data from the target distribution and is tasked with producing a classifier that performs best on the test (target) data. In general, the problem is widely known to be impossible, with a number of known impossibility theorems (Ben-David et al., 2010). Trivially, one can see that given no assumptions about either the support or labeling function for the target distribution, there is no free lunch to be had. However, several assumptions allow one to theoretically guarantee performance on target data. Schölkopf et al. (2012) illustrated correspondences between the various assumptions one might make on precisely what is invariant between train and test times and causal assumptions on the data generating process. Assuming that the support of the target distributions is a subset of the training distribution $q(\boldsymbol{x}) > 0 \Rightarrow p(\boldsymbol{x}) > 0$ and that the conditional distribution $p(y|\boldsymbol{x}) = q(y|\boldsymbol{x})$ stays fixed gives the familiar problem of learning under covariate shift, which corresponds to the assumption that $\boldsymbol{x}$ causes y. Covariate shift has been widely studied in the machine learning community since Shimodaira (2000) and subsequently addressed with a popular kernel-mean matching approach due to Gretton et al. (2012). Under the label shift assumption, the reverse conditional probability remains fixed $p(\boldsymbol{x}|y) = q(\boldsymbol{x}|y)$. This setting corresponds to the reverse assumption that the label causes the covariates, as is applicable in diagnostic-type problems, and has been extensively explored by Beijbom et al. (2012); Chan and Ng (2005); Lipton et al. (2018); Storkey (2009); Zhang et al. (2013). Recently, Lipton et al. (2018) showed that given a classifier whose confusion matrix is invertible, one can produce a consistent estimator of the test set label distribution with sharp error bounds. Under the label shift assumption, detecting distribution shift simplifies to the task of detecting label shift.

2.4 Out-Of-Distribution Sample Detection

Several recent papers have proposed outlier detection mechanisms dubbing the task out-of-distribution (OOD) sample detection. Hendrycks and Gimpel (2017) proposes to simply threshold the maximum softmax entry of a neural network classifier which already seems to contain a relevant signal. Liang et al. (2018) and Lee et al. (2018) extend this idea by either adding temperature scaling and adversarial-like perturbations on the input or by explicitly adapting the loss to aid OOD detection. Choi and Jang (2018) and Shalev et al. (2018) employ model ensembling to further improve detection reliability. Alemi et al. (2018) motivate use of the variational information bottleneck. Hendrycks et al. (2019) expose the model to OOD samples, exploring heuristics for discriminating between in-distribution and out-of-distribution samples. Shafaei et al. (2018) survey and compare numerous OOD detection techniques.

Chapter 3

Machine Learning and Statistical Foundations

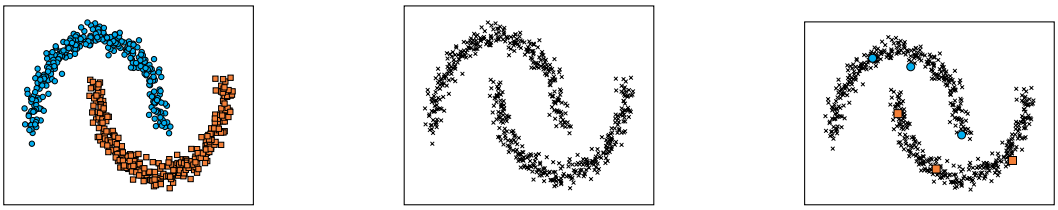
Before diving into distribution shifts and methods for detecting them, we need to introduce a couple of basic foundational terms and concepts from statistics and machine learning. Since the purpose of this chapter is to give an overview of the key concepts we make explicit use of in this thesis, it is not designed to be a comprehensive review of machine learning or statistics as a whole. A thorough treatment of machine learning is provided by Bishop (2006), Murphy (2012), Goodfellow et al. (2016), and Mohri et al. (2018) while key statistical concepts are provided by Friedman et al. (2001), James et al. (2013), and Georgii (2015). Readers well versed in both machine learning and statistics can freely skip this chapter and continue reading with Chapter 4.

3.1 Machine Learning Basics

3.1.1 Definition

Originally introduced by Arthur Samuel, the term machine learning refers to the set of algorithms and models used to make a computer perform a certain task by only providing a collection of data points and no explicit instructions on how to succeed in said task. Hence, machine learning can also be viewed as the process of learning and generating insights from data. Perhaps the most intuitively understandable definition of machine learning is given by Andrew Ng:

> Machine learning is the science of getting computers to act without being explicitly programmed.



(a) Supervised (b) Unsupervised (c) Semi-supervised

Fig. 3.1 Canonical machine learning problems visualized. Raw data points are depicted as black crosses, while labeled data points are shown as colored circles and squares.

A more formal definition of what it means for a computer to learn was given by Tom Mitchell (Mitchell, 1997):

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E.

3.1.2 Canonical Machine Learning Problems

Machine learning problems can generally be divided into four distinct canonical subgroups: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning (Mohri et al., 2018). Each of them comes with their own set of assumptions about how data is collected, represented, and consequently used for learning. Most importantly, these subgroups differ in the amount of external supervision they receive from either field experts or the environment. We briefly introduce the formal setup for each of the mentioned problem classes below, skipping reinforcement learning as we do not cover reinforcement learning as part of our shift detection strategy.

Supervised Learning The strongest degree of external supervision is provided in *supervised learning*, whose purpose is to learn a function mapping inputs to outputs. In order to learn that function, a labeled dataset

$$\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\} \in (\mathcal{X} \times \mathcal{Y}) \tag{3.1}$$

is given where we define $\mathcal{X} = \mathbb{R}^D$ with $D \in \mathbb{N}$. Depending on the problem setup, we either refer to the problem class as *regression* if inputs map to continuous outputs, formally $\mathcal{Y} = \mathbb{R}$, or *classification* if inputs map to a discrete set of outputs, formally $\mathcal{Y} = \{y_1, \ldots, y_C\}$ with $C \in \mathbb{N}$.

Unsupervised Learning While most of the currently commercially viable applications of machine learning are based on labeled datasets, most of the data available in the wild is in fact unlabeled, which means that we can only collect raw data points

$$\mathcal{D} = \boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathcal{X} \tag{3.2}$$

where we again define $\mathcal{X} = \mathbb{R}^D$. The central aim of *unsupervised learning* is therefore to uncover hidden structures in the data without external supervision. Typical tasks encompass clustering, density estimation, dimensionality reduction, outlier detection, and learning in latent variable models.

Semi-Supervised Learning In some cases, both labeled and unlabeled data points are both present in the same collection of data, i.e.

$$\mathcal{D} = (\mathcal{X} \times \mathcal{Y}) \ni \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\} \cup \{\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_N\} \in \mathcal{X}. \tag{3.3}$$

While naive algorithms might discard either unlabeled data points or labels from labeled points to make the entire dataset suitable for either supervised or unsupervised learning, *semi-supervised learning* algorithms can make use of both labeled and unlabeled points. Typically, semi-supervised learning is applied in cases where a lot of unlabeled data points and only a few labeled data points are given.

A visual example of supervised, semi-supervised, and unsupervised data representations is provided in Figure 3.1.

3.1.3 The Typical Machine Learning Pipeline

While machine learning use-cases can vary greatly, there are some established best practices of how to architect machine learning pipelines (Géron, 2017). An overview of the most important steps is depicted in Figure 3.2 and described below:

1. **Data collection**: First, a dataset is collected from which a model of the underlying data generating process can subsequently be derived.

2. **Data exploration**: Data collection is usually followed by data exploration, a stage in which summary statistics and visualizations play a key role in gaining initial insights from the data.

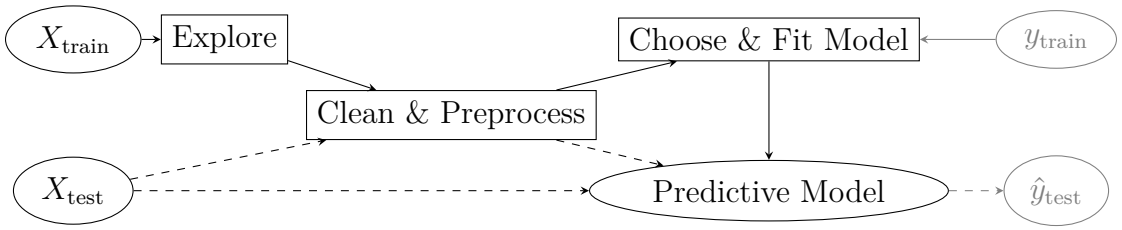


Fig. 3.2 The typical machine learning pipeline. Oval nodes depict components while rectangular nodes depict actions. Solid arrows indicate paths explored during training while dashed arrows indicate paths explored during prediction. Gray nodes are only applicable in supervised learning settings.

3. **Data cleaning & pre-processing**: In many cases, the collected data can not be directly used for model fitting, which is why the data needs to undergo some cleaning and pre-processing in order to be used more effectively during training.

4. **Model selection & fitting**: Having gained some initial understanding of the data space, a suitable model can be selected and trained on the cleaned data.

5. **Prediction**: Finally, new incoming data is fed into the model to obtain predictions. In some cases, these data points might also have to go through a cleaning-stage, while they might be directly fed into the model without any preprocessing in other scenarios.

3.2 Supervised Learning Algorithms

As part of this section, we briefly introduce foundational non-linear supervised learning algorithms: kernel methods and neural networks.

3.2.1 Kernel Methods

While some (toy-)problems can be solved using linear models, real-world interactions between different model inputs are usually highly non-linear, limiting the usage of purely linear models. Concretely, regression problems are often not solvable using a simple linear equation and classes in classification problems cannot usually be discriminated against using a simple linear decision boundary.

Kernel methods, which are thoroughly discussed in Schölkopf and Smola (2001), provide one way of addressing non-linear classification problems by mapping inputs into a higher-dimensional space in which a linear (or at least approximately linear)

dividing margin, i.e. a decision boundary, between the classes can be established. Since kernel methods conveniently make use of kernel functions (hence their name), the high-dimensional feature space they operate in is never explicitly constructed. Instead, kernel functions, which can be thought of as measuring similarity between pairs of data points in feature space, solely rely on computing inner products between data points, which is computationally favorable since the feature space in only implicitly constructed. This computational advantage is often referred to as the kernel trick.

Formally, a kernel κ is defined as a mapping $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Further, we assume the existence of an inner product space $\mathcal{F}$ and a function φ mapping inputs into this space, i.e. $\varphi : \mathcal{X} \to \mathcal{F}$, which satisfies

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}') \rangle_{\mathcal{F}}. \tag{3.4}$$

According to Mercer's theorem (Mercer, 1909), any function $\kappa(\cdot, \cdot)$ qualifies as a kernel if κ is symmetric, i.e. $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}', \boldsymbol{x})$, and if the matrix containing the computed similarities using κ (also called Gram matrix) $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is positive semi-definite.

One of the most prominent members in the family of kernel methods is the *support vector machine (SVM)* (Schölkopf and Smola, 2001), which is predominantly used for classification tasks. SVMs try to find a decision boundary between classes such that the neighborhood around the dividing margin is free from any data points, hence the alternative name *maximum margin classifier*.

3.2.2 Neural Networks

Inspired by biological neural networks found in biological structures like animal brains, an even more powerful way of addressing non-linear problems is through *(artificial) neural networks* (ANNs) (Bishop, 1995). Although reliant on large amounts of data, modern neural networks are now considered the work-horses of supervised learning and have reached close to human (sometimes even super-human) performance in a wide variety of tasks (Abiodun et al., 2018).

Plain ANNs are formed by combining multiple neurons into a connected network of neurons/nodes. Each node receives a fixed number of inputs $\boldsymbol{i} = (i_1, \ldots, i_D)$ weighted with their corresponding weights $\boldsymbol{w} = (w_1, \ldots, w_D)$. A neuron effectively computes a weighted sum over its inputs to which a bias term b is added, followed by the application of a non-linear activation. Hence, nodes can be thought of as non-linear weighting

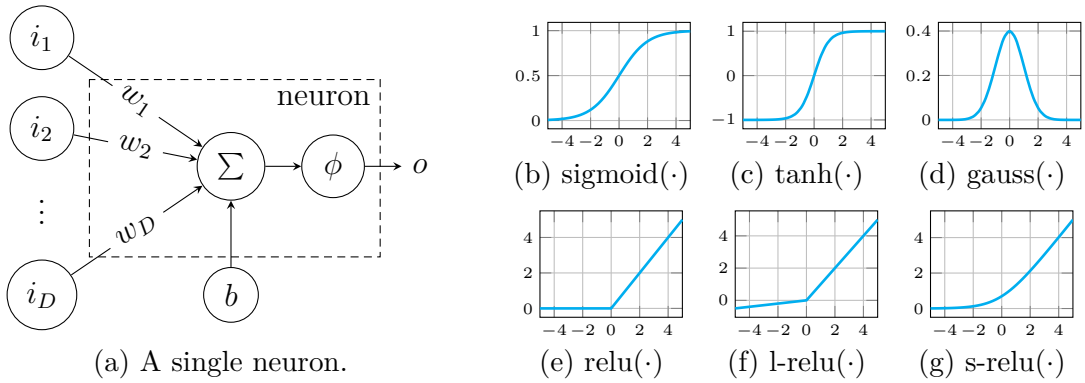


Fig. 3.3 A neuron is the basic building block of any neural network. (a) shows the inner workings of a single neuron, which involves computing a weighted sum over the inputs, adding a bias term b, and then applying a nonlinear activation function ϕ. (b)-(g) show popular activation functions.

functions, formally

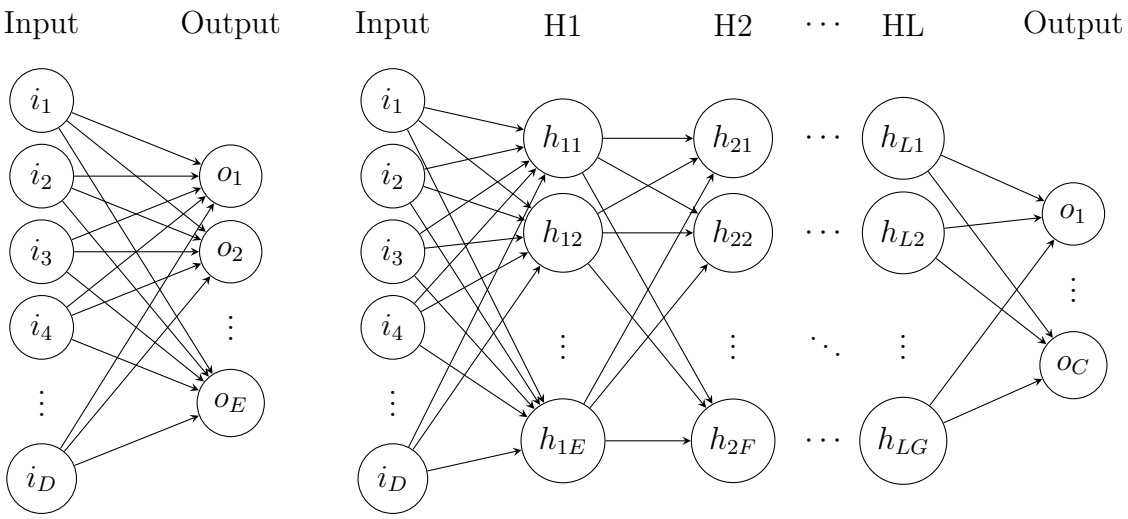
$$o = \phi\left(\sum_{j=1}^{D} i_j w_j + b\right). \tag{3.5}$$

The inner workings of a neuron and some popular non-linear activation functions are shown in Figure 3.3.

By stacking multiple of these neurons in parallel, a neural layer arises; and by sequentially combining multiple of these layers, a neural network is formed (see Figure 3.4).

Neural networks are trained via optimizers which make extensive use of back-propagation (Rumelhart et al., 1988), an iterative algorithm in which the weights and biases are optimized to minimize some loss function by repeatedly making use of the chain rule to derive the degree of change present at a certain neuron with respect to its inputs. Back-propagation consists of two main stages: (i) In the first stage inputs are fed through the network by using the current set of parameters and the output residuals are computed; and (ii) in the second stage, gradients with respect to the parameters are calculated at every node all the way back to the input nodes and the weights are updated accordingly.

Lots of different optimizers, which are designed to minimize a given loss function as quickly as possible have been proposed, amongst the most prominently used variants are: stochastic gradient descent (SGD) (Robbins and Monro, 1951), adaptive moment estimation (ADAM) (Kingma and Ba, 2014), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), and RMSprop. Similar to other machine learning problems, the optimizer choice and its parameters (learning rate, momentum, weight decay, batch size, etc.) but



(a) Parallel neuron stacking to create layers.

(b) Multiple layers being combined into a neural network.

Fig. 3.4 Formation process of neural networks. Bias nodes omitted for increased clarity.

also the network structure itself are considered hyper-parameters and can have a severe impact on model performance.

Convolutional Neural Networks (ConvNets)

Like plain neural networks, *convolutional neural networks* are biologically inspired, specifically by the visual cortex found in animal brains (Hubel and Wiesel, 1959). Although predominantly used for visual recognition tasks (LeCun et al., 1989), convolutional neural networks can also be useful for capturing temporal dynamics.

ConvNets essentially extend upon classical neural networks by introducing two additional layer types: convolution layers and max-pooling layers. Convolutional layers themselves consist of a set of learnable filters (i.e. they need not be hand-engineered like in plain neural networks), while max-pooling layers down-sample the output obtained by a convolutional layer. Typical architectures first stack multiple convolution/max-pooling layers in succession, then flatten the output to a vector and finally pass said vector to a fully-connected plain neural network.

Residual Networks (ResNets)

Residual networks, which are inspired by pyramidal cells found in the cerebral cortex, provide further improvements over standard ConvNets by introducing skip-connections into the architecture (He et al., 2016). These connections can jump over layers to

directly feed into the next layer. This construction allows more effective training of deep convolutional neural networks, since the skip-connections accelerate learning and help fighting the vanishing gradient problem.

3.3 Unsupervised Learning Algorithms

We introduce the basics underlying dimensionality reduction and outlier/novelty detection.

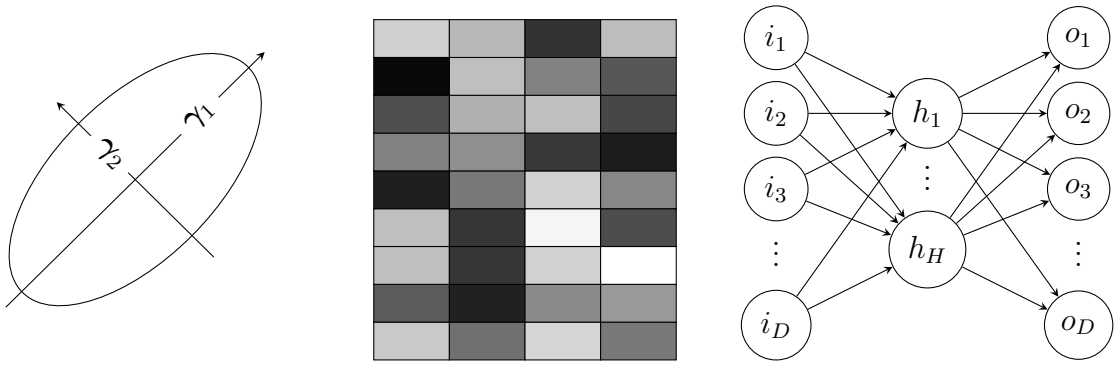
3.3.1 Dimensionality Reduction

It is not uncommon in modern machine learning to deal with high-dimensional datasets, i.e. datasets for which D is staggeringly high. While it is hard to establish an exact threshold for D in order to be speaking of high-dimensional data, common definitions start from a few hundred dimensions reaching to billions of dimensions. Such high-dimensional data is generally challenging for a number of different reasons (Leskovec et al., 2014):

- Due to the curse of dimensionality we need exponential amounts of data to characterize the density as the dimensionality increases.
- Similarity computations are expensive because of the high complexity of the employed distance functions.
- Some algorithms might have trouble detecting structure in the data if dimensions are strongly correlated.
- Exploratory data analysis becomes increasingly prohibitive as it becomes harder to visualize high-dimensional data.

Note that the term high dimensional *data* differs from the term high dimensional *problems*, which usually only refers to datasets where the number of dimensions far exceeds the number of data points, formally $N \ll D$.

While these findings might seem discouraging at first sight, the data often lies in a lower-dimensional manifold which is embedded in a high-dimensional space. It is therefore our goal to derive a set of degrees of freedom which can be used to reproduce most of the variability of a dataset. When performing dimensionality reduction on a dataset from D to K dimensions (where $K \ll D$, $K \in \mathbb{N}$) we try to reduce the dimensionality while avoiding high information loss at the same time. Luckily, this is possible in many cases as many dimensions show either no or only little variability.



(a) PCA example. (b) SRP example. (c) Autoencoder example.

Fig. 3.5 Popular dimensionality reduction methods visualized.

Dimensionality reduction does not only produce a compact low-dimensional encoding of a given high-dimensional dataset, but also has many additional advantages:

- Preprocessing for supervised learning: Simplify, clean, and reduce the data for subsequent training.
- Data visualization: Provide a graphical interpretation of a given dataset.

As a result of dimensionality reduction, data requires less storage and algorithms can process the data more quickly. Also, algorithms which exhibit a quadratic or even cubic dependance on the dimensionality can become tractable when reducing dimensions.

There are various strategies for how to perform dimensionality reduction. Naive ideas include removing unimportant features or removing features with low variance. While these approaches have no need for an intensive preprocessing or training phase to determine relevant dimensions, they are limited in applicability since they either require expert knowledge or only focus on individual features ignoring cross-correlations to other dimensions.

More advanced methods include but are not limited to (standard/probabilistic/kernel) principal components analysis (PCA/PPCA/KPCA), (gaussian/sparse) random projections (GRP/SRP), non-negative matrix factorization (NNMF) (Lee and Seung, 2001), locality-sensitive hashing (LSH) (Gionis et al., 1999), (linear/generalized) discriminant analysis (LDA/GDA) (Fisher, 1936), and autoencoders. As we make extensive use of dimensionality reduction as part of our proposed shift detection algorithm, we introduce the most relevant DR techniques in detail below.

Principal Components Analysis (PCA)

One of the most widely used dimensionality reduction techniques is *principal components analysis* (Pearson, 1901), which finds the optimal orthogonal linear transformation matrix to turn a set of potentially correlated input features into a set of uncorrelated variables. When performing this transformation, the first and most prominent of these variables is called the first principal component and captures the direction of the greatest variance present in the data. The projection ensures that all succeeding components capture increasingly smaller degrees of variability, essentially creating a ranking of the most important orthogonal directions.

Given the covariance matrix $\boldsymbol{\Sigma}$ calculated on the zero-centered data matrix $\tilde{\boldsymbol{X}}$, we can calculate the eigendecomposition of $\boldsymbol{\Sigma}$ as follows

$$\boldsymbol{\Sigma} = \boldsymbol{\Gamma}\boldsymbol{\Lambda}\boldsymbol{\Gamma}^T \tag{3.6}$$

where $\boldsymbol{\Gamma}, \boldsymbol{\Lambda} \in \mathbb{R}^{D\times D}$. Note that $\boldsymbol{\Gamma}$ is an orthonormal matrix containing the eigenvectors of $\boldsymbol{\Sigma}$ as its columns while $\boldsymbol{\Lambda}$ is a diagonal matrix and contains the respective eigenvalues. Dimensionality reduction can then be performed by matrix-multiplying the zero-centered data matrix $\tilde{\boldsymbol{X}}$ with the truncated eigenvector matrix $\boldsymbol{\Gamma}_K$ (which only includes the first K columns of $\boldsymbol{\Gamma}$), formally

$$\hat{\boldsymbol{X}} = \tilde{\boldsymbol{X}}\boldsymbol{\Gamma}_K. \tag{3.7}$$

A visual example of PCA can be found in Figure 3.5 (a).

Random Projections

While PCA finds the optimal projection for reducing dimensionality, computing this projection might be exceedingly expensive for very high dimensional data. In some scenarios, it is therefore desirable to accept larger reconstruction errors while at the same time significantly improving runtime complexity.

Based on extensive empirical results, the idea of *random projections* emerged, in which the data is reduced by transforming the data matrix $\boldsymbol{X}$ with a randomly populated reduction matrix $\boldsymbol{R}$ as follows: $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{R}$. While there is no guarantee that the chosen projection matrix is orthogonal, a large amount of almost orthogonal subspaces exist in high-dimensional spaces, which results in the fact that arbitrary vectors are sufficiently orthogonal for many use cases. The core theoretical result backing these insights is provided by the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss,

1984), stating that sufficiently high-dimensional points may be projected into a subspace of lower dimension that approximately preserves pair-wise distances between data points.

As the entries of the randomly populated reduction matrix $\boldsymbol{R}$ are only required to be zero-centered and i.i.d in order to preserve pair-wise distances, a couple of different techniques for choosing an appropriate $\boldsymbol{R}$ have been explored. We briefly explain two of these methods below.

Gaussian Random Projections When using *Gaussian random projections*, the entries of $\boldsymbol{R}$ are sampled from a zero-mean normal distribution with variance $\sigma^2 = \frac{1}{K}$, formally

$$R_{ij} \sim \mathcal{N}(0, \frac{1}{K}). \tag{3.8}$$

Sparse Random Projections When using *sparse random projections*, the entries of $\boldsymbol{R}$ are generated using the following rule set:

$$R_{ij} = \begin{cases} +\sqrt{\frac{v}{K}} & \text{with probability } \frac{1}{2v} \\ 0 & \text{with probability } 1 - \frac{1}{v} \\ -\sqrt{\frac{v}{K}} & \text{with probability } \frac{1}{2v} \end{cases} \tag{3.9}$$

where $v = \frac{1}{\sqrt{D}}$ (Achlioptas, 2003; Li et al., 2006). A visual example of a random projection matrix can be found in Figure 3.5 (b).

Autoencoders

Autoencoders (Ballard, 1987) are neural networks which are specifically designed to perform dimensionality reduction on its inputs. The neural network can be conceptually divided into two sub-networks, where the first network functions as an encoder into a reduced representation, while the second network is used to project a reduced representation back into the original representation. The encoder and decoder are then jointly optimized such that the latent representation contains as much information as possible given the size reduction constraint.

Formally, an autoencoder consists of an encoding function $\xi : \mathcal{X} \to \mathcal{H}$ and a decoding function $\psi : \mathcal{H} \to \mathcal{X}$ where the latent space $\mathcal{H}$ has lower dimensionality than the input space $\mathcal{X}$. Both functions are optimized such that the reconstruction loss is minimized, i.e. we choose ξ and ψ such that latent codes contain as much information as possible

under their size constraint:

$$\xi, \psi = \arg\min_{\xi,\psi} \|\boldsymbol{X} - (\psi \circ \xi)\boldsymbol{X}\|. \tag{3.10}$$

An exemplary autoencoder architecture can be found in Figure 3.5 (c).

Plain autoencoders impose no specific restrictions on the latent dimension other than its size being smaller than the input size. This constraint is crucial since non-reducing autoencoders could potentially learn the identity function, resulting in useless latent codes. Latent dimensions of the same or larger size than the input domain are only a reasonable choice if sparsity is explicitly enforced in the latent layers (e.g. by using the L_1 loss), a strategy applied by *sparse autoencoders* (Makhzani and Frey, 2013). Recently, a variational learning approach for latent variable modeling has been applied to autoencoders, giving rise to *variational autoencoders* (Kingma and Welling, 2013). These autoencoders however, do not merely function and compressors, but can also function as a generative model by sampling a latent vector from the learned hidden state and propagating that sample through the decoder network.

3.3.2 Outlier and Novelty Detection

Since both scenarios indicate a deviation from the expected norm, there exists a strong connection between distribution shift detection and *anomaly detection*. While both techniques try to assess whether incoming data can still be regarded coming from the same population as so far seen data, anomaly detection usually assesses this condition on every single sample. In contrast, distribution shift detection aims to capture top-level shift dynamics. In this section, we will briefly explain two different kinds of anomaly detection classes, *outlier and novelty detection*, and give pointers to popular methods for detecting unusual observations. Anomaly detection is often also referred to as out-of-distribution sample detection (see Chapter 2 for an overview of current methods).

The difference between novelty and outlier detection can be summarized as follows:

1. **Novelty detection**: The dataset used for learning the detector is assumed to be pure and the main goal is to detect abnormal observations in new incoming data.

2. **Outlier detection**: The dataset itself is already contaminated with anomalies and the main goal is to discriminate high- from low-density regions.

Most anomaly detection algorithms are applicable in both cases. Popular algorithms for anomaly detection include the local outlier factor (LOF) (Breunig et al., 2000), isolation forests (Liu et al., 2008), and the one-class SVM (Schölkopf et al., 2000).

3.4 Statistical Foundations

Machine learning is heavily tied to many other disciplines in the natural sciences realm, like mathematics, physics, and neuroscience. A particularly strong connection is held to the field of statistics, which is concerned with collecting, presenting, and analyzing various kinds of data. Large parts of machine learning are strongly dependent on basic concepts in inferential statistics, such as deriving parameter estimates using maximum likelihood estimation or Bayesian inference, constructing confidence intervals, carrying out statistical hypothesis testing, and many more.

As we apply many statistical concepts as part of this thesis, especially statistical hypothesis testing, we briefly outline some basic principles of statistics as part of this section.

3.4.1 Statistical Model

A foundational building block of statistical inference is the *statistical model*, which encodes a set of statistical assumptions about the observed data and its underlying data-generating process. Although a statistical model is often thought of as an idealized non-deterministic mathematical formulation, it allows for the derivation of many crucial concepts in inferential statistics (Cox, 2006).

Formally, a statistical model is defined as a triplet $\mathcal{M} = (\mathcal{X}, \mathcal{S}, (P_\theta)_{\theta\in\Theta})$ where $\mathcal{X}$ is the set of all possible observations (also called sample space), $\mathcal{S}$ is a σ-algebra defined over $\mathcal{X}$, and $(P_\theta)_{\theta\in\Theta}$ is either a family of probability measures or a family of probability distribution functions defined over $(\mathcal{X}, \mathcal{S})$. If $\mathcal{X}$ is countable, then we call $\mathcal{M}$ discrete with $\mathcal{S} = \mathcal{P}(\mathcal{X})$ being the power set of $\mathcal{X}$ and each P_θ is assigned a discrete density $\rho_\theta : \mathcal{X} \to [0, 1]$ with $\boldsymbol{x} \mapsto P_\theta(\boldsymbol{x})$. If $\mathcal{X}$ is uncountable (e.g. $\mathcal{X} \subset \mathbb{R}^N$), then we call $\mathcal{M}$ continuous with $\mathcal{S} = \mathcal{B}^N_\mathcal{X}$ being the Borel set of $\mathcal{X}$ and each P_θ is assigned a Lebesgue density $\rho_\theta : \mathcal{X} \to \mathbb{R}_+$. We further call a model $\mathcal{M}$ parametric (or U-dimensional) if $\Theta \subset \mathbb{R}^U$ for any given $U \in \mathbb{N}$. Note that U refers to the dimensionality of the model (i.e. the total count of parameters we wish to estimate) while D refers to the dimensionality of the data (i.e. the total count of distinct data features).

3.4.2 Statistical Hypothesis Testing

Given a statistical model $\mathcal{M} = (\mathcal{X}, \mathcal{S}, (P_\theta)_{\theta\in\Theta})$, we define a test problem as the disjoint decomposition $\Theta = \Theta_0 \cup \Theta_1$ into the null hypothesis space Θ_0 and an alternative hypothesis space Θ_1. A *statistical hypothesis test* $H_0 : \theta \in \Theta_0$ vs $H_A : \theta \in \Theta_1$ is then

defined as a function $S : \mathcal{X} \to \{0, 1\}$ with $R := \{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) = 1\}$ being the rejection region of the test and $A := \{\boldsymbol{X} \in \mathcal{X} : S(\boldsymbol{X}) = 0\}$ being the acceptance region (Lehmann and Romano, 2006). The null hypothesis H_0 is thought of as stating the norm while the alternative hypothesis H_A states a deviation from the norm which we wish to test for.

When performing a statistical test we can encounter two different error cases:

- α error: We wrongly reject the null hypothesis, i.e. $S(\boldsymbol{X}) = 1$ while $\theta \in \Theta_0$.
- β error: We wrongly accept the null hypothesis, i.e. $S(\boldsymbol{X}) = 0$ while $\theta \in \Theta_1$.

When constructing tests, we wish to keep both the α and the β error as low as possible. Unfortunately, it is not possible to simultaneously reduce both errors as a reduction of either one of the two errors generally leads to an increase in the other error. Luckily, we can get around this problem by minimizing both errors asymmetrically. By first defining an acceptable significance level $\alpha \in (0, 1)$ (also called maximum α error) and then enlisting all possible tests with that particular significance level, i.e. all tests for which $\sup_{\theta \in \Theta_0} P_\theta(R) \leq \alpha$, we can then choose the most powerful test, i.e. the test which maximizes the power function $\beta_S(\theta) := P_\theta(R) = \mathbb{E}[S]$ for all $\theta \in \Theta_1$ (which simultaneously lowers the β error).

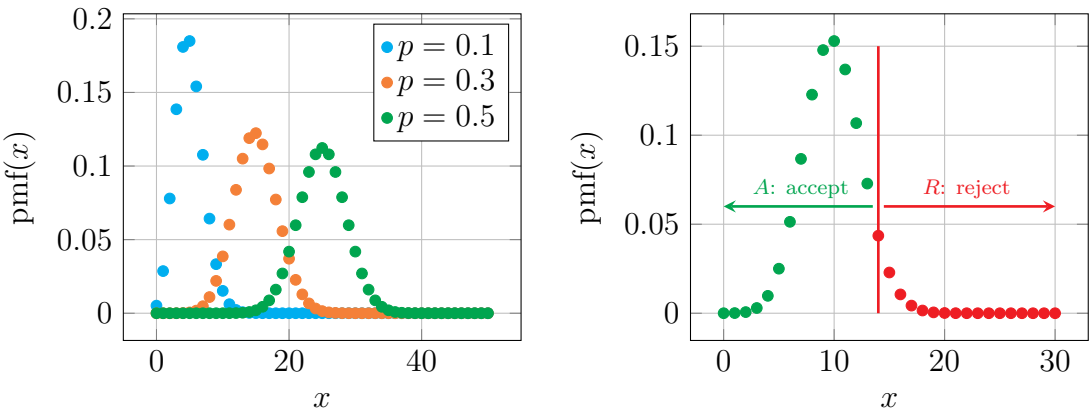
Statistical hypothesis testing is generally a multi-stage process consisting of the following sequential steps:

1. Define an appropriate statistical model $\mathcal{M}$ for the collected data.
2. Define a null hypothesis H_0 and an alternative hypothesis H_A.
3. Choose a statistical significance threshold α (usually $\alpha \in \{0.1, 0.05, 0.01\}$).
4. Choose an appropriate decision rule or test statistic S at the significance level α which has maximum power.
5. Conduct the test and make a decision.

After having established the formal basis of statistical hypothesis testing, we can now introduce a couple of specific tests which we use as part of our shift detection mechanism.

Binomial Test

The *binomial test* (Georgii, 2015) is a test which evaluates whether frequency observations from two distinct categories (usually defined as success and failure) follow a predefined theoretical distribution over the same categories. Given an absolute count of the successes



(a) Binomial distributions for different success probabilities p for a total of 50 trials.

(b) Binomial testing scenario for Bin(0.1,30).

Fig. 3.6 Binomial testing scenario.

X and the total sample size n, we can easily calculate the success probability p_0. This allows us to set up the (two-sided) hypothesis test for the unknown true success probability p as follows:

$$H_0 : p = p_0 \qquad \text{vs} \qquad H_A : p \neq p_0. \tag{3.11}$$

Under the null hypothesis, X is $\text{Bin}(p_0, n)$-distributed where

$$\text{Bin}(i|p_0, n) = P(X = i) = \binom{n}{i} p_0^i (1 - p_0)^{n-i}. \tag{3.12}$$

The binomial testing scenario is shown in Figure 3.6.

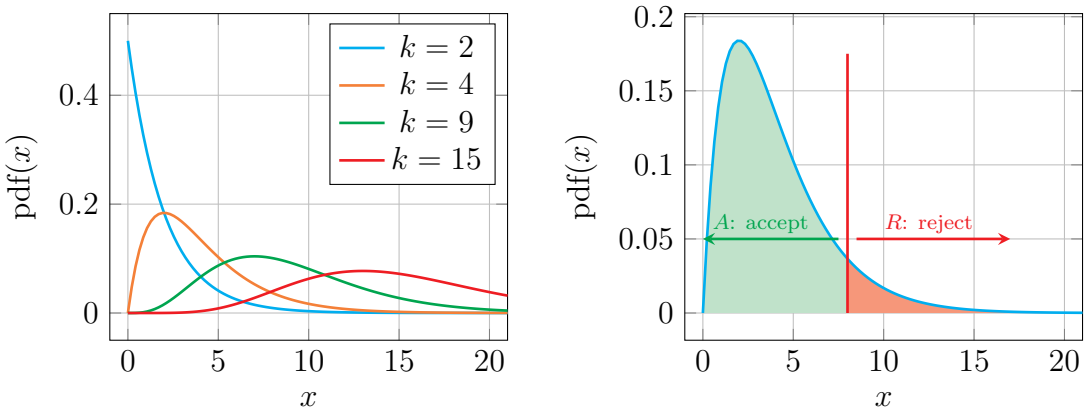
The binomial test is a *parametric tests*, which means that it assumes a specific data distribution. Specifically, the binomial test assumes that the data follows a binomial distribution. In cases where we have prior knowledge about the underlying data distribution, it is relatively straight-forward to come up with a corresponding parametric test. Indeed, parametric tests are always favorable over *non-parametric tests* if the distribution assumption holds, since it has been shown that parametric tests have more power is such scenarios (Georgii, 2015). However, observed data does not always follow a presumed distribution and some distributions are hard to estimate. For that purpose, we also introduce two non-parametric tests, which we are going to use in the upcoming chapters.

Chi-Squared (χ^2) Test

The *chi-squared test* (sometimes denoted χ^2 test) (Greenwood and Nikulin, 1996) is used to test whether some observed class-assignment frequencies roughly match the expected

Table 3.1 Chi-Squared contingency table.

Sample	Category 1	$\cdots$	Category J	$\sum$
X_1	n_{11}	$\cdots$	n_{1J}	$n_{11} + n_{12} + \cdots + n_{1J} = n_{1\bullet}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_I	n_{I1}	$\cdots$	n_{IJ}	$n_{I1} + n_{I2} + \cdots + n_{IJ} = n_{I\bullet}$
$\sum$	$n_{11} + \cdots + n_{I1} = n_{\bullet 1}$	$\cdots$	$n_{1J} + \cdots + n_{IJ} = n_{\bullet J}$	$n_{1\bullet} + n_{2\bullet} + \cdots + n_{I\bullet} = n$



(a) Chi-Squared distribution for different degrees of freedom k.

(b) Chi-Squared testing example for χ^2_4.

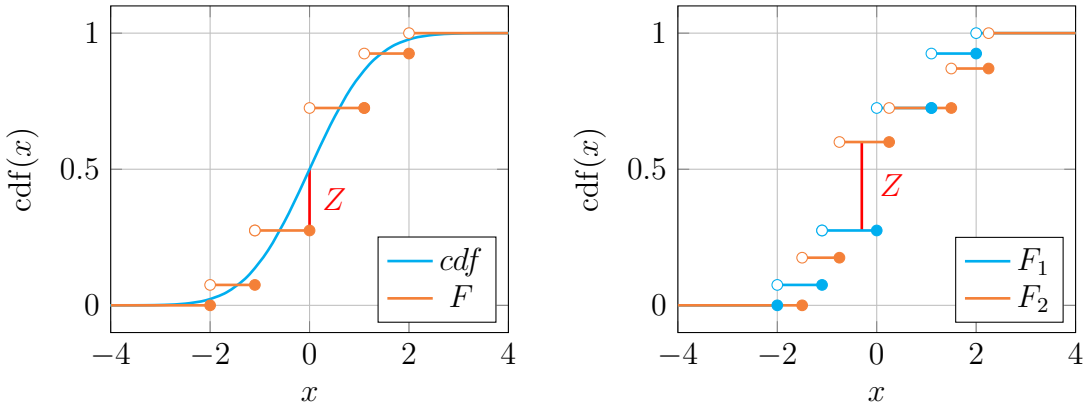
Fig. 3.7 Chi-Squared testing scenario.

class-assignment frequencies. There exist three variations of the chi-squared test: (i) the chi-squared goodness-of-fit test, (ii) the chi-squared test of homogeneity, and (iii) the test of independence. For our subsequent use cases, the chi-squared test of homogeneity is the most relevant one, which is why we introduce its most important details below. Figure 3.7 visualizes the chi-squared testing scenario.

Given I-many datasets consisting of categorical samples $\{X_1, \ldots, X_I\}$ falling into J-many categories and their corresponding categorical distribution functions $\{F_1, \ldots, F_I\}$, we wish to test

$$H_0 : F_1 = F_2 = \ldots = F_I \qquad \text{vs} \qquad H_A : \exists\ i \neq j \text{ such that } F_i \neq F_j. \tag{3.13}$$

The observed data $\{X_1, \ldots, X_I\}$ can be represented in an $I \times J$ contingency table (see Table 3.1), effectively storing the absolute class frequencies for all datasets. Next, the observed values are compared to the values which we would expect to observe under the null hypothesis and decide for a rejection if the deviations are statistically significant.



(a) Kolmogorov-Smirnov goodness of fit testing example.

(b) Kolmogorov-Smirnov two sample testing example.

Fig. 3.8 Kolmogorov-Smirnov testing scenario.

The relevant test statistic X^2 can be calculated using

$$X^2 = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(n_{ij} - E_{ij})^2}{E_{ij}} \qquad \text{with} \qquad E_{ij} = n_{i\bullet} \cdot \frac{n_{\bullet j}}{n}. \tag{3.14}$$

This test statistic is approximately chi-squared distributed with $(I-1)(J-1)$ degrees for freedom, formally $X^2 \sim \chi^2_{(I-1)(J-1)}$.

Kolmogorov-Smirnov Test

While we are going to use both the binomial test and the chi-squared test as proxies for detecting distribution shift, the *Kolmogorov-Smirnov (KS) test* (Massey Jr, 1951) is explicitly designed to test whether two probability distributions match. Similar to the chi-squared test, the KS test comes in two flavors: (i) the (1-sample) KS goodness-of-fit test which evaluates whether collected data follows a presumed distribution; and (ii) the two-sample KS test which evaluates whether two different datasets follow the same distribution. A visual comparison of the two approaches is given in Figure 3.8. For our work, we found the two-sample KS test particularly useful, which is why we explain the key workings of the KS test using the two-sample formulation.

The two-sample KS test explicitly tests for a difference in distribution between two sample sets, i.e.

$$H_0 : F_1(\boldsymbol{z}) = F_2(\boldsymbol{z}') \qquad \text{vs} \qquad H_A : F_1(\boldsymbol{z}) \neq F_2(\boldsymbol{z}') \tag{3.15}$$

where F_1 and F_2 correspond to the empirical cumulative distribution functions (ECDF) of the two distributions we wish to test for equality. The ECDF for a data vector $\boldsymbol{z}$ with I-many elements is given by

$$F(\boldsymbol{z}) = \frac{1}{I} \sum_{i=1}^{I} \mathbb{I}_{(-\infty,x]}(z_i) \tag{3.16}$$

where $\mathbb{I}_A(X) \to \{0, 1\}$ is the indicator of event A.

The KS test statistic is then given by the largest distance between the two ECDFs, formally

$$Z = \sup_{\boldsymbol{z}} |F_1(\boldsymbol{z}) - F_2(\boldsymbol{z})|. \tag{3.17}$$

The null hypothesis is going to be rejected if

$$Z > \sqrt{-\frac{1}{2} \ln \alpha} \sqrt{\frac{N_1 + N_2}{N_1 N_2}} \tag{3.18}$$

where α again corresponds to the significance level of the test and N_1 and N_2 correspond to the sample sizes of the two respective distributions.

3.4.3 Multiple Hypothesis Testing

As part of the shift detection approach we are going to propose in Chapter 4, we might have to perform multiple simultaneous statistical inferences on the same sample. Since it is increasingly more likely to draw erroneous inferences the more inferences are being made based on the same dataset, effectively inflating the global α-error, we need to correct for multiple hypothesis testing on the same sample (Rupert Jr et al., 2012).

Different ways of accounting for the multiple hypothesis testing problem have been proposed, with most approaches either bounding the *Family-Wise Error Rate (FWER)* or the *False Discovery Rate (FDR)*.

Family-Wise Error Rate (FWER)

The most stringent control in multiple hypothesis testing is given by procedures controlling the FWER, which limits the probability of making at least one false positive, formally

$$\text{FWER} = P(V \geq 1) < \alpha \tag{3.19}$$

where V is the total amount of false discoveries (Hochberg, 1987).

The Bonferroni Correction One of the easiest and at the same time most conservative corrections is given by the *Bonferroni correction*, which bounds the family-wise error rate by distributing the global α equally among the K-many tests. Given a family of hypotheses $\{H_1, \ldots, H_K\}$ and their corresponding p-values $\{p_1, \ldots, p_K\}$, the Bonferroni correction rejects the global null hypothesis if any $p_i \leq \frac{\alpha}{K}$. The shift detection procedure proposed by us makes explicit use of the Bonferroni correction.

Other approaches for bounding the FWER include but are not limited to the Šidák correction (Šidák, 1967), Tukey's range test (Tukey et al., 1949), the Holm-Bonferroni method (Holm, 1979), and Hochberg's step-up procedure (Hochberg).

False Discovery Rate (FDR)

A less stringent but more powerful alternative to the FWER is the FDR, which limits the expected proportion of false positives, formally

$$\text{FDR} = \mathbb{E}\left[\frac{V}{M}\right] < \alpha \tag{3.20}$$

where M is the total amount of discoveries. Popular approaches for bounding the FDR include but are not limited to the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) and the Benjamini-Yekutieli procedure (Benjamini et al., 2001).

Chapter 4

Distribution Shift Detection

4.1 Distribution Shifts

They key question distribution shift detection aims to answer is whether the data generating distribution of the two environments is in fact the same. Formally, we wish to determine whether the source distribution p over data points $\boldsymbol{x} \in \boldsymbol{X}$ and labels $y \in \boldsymbol{y}$ matches the target distribution q over data points $\boldsymbol{x}' \in \boldsymbol{X}'$ and labels $y' \in \boldsymbol{y}'$, formally

$$p(\boldsymbol{x}, y) \stackrel{?}{=} q(\boldsymbol{x}', y'). \tag{4.1}$$

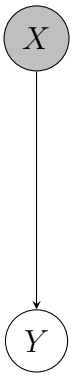
Since test data is unlabeled in real-life settings, it is usually not possible to directly assess whether the joint distribution between data points and true labels differs between the two environments. Instead, we often only have estimates $\hat{y}'$ of the true label y' at our disposal, altering the problem formulation as follows:

$$p(\boldsymbol{x}, y) \stackrel{?}{=} q(\boldsymbol{x}', \hat{y}'). \tag{4.2}$$

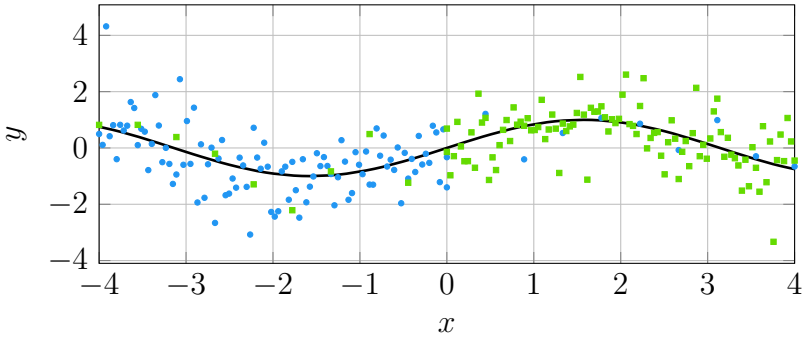
If data was collected in an unsupervised fashion (or when we simply intend to analyze the data in an unsupervised manner), the shift detection question simplifies to

$$p(\boldsymbol{x}) \stackrel{?}{=} q(\boldsymbol{x}'). \tag{4.3}$$

Next, we introduce some basic shifts. To that end, we borrow heavily from Sugiyama et al. (2017). More complex shifts, such as sample selection bias, domain shift, and source component shift are described extensively in their work.

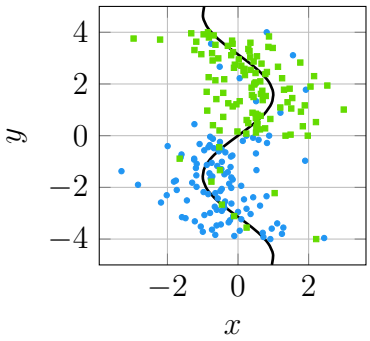


(a) Covariate shift causal graphical model.

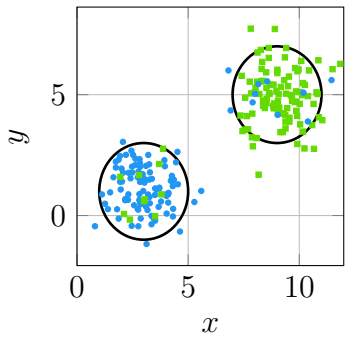


(b) Covariate shift example.

(c) Label shift causal graphical model.



(d) Label shift example (regr.).



(e) Label shift example (classif.).

Fig. 4.1 Covariate and label shift visualized (Sugiyama et al., 2017). If the distribution of a variable may change between the source and the target domain the respective node is shown in gray in the causal graphical model. Blue circles in the example plots indicate source samples, green squares indicate target samples.

4.1.1 Covariate Shift

Under the *covariate shift* model, only the distribution over covariates $\boldsymbol{x}$ (i.e. the input features) changes between the two domains, i.e. $p(\boldsymbol{x}) \neq q(\boldsymbol{x})$, while the conditional label distribution remains fixed, i.e. $p(y|\boldsymbol{x}) = q(y|\boldsymbol{x})$. As is expected, this shift in the covariates also leads to a shift in the joint probability distribution:

$$[p(\boldsymbol{x}) \neq q(\boldsymbol{x}) \wedge p(y|\boldsymbol{x}) = q(y|\boldsymbol{x})] \Rightarrow p(y|\boldsymbol{x})p(\boldsymbol{x}) \neq q(y|\boldsymbol{x})q(\boldsymbol{x}) \Rightarrow p(\boldsymbol{x}, y) \neq q(\boldsymbol{x}, y) \quad (4.4)$$

Both a causal graphical model as well as an example of covariate shift is shown in Figure 4.1 (a) and (b).

4.1.2 Label Shift

Under the *label shift* model, only the distribution over labels y changes between the two domains, i.e. $p(y) \neq q(y)$, while the conditional covariate distribution remains fixed, i.e. $p(\boldsymbol{x}|y) = q(\boldsymbol{x}|y)$. As is expected, this shift in the labels also leads to a shift in the joint probability distribution:

$$[p(y) \neq q(y) \wedge p(\boldsymbol{x}|y) = q(\boldsymbol{x}|y)] \Rightarrow p(\boldsymbol{x}|y)p(y) \neq q(\boldsymbol{x}|y)q(y) \Rightarrow p(\boldsymbol{x}, y) \neq q(\boldsymbol{x}, y) \quad (4.5)$$

This type of shift is sometimes also referred to as *prior probability shift* or *target shift*. Both a causal graphical model as well as two examples of label shift is shown in Figure 4.1 (c), (d), and (e).

4.1.3 Concept Drift

Under the *concept drift* model, either one of the conditional distribution over the covariates or the labels changes, i.e. $p(y|\boldsymbol{x}) \neq q(y|\boldsymbol{x})$ or $p(\boldsymbol{x}|y) \neq q(\boldsymbol{x}|y)$, while the marginals remain fixed, i.e. $p(\boldsymbol{x}) = q(\boldsymbol{x})$ or $p(y) = q(y)$. As is expected, this shift also leads to a shift in the joint probability distribution. It affects the joint distribution as follows in the case of a changing label conditional

$$[p(y|\boldsymbol{x}) \neq q(y|\boldsymbol{x}) \wedge p(\boldsymbol{x}) = q(\boldsymbol{x})] \Rightarrow p(y|\boldsymbol{x})p(\boldsymbol{x}) \neq q(y|\boldsymbol{x})q(\boldsymbol{x}) \Rightarrow p(\boldsymbol{x}, y) \neq q(\boldsymbol{x}, y) \quad (4.6)$$

and as follows in the case of a changing covariate conditional:

$$[p(\boldsymbol{x}|y) \neq q(\boldsymbol{x}|y) \wedge p(y) = q(y)] \Rightarrow p(\boldsymbol{x}|y)p(y) \neq q(\boldsymbol{x}|y)q(y) \Rightarrow p(\boldsymbol{x}, y) \neq q(\boldsymbol{x}, y) \quad (4.7)$$

4.1.4 The Need for Shift Detection

As noted by Sugiyama et al. (2017), there is an inherent need for methods to first detect the mere presence of distribution shift and, as a consequence, characterize the shift in a second step. While methods explicitly developed to address covariate or label shift are usually favorable to use since their explicit assumption allows them to derive mathematically sound algorithms for addressing shift, they can lead to false predictions if these assumptions are not met. Also, pure covariate or label shifts rarely occur in real life settings, which is why techniques designed to correct for only one type of shift has only limited applicability in practice. At the same time, it would be desirable for a general purpose detection scheme to seamlessly fit into the existing machine learning

pipeline. As we will see in the subsequent sections, our proposed method is capable of achieving both of these desiderata.

4.2 Shift Detection on Dimensionality-Reduced Representations

Given labeled data $\{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\} \sim p$ and unlabeled data $\{\boldsymbol{x}'_1, ..., \boldsymbol{x}'_m\} \sim q$, our task is to determine whether $p(\boldsymbol{x})$ equals $q(\boldsymbol{x}')$:

$$H_0 : p(\boldsymbol{x}) = q(\boldsymbol{x}') \qquad \text{vs} \qquad H_A : p(\boldsymbol{x}) \neq q(\boldsymbol{x}'). \tag{4.8}$$

Chiefly, we explore the following design considerations: (i) what **representation** to run the test on; (ii) which **two-sample test** to run; (iii) when the representation is multidimensional; whether to run **multivariate or multiple univariate two-sample tests**; and (iv) **how to combine** their results.

4.2.1 Dimensionality Reduction

We now introduce the multiple dimensionality reduction (DR) techniques that we compare vis-a-vis their effectiveness in shift detection (in concert with two-sample testing). Note that absent assumptions on the data, these mappings, which reduce the data dimensionality from D to K (with $K \ll D$), are in general surjective, with many inputs mapping to the same output. Thus, it is trivial to construct pathological cases where the distribution of inputs shifts while the distribution of low-dimensional latent representations remains fixed, yielding false negatives. However, we speculate that in a non-adversarial setting, such shifts may be exceedingly unlikely. Thus our approach is (i) empirically motivated; and (ii) not put forth as a defense against worst-case adversarial attacks.

No Reduction (*NoRed* ○) To justify the use of any DR technique, our default baseline is to run tests on the original raw features.

Principal Components Analysis (*PCA* ⬡) Principal components analysis is a standard tool that finds an optimal orthogonal transformation matrix $\boldsymbol{R}$ such that points are linearly uncorrelated after transformation. This transformation is learned in such a way that the first principal component accounts for as much of the variability in the dataset as possible, and that each succeeding principal component captures as much

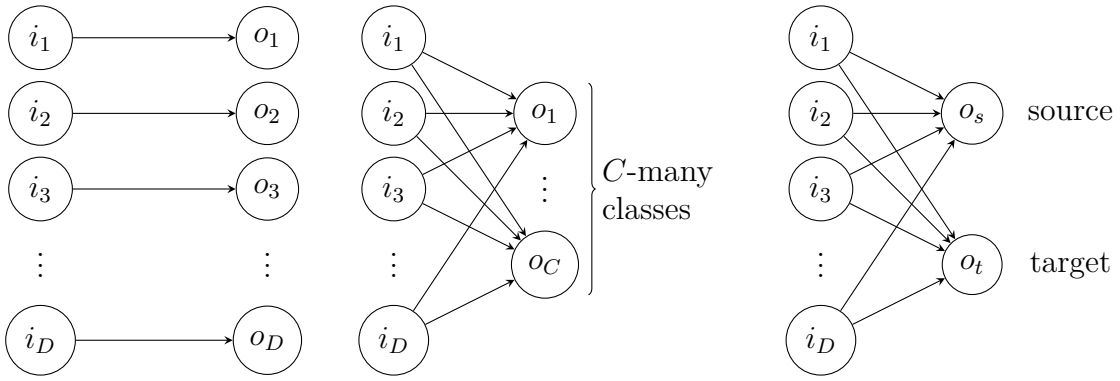
of the remaining variance as possible subject to the constraint that it be orthogonal to the preceding components. Formally, we wish to learn $\boldsymbol{R}$ given $\boldsymbol{X}$ under the mentioned constraints such that $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{R}$ yields a more compact data representation.

Sparse Random Projection (*SRP* ⬠) Since computing the optimal transformation might be expensive in high dimensions, random projections are a popular DR technique which trade a controlled amount of accuracy for faster processing times. Specifically, we make use of sparse random projections, a more memory- and computationally-efficient modification of standard Gaussian random projections. Formally, we generate a random projection matrix $\boldsymbol{R}$ and use it to reduce the dimensionality of a given data matrix $\boldsymbol{X}$, such that $\hat{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{R}$.

Autoencoders (*TAE* ◇ and *UAE* □) We compare the above-mentioned linear models to non-linear reduced-dimension representations using both *trained* (TAE) and *untrained* autoencoders (UAE). Formally, an autoencoder consists of an encoder function $\xi : \mathcal{X} \to \mathcal{H}$ and a decoder function $\psi : \mathcal{H} \to \mathcal{X}$ where the latent space $\mathcal{H}$ has lower dimensionality than the input space $\mathcal{X}$. As part of the training process, both the encoding function ξ and the decoding function ψ are learned jointly to reduce the reconstruction loss: $\xi, \psi = \arg\min_{\xi,\psi} \|\boldsymbol{X} - (\psi \circ \xi)\boldsymbol{X}\|^2$.

Label Classifiers (*BBSDs* ◁ and *BBSDh* ▷) Motivated by recent results achieved by black box shift detection (BBSD) (Lipton et al., 2018), we also propose to use the outputs of a (deep network) *label classifier* trained on source data as our dimensionality-reduced representation. We explore variants using either the softmax outputs (BBSDs) or the hard-thresholded predictions (BBSDh) for subsequent two-sample testing. Since both variants provide differently sized output (with BBSDs providing an entire softmax vector and BBSDh providing a one-dimensional class prediction), different statistical tests are carried out on these representations. Also note that while we do not have access to either true or estimated labels in the other dimensionality reduction methods, we can predict labels using the trained label classifier, which is why our formal testing setup for BBSD changes to

$$H_0 : p(\boldsymbol{x}, y) = q(\boldsymbol{x}', \hat{y}') \qquad \text{vs} \qquad H_A : p(\boldsymbol{x}, y) \neq q(\boldsymbol{x}', \hat{y}'). \tag{4.9}$$



(a) No reduction example. (b) Label classfier example. (c) Difference classifier example.

Fig. 4.2 Additional dimensionality reduction methods explored in this study.

Domain Classifier (*Classif* ×) Here, we attempt to detect shift by explicitly training a *domain classifier* to discriminate between data from source and target domains. To this end, we partition both the source data and target data into two halves, using the first to train a domain classifier to distinguish source (class 0) from target (class 1) data. We then apply this model to the second half conducting a significance test to determine if the classifier's performance is different from random chance.

4.2.2 Statistical Hypothesis Testing

The DR techniques each yield a representation, either uni- or multi-dimensional, and either continuous or discrete, depending on the method. The next step is to choose a suitable statistical hypothesis test for each of these representations.

Multivariate Two-Sample Tests: Maximum Mean Discrepancy (MMD): For all multi-dimensional representations, we evaluate the *Maximum Mean Discrepancy*, a popular kernel-based technique for multivariate two-sample testing. MMD allows us to distinguish between two probability distributions p and q based on the mean embeddings $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$ of the distributions in a reproducing kernel Hilbert space $\mathcal{F}$, formally

$$\mathrm{MMD}(\mathcal{F}, p, q) = ||\boldsymbol{\mu}_p - \boldsymbol{\mu}_q||^2_{\mathcal{F}}. \tag{4.10}$$

Given samples from both distributions, we can calculate an unbiased estimate of the squared MMD statistic as follows

$$\begin{aligned}\mathrm{MMD}^2 =& \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \kappa(\boldsymbol{x}'_i, \boldsymbol{x}'_j) \\ & - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \kappa(\boldsymbol{x}_i, \boldsymbol{x}'_j)\end{aligned} \tag{4.11}$$

where we use a squared exponential kernel $\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\frac{1}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}$. A p-value can then be obtained by carrying out a permutation test on the resulting kernel matrix.

Multiple Univariate Testing: Kolmogorov-Smirnov (KS) Test + Bonferroni Correction: As a simple baseline alternative to MMD, we consider the approach consisting of testing each of the K dimensions separately (instead testing over all dimensions jointly). Here, for continuous data, we adopt the Kolmogorov-Smirnov (KS) test, a non-parametric test whose statistic is calculated by computing the largest difference Z of the cumulative density functions (CDFs) over all values $\boldsymbol{z}$ as follows:

$$Z = \sup_{\boldsymbol{z}} |F_p(\boldsymbol{z}) - F_q(\boldsymbol{z})| \tag{4.12}$$

where F_p and F_q are the empirical CDFs of the source and target data, respectively. Under the null hypothesis, Z follows the Kolmogorov distribution.

Since we carry out a KS test on each of the K components, we must subsequently combine the p-values from each test, raising the issue of multiple hypothesis testing. As we cannot make strong assumptions about the (in)dependence among the tests, we rely on a conservative aggregation method, notably the Bonferroni correction (Bland and Altman, 1995), which rejects the null hypothesis if the minimum p-value among all tests is less than α/K (where α is the significance level of the test). While several less conservative aggregations methods have been proposed (Heard and Rubin-Delanchy, 2018; Loughin, 2004; Simes, 1986; Vovk and Wang, 2018; Zaykin et al., 2002), they typically require assumptions on the dependencies among the tests. Moreover, even using the conservative test, in our experiments, the univariate approach generally outperformed kernel two-sample testing given identical representations (see Chapter 6).

Categorical Testing: Chi-Squared Test: For the hard-thresholded label classifier (BBSDh), we employ Pearson's chi-squared test, a parametric tests designed to evaluate whether the frequency distribution of certain events observed in a sample is consistent with

a particular theoretical distribution. Specifically, we use a test of homogeneity between the class distributions (expressed in a contingency table) of source and target data. The testing problem can be formalized as follows: Given a contingency table with 2 rows (one for absolute source and one for absolute target class frequencies) and C columns containing observed counts O_{ij}, the expected frequency under the independence hypothesis for a particular cell is $E_{ij} = N_{\text{sum}} p_{i\bullet} p_{\bullet j}$ with N_{sum} being the sum of all cells in the table, $p_{i\bullet} = \frac{O_{i\bullet}}{N_{\text{sum}}} = \sum_{j=1}^{c} \frac{O_{ij}}{N_{\text{sum}}}$ being the fraction of row totals, and $p_{\bullet j} = \frac{O_{\bullet j}}{N_{\text{sum}}} = \sum_{i=1}^{r} \frac{O_{ij}}{N_{\text{sum}}}$ being the fraction of column totals. The relevant test statistic X^2 can be computed as

$$X^2 = \sum_{i=1}^{2} \sum_{j=1}^{C} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{4.13}$$

which, under the null hypothesis, follows a chi-squared distribution with $C - 1$ degrees of freedom: $X^2 \sim \chi^2_{C-1}$.

Binomial Testing: For the domain classifier, we simply compare its accuracy (acc) on held-out data to random chance via a binomial test. Formally, we set up a testing problem $H_0 : \text{acc} = 0.5$ vs $H_A : \text{acc} \neq 0.5$. Under the null hypothesis, the accuracy of the classifier follows a binomial distribution: $\text{acc} \sim \text{Bin}(N_{\text{samp}}, 0.5)$, where N_{samp} corresponds to the number of held-out samples.

4.2.3 Obtaining Most Anomalous Samples

As our detection framework does not detect outliers but rather aims at capturing top-level shift dynamics, it is not possible for us to decide whether any given sample is in- or out-of-distribution. However, we can still provide an indication of what typical samples from the shifted distribution look like by harnessing domain assignments from the domain classifier. Specifically, we can identify the exemplars which the classifier was most confident in assigning to the target domain. Hence, whenever the binomial test signals a statistically significant accuracy deviation from chance, we can use use the domain classifier to obtain the most anomalous samples and present them to the user.

In contrast to the domain classifier, the other shift detectors do not base their shift detection potential on explicitly deciding which domain a single sample belongs to, instead comparing entire distributions against each other. While we did explore initial ideas on identifying samples which if removed would lead to a large increase in the overall p-value, the results we obtained were unremarkable.

4.2.4 Determining the Malignancy of a Shift

Theoretically, absent further assumptions, distribution shifts can cause arbitrarily severe degradation in performance. However, in practice distributions shift constantly, and often these changes are benign. Practitioners should therefore be interested in distinguishing malignant shifts that damage predictive performance from benign shifts that negligibly impact performance. Although prediction quality can be assessed easily on source data on which the black-box model $f(\cdot)$ was trained, we are not able compute the target error directly without labels. We therefore explore heuristic methods for approximating the target performance through the following two methods:

1) **Difference classifier assignments:** In case no prior knowledge about the nature of the shift is available, the black-box model's accuracy on the labeled top anomalous samples can serve as a standalone *implicit* characterization of the shift. Also, since the difference classifier picks up on the most noticeable deviations, the label classifier's accuracy on the top anomalous samples functions as a worst-case bound on the model's accuracy under the identified shift type.

2) **Domain expert:** Alternatively, we can get hints about the target accuracy by evaluating the classifier on held-out source data that has been *explicitly* perturbed by a function determined by a domain expert to be representative of the sort of perturbations expected to occur at test time. The better this function captures the kinds of perturbations encountered in the target domain, the better the clue given by this method. The domain expert can further use the predictions yielded by the difference classifier to gain additional insights into the qualitative nature of the shift.

Chapter 5

Experimental Setup

5.1 General setup

Our experiments were carried out on the MNIST ($N_{\text{tr}} = 50000$; $N_{\text{val}} = 10000$; $N_{\text{te}} = 10000$; $D = 28 \times 28 \times 1$; $C = 10$ classes) (LeCun et al., 1998) and CIFAR-10 ($N_{\text{tr}} = 40000$; $N_{\text{val}} = 10000$; $N_{\text{te}} = 10000$; $D = 32 \times 32 \times 3$; $C = 10$ classes) (Krizhevsky and Hinton, 2009) image datasets. For the autoencoder (UAE & TAE) experiments, we employ a convolutional architecture with 3 convolutional layers and 1 fully-connected layer. For both the label and the domain classifier we use a ResNet-18 (He et al., 2016). We train all networks (TAE, BBSDs, BBSDh, Classif) using stochastic gradient descent with momentum in batches of 128 examples over 200 epochs with early stopping.

For PCA, SRP, UAE, and TAE, we reduce dimensionality to $K = 32$ latent dimensions, which for PCA explains roughly 80% of the variance in the CIFAR-10 dataset. The label classifier BBSDs reduces dimensionality to the number of classes C. Both the hard label classifier BBSDh and the domain classifier Classif reduce dimensionality to a one-dimensional class prediction, where BBSDh predicts label assignments and Classif predicts domain assignments.

To challenge our detection methods, we simulate a variety of shifts, affecting both the covariates and the label proportions. For all shifts, we evaluate the various methods' abilities to detect shift at a significance level of $\alpha = 0.05$. We also include the no-shift case to check against false positives. We randomly split all of the data into training, validation, and test sets according to the indicated proportions N_{tr}, N_{val}, and N_{te} and then apply a particular shift to the test set only. In order to qualitatively quantify the robustness of our findings, shift detection performance is averaged over a total of 5 random splits, which ensures that we apply the same type of shift to different subsets of the data. The selected training data used to fit the DR methods is kept constant

across experiments with only the splits between validation and test changing across the random runs. Note that DR methods are learned using training data, while shift detection is being performed on dimensionality-reduced representations of the validation and the test set. We evaluate the models with various amounts of samples from the test set $s \in \{10, 20, 50, 100, 200, 500, 1000, 10000\}$. Because of the unfavorable dependence of kernel methods on the dataset size, we run these methods only up until 1000 target samples have been acquired.

5.2 Shift Simulation

For each shift type (as appropriate) we explored three levels of shift intensity (e.g. the magnitude of added noise) and various percentages of affected data $\delta \in \{0.1, 0.5, 1.0\}$. Specifically, we explore the following types of shifts:

(i) **Adversarial shift (*adv_shift*)**: We turn a given percentage δ of test samples into adversarial examples via the fast gradient sign method (Goodfellow et al., 2014);

(ii) **Knock-out shift (*ko_shift*)**: We remove a fraction δ of data points from class $c = 0$, creating class imbalance (Lipton et al., 2018);

(iii) **Gaussian noise shift (*gn_shift*)**: We corrupt covariates of a fraction δ of test set samples by Gaussian noise centered on the datapoint with standard deviation $\sigma \in \{1, 10, 100\}$ (denoted *small_gn_shift*, *medium_gn_shift*, and *large_gn_shift*);

(iv) **Image shift (*img_shift*)**: We also consider more natural shifts to images, modifying a fraction δ of images with combinations of random amounts of rotations $\{10, 40, 90\}$, (x, y)-axis-translation percentages $\{0.05, 0.2, 0.4\}$, as well as zoom-in percentages $\{0.1, 0.2, 0.4\}$ (denoted *small_img_shift*, *medium_img_shift*, and *large_img_shift*);

(v) **Image shift + knock-out shift (*medium_img_shift+ ko_shift*)**: In addition to pure covariate or label shifts, we also explore test sets which are affected by both shifts at the same time. Here, we apply a fixed medium image shift with $\delta_1 = 0.5$ and a variable knock-out shift δ;

(vi) **Only-zero shift + image shift (*only_zero_shift+ medium_img_shift*)**: Here, we only include images from class $c = 0$ in combination with a variable medium image shift affecting only a fraction δ of the data;

(vii) **Original splits**: We evaluate our detectors on the original source/target splits provided by the creators of MNIST, CIFAR-10, Fashion MNIST, and SVHN datasets (assumed to be i.i.d.);

(viii) **Domain adaptation datasets**: Finally, we also consider data from the domain adaptation task transferring from MNIST (source) to USPS (target) ($N_{\text{tr}} = N_{\text{val}} = N_{\text{te}} = 1000$; $D = 16 \times 16 \times 1$; $C = 10$ classes) (Long et al., 2013) as well as data from the COIL-10 dataset ($N_{\text{tr}} = N_{\text{val}} = N_{\text{te}} = 2400$; $D = 32 \times 32 \times 3$; $C = 10$ classes, a modification of the COIL-100 dataset which only includes the first 10 classes) (Nene et al., 1996) where images between 0° and 175° are sampled by the source and images between 180° and 355° are sampled by the target distribution.

Chapter 6

Discussion

We now discuss the salient findings from our empirical investigation.

6.1 Univariate vs multivariate tests

We first evaluate whether we can detect shifts more easily using multiple univariate tests and aggregating their results via the Bonferroni correction or by using multivariate kernel tests. We were surprised to find that across DR methods, aggregated univariate tests outperformed multivariate tests (see Tables 6.1, 6.2, 6.3, and 6.4).

6.2 Dimensionality reduction methods

For each testing method and experimental setting, we evaluate which DR technique is best suited for shift detection. In the multiple-univariate-testing case (and thus overall), BBSDs was the best-performing DR method. In the multivariate-testing case, the autoencoders (UAE and TAE) performed best. In both cases, these methods consistently outperformed others across sample sizes. The domain classifier performs badly in the low-sample regime (≤ 100 samples), but catches up as more samples are obtained. Noticeably, the multivariate test performs poorly in the no reduction case, especially on CIFAR-10, perhaps owing to the high dimensionality of the dataset. Table 6.1 summarizes these results.

Table 6.1 Detection accuracy of different dimensionality reduction techniques across all simulated shifts on MNIST and CIFAR-10. **Green bold** entries indicate the best DR method at a given sample size, *red italic* the worst. Underlined entries indicate accuracy values larger than 0.5.

Test	DR	Number of samples from test							
		10	20	50	100	200	500	1,000	10,000
Univ. tests	NoRed	0.18	0.25	0.36	0.39	0.45	0.49	0.57	0.70
	PCA	0.13	0.22	0.25	0.30	0.35	0.41	*0.46*	0.58
	SRP	0.18	0.21	0.27	0.32	0.37	0.46	0.53	0.61
	UAE	0.20	0.25	0.33	0.42	0.48	0.54	0.65	**0.74**
	TAE	0.20	0.26	0.37	0.45	0.44	0.53	0.58	0.67
	BBSDs	**0.29**	**0.38**	**0.43**	**0.49**	**0.55**	**0.61**	**0.66**	0.72
χ^2	BBSDh	0.14	0.18	0.23	0.26	0.32	0.41	0.47	*0.48*
Bin	*Classif*	*0.04*	*0.09*	*0.10*	*0.10*	*0.28*	*0.38*	0.47	0.66
Multiv. tests	*NoRed*	*0.00*	*0.00*	*0.00*	*0.04*	0.15	0.15	*0.18*	–
	PCA	0.01	0.05	0.09	0.11	0.15	0.22	0.28	–
	SRP	*0.00*	*0.00*	0.03	0.08	*0.13*	*0.14*	0.19	–
	UAE	**0.19**	**0.26**	**0.36**	0.36	0.42	0.49	**0.59**	–
	TAE	**0.19**	0.22	**0.36**	**0.44**	**0.46**	**0.50**	0.58	–
	BBSDs	0.16	0.19	0.23	0.31	0.30	0.43	0.48	–

6.3 Shift types

Table 6.2 lists shift detection accuracy values for each distinct shift as an increasing amount of samples is obtained from the target domain. Specifically, we see that *large_gn_shift*, *medium_gn_shift*, *large_img_shift*, *medium_img_shift+ko_shift*, and *only_zero_shift+medium_img_shift* are easily detectable even with few samples, while *small_gn_shift*, *medium_gn_shift*, *adv_shift*, and *ko_shift* are hard to detect even with many samples. With a few exceptions, the best DR technique (BBDSs for multiple univariate tests, UAE & TAE for multivariate tests) is significantly faster and more accurate at detecting shift than the mean of all dimensionality reduction methods.

6.4 Shift intensity

Based on the results in Table 6.3, we can conclude that the small shifts (*small_gn_shift*, *small_img_shift*, and *ko_shift*) are harder to detect than medium shifts (*medium_gn_shift*, *medium_img_shift*, and *adv_shift*) which in turn are harder to detect than large shifts

Table 6.2 Detection accuracy of different shifts on MNIST and CIFAR-10. The first entry in each cell shows to the accuracy of the best DR technique (univariate: BBSDs, multivariate: mean of UAE and TAE), while the value in parentheses shows the accuracy across all dimensionality reduction techniques. **Green bold** shifts are identified as harmless, *red italic* shifts as harmful. Underlined entries indicate accuracy values larger than 0.5.

Test type	Simulated shift type	Number of samples from test							
		10	20	50	100	200	500	1,000	10,000
Univariate tests	**small_gn_shift**	0.00 (0.07)	0.03 (0.07)	0.07 (0.07)	0.10 (0.12)	0.10 (0.12)	0.10 (0.16)	0.10 (0.19)	0.10 (0.23)
	medium_gn_shift	0.00 (0.09)	0.03 (0.09)	0.10 (0.13)	0.10 (0.16)	0.14 (0.19)	0.24 (0.25)	0.24 (0.31)	0.38 (0.41)
	large_gn_shift	0.45 (0.34)	0.52 (0.37)	0.59 (0.51)	0.72 (0.53)	0.83 (0.64)	0.86 (0.72)	0.97 (0.79)	1.00 (0.87)
	small_img_shift	0.14 (0.05)	0.21 (0.08)	0.31 (0.13)	0.45 (0.19)	0.59 (0.26)	0.59 (0.30)	0.69 (0.38)	0.97 (0.58)
	medium_img_shift	0.34 (0.13)	0.55 (0.25)	0.66 (0.31)	0.79 (0.36)	0.83 (0.41)	0.90 (0.54)	0.93 (0.60)	1.00 (0.83)
	large_img_shift	0.48 (0.23)	0.66 (0.32)	0.72 (0.40)	0.83 (0.49)	0.83 (0.62)	0.93 (0.73)	1.00(0.82)	1.00 (0.91)
	adv_shift	0.07 (0.08)	0.10 (0.11)	0.10 (0.14)	0.10 (0.13)	0.17 (0.17)	0.21 (0.20)	0.34 (0.31)	0.45 (0.38)
	ko_shift	0.00 (0.04)	0.00 (0.04)	0.00 (0.04)	0.07 (0.08)	0.10 (0.12)	0.34 (0.22)	0.48 (0.34)	0.72 (0.60)
	medium_img_shift+ko_shift	0.45 (0.14)	0.66 (0.18)	0.79 (0.31)	1.00 (0.46)	1.00 (0.55)	1.00 (0.67)	1.00 (0.76)	1.00 (0.97)
	only_zero_shift+medium_img_shift	1.00 (0.52)	1.00 (0.79)	1.00 (0.90)	1.00 (0.91)	1.00 (0.98)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)
Multivariate tests	small_gn_shift	0.02 (0.01)	0.02 (0.02)	0.05 (0.06)	0.05 (0.08)	0.08 (0.08)	0.10 (0.09)	0.12 (0.09)	–
	medium_gn_shift	0.03 (0.01)	0.08 (0.04)	0.17 (0.08)	0.19 (0.12)	0.29 (0.13)	0.32 (0.18)	0.39 (0.21)	–
	large_gn_shift	0.51 (0.20)	0.53 (0.22)	0.71 (0.36)	0.75 (0.37)	0.81 (0.44)	0.88 (0.51)	0.97 (0.58)	–
	small_img_shift	0.12 (0.04)	0.15 (0.06)	0.20 (0.08)	0.29 (0.12)	0.39 (0.16)	0.41 (0.21)	0.51 (0.28)	–
	medium_img_shift	0.29 (0.12)	0.31 (0.12)	0.36 (0.15)	0.39 (0.21)	0.44 (0.31)	0.54 (0.36)	0.66 (0.45)	–
	large_img_shift	0.29 (0.12)	0.32 (0.16)	0.36 (0.25)	0.46 (0.30)	0.59 (0.42)	0.75 (0.47)	0.81 (0.53)	–
	adv_shift	0.03 (0.03)	0.08 (0.03)	0.20 (0.08)	0.22 (0.08)	0.22 (0.09)	0.25 (0.11)	0.32 (0.14)	–
	ko_shift	0.05 (0.06)	0.10 (0.06)	0.17 (0.08)	0.19 (0.12)	0.20 (0.12)	0.24 (0.16)	0.34 (0.23)	–
	medium_img_shift+ko_shift	0.17 (0.08)	0.22 (0.10)	0.39 (0.16)	0.51 (0.22)	0.54 (0.24)	0.58 (0.37)	0.78 (0.50)	–
	only_zero_shift+medium_img_shift	0.36 (0.25)	0.75 (0.40)	0.86 (0.50)	1.00 (0.63)	1.00 (0.75)	1.00 (0.76)	1.00 (0.82)	–

Table 6.3 Detection accuracy for small, medium, and large simulated shifts on MNIST and CIFAR-10 using univariate tests + Bonferroni correction on BBSDs. Reported accuracy values are results of the best DR technique (univariate: BBSDs, multivariate: mean of UAE and TAE). Underlined entries indicate accuracy values larger than 0.5.

Test	Intensity	Number of samples from test							
		10	20	50	100	200	500	1,000	10,000
Univ.	Small	0.06	0.11	0.13	0.15	0.23	0.3	0.38	0.52
	Medium	0.17	0.29	0.37	0.42	0.47	0.56	0.58	0.69
	Large	0.59	0.7	0.76	0.88	0.91	0.94	0.99	1.00
Multiv.	Small	0.07	0.08	0.18	0.20	0.21	0.24	0.32	–
	Medium	0.17	0.20	0.27	0.28	0.37	0.44	0.53	–
	Large	0.34	0.44	0.59	0.67	0.73	0.80	0.89	–

(*large_gn_shift*, *large_img_shift*, *medium_img_shift+ko_shift*, and *only_zero_shift+ medium_img_shift*). Specifically, we see that large shifts can on average already be detected with better than chance accuracy at only 10 samples in the multiple univariate testing setting. Medium and small shifts require orders of magnitude more samples in order to achieve similar accuracy.

6.5 Test sample size

As we can clearly see from the results in Tables 6.1, 6.2, 6.3, and 6.4, the more samples we obtain from the target domain, the better we can detect shifts.

6.6 Most Anomalous Samples and Shift Malignancy

Across all experiments, we observed that the most different and most similar examples returned by the difference classifier are useful in characterizing the shift qualitatively (see Appendix A for detailed documentation). Furthermore, we can successfully distinguish malignant from benign shifts using labeled top anomalous samples by using the framework proposed in Section 4.2.4.

Table 6.4 Detection accuracy for low (10%), medium (50%), and high (100%) percentages of perturbed target samples across all shifts on MNIST and CIFAR-10. Reported accuracy values are results of the best dimensionality reduction technique (univariate: BBSDs, multivariate: mean of UAE and TAE). Underlined entries indicate accuracy values larger than 0.5.

Test	Percentage	Number of samples from test							
		10	20	50	100	200	500	1,000	10,000
Univ.	10%	0.19	0.20	0.25	0.34	0.36	0.41	0.48	0.56
	50%	0.28	0.42	0.49	0.56	0.60	0.61	0.70	0.80
	100%	0.41	0.54	0.55	0.60	0.70	0.81	0.82	0.82
Multiv.	10%	0.14	0.15	0.24	0.27	0.29	0.31	0.41	–
	50%	0.20	0.22	0.39	0.41	0.45	0.54	0.61	–
	100%	0.26	0.38	0.46	0.54	0.60	0.68	0.76	–

6.7 Original splits

According to our tests, the original splits from the MNIST dataset appear to exhibit a dataset shift. After inspecting the most anomalous samples returned by the difference classifier, we observed that many of these samples depicted the digit 6. A mean-difference plot (see Figure 6.1) between sixes from the training set and sixes from the test set revealed that the training instances are rotated slightly to the right, while the test samples are drawn more open and centered. To back up this claim even further, we also carried out a two-sample KS test between the two sets of sixes in the input space and found that the two sets can conclusively be regarded as different with a p-value of $2.7 \cdot 10^{-10}$, significantly undercutting the respective Bonferroni threshold of $6.3 \cdot 10^{-5}$. While this particular shift does not look practically significant to the human eye (and is also declared harmless by our malignancy detector), this result however still shows that the original MNIST split is not truly i.i.d.

6.8 Individual Examples

While full results are presented in the supplementary material, we briefly present two illustrative results in detail:

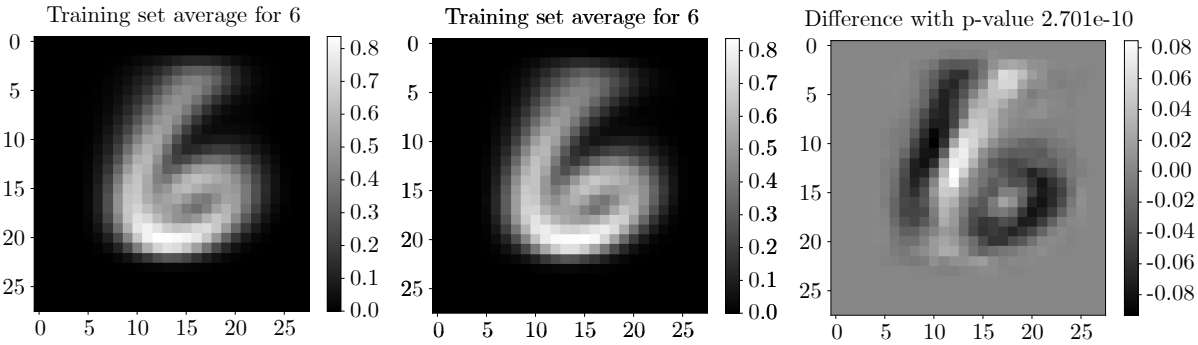


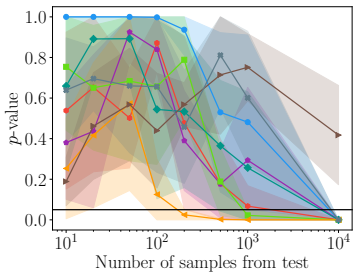
Fig. 6.1 Difference plot for training and test set sixes.

6.8.1 Synthetic medium image shift on MNIST (Figure 6.2)

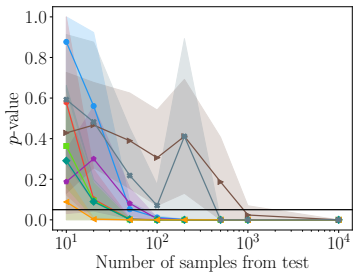
From subfigures (a)-(c), we see that most methods are able to detect the simulated shift with BBSDs being the quickest method for all tested perturbation percentages. We further observe in subfigures (d)-(f) that the true target accuracy increasingly deviates from the model's performance on source data as more samples are perturbed. Since the true target accuracy is usually unknown, we use the accuracy obtained on the top anomalous labeled instances returned by the domain classifier. As we can see, the obtainable accuracy on the samples returned by the difference classifier significantly deviates from the accuracy obtained on source data, which is why we consider this shift harmful to the label classifier's performance.

6.8.2 Rotation angle partitioning on COIL-10 (Figure 6.3)

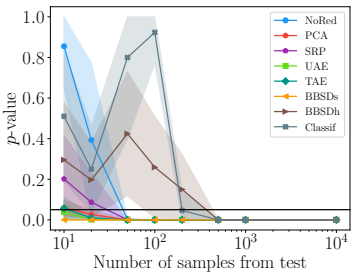
Subfigures (a) and (b) show that our testing framework correctly claims the randomly shuffled dataset containing images from all angles to not contain a shift, while it identifies the partitioned dataset to be noticeably different. However, as we can see from subfigure (e), this shift does not harm the classifier's performance, meaning that the classifier can safely be deployed even when encountering this specific dataset shift.



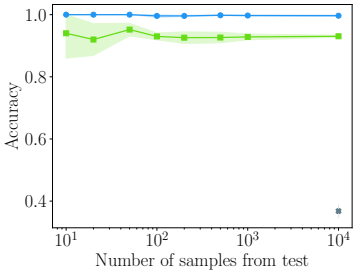
(a) Shift test with 10% perturbed test data.



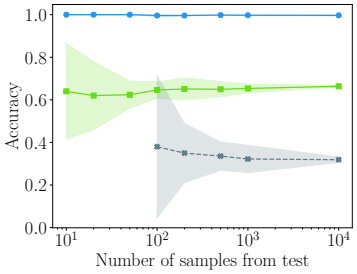
(b) Shift test with 50% perturbed test data.



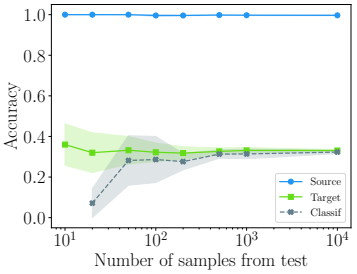
(c) Shift test with 100% perturbed test data.



(d) Classification accuracy on 10% perturbed data.



(e) Classification accuracy on 50% perturbed data.

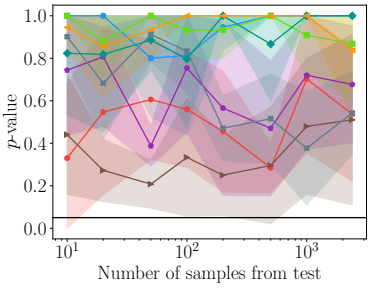


(f) Classification accuracy on 100% perturbed data.

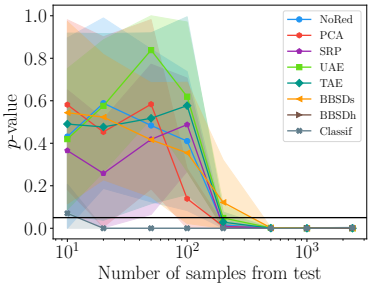
(g) Top different samples.

(h) Top similar samples.

Fig. 6.2 Shift detection results for medium image shift on MNIST. Subfigures (a)-(c) show the p-value evolution of the different DR methods with varying percentages of perturbed data, while subfigures (d)-(f) show the obtainable accuracies over the same perturbations. Subfigures (g) and (h) show the most different and most similar exemplars returned by the domain classifier across perturbation percentages. Plots show mean values obtained over 5 random runs with a 1-σ error-bar.

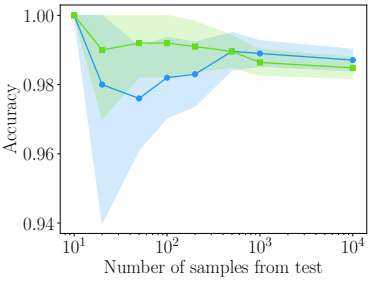


(a) Shift test with randomly shuffled sets containing images from all angles.

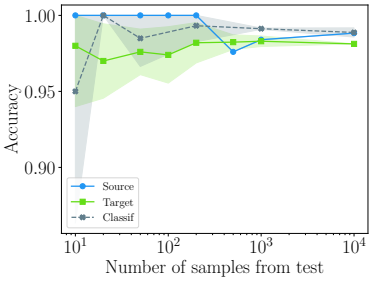


(b) Shift test with angle partitioned source and target sets.

(c) Top different samples.



(d) Classification accuracy on randomly shuffled sets containing images from all angles.



(e) Classification accuracy on angle partitioned source and target sets.

(f) Top similar samples.

Fig. 6.3 Shift detection results on COIL-10 dataset. Subfigure organization is similar to Figure 6.2.

Chapter 7

Conclusion & Future Work

7.1 Summary

In this thesis, we put forth a comprehensive empirical investigation, examining the ways in which dimensionality reduction and two-sample testing might be combined to produce a practical pipeline for detecting distribution shift in real-life machine learning systems. Our results yielded the surprising insights that (i) black-box shift detection with soft predictions works well across a wide variety of scenarios, even when the underlying assumption of invariant class-conditional distributions does not hold; (ii) that given a suitable low-dimensional representation for shift detection, aggregated univariate tests performed separately on each latent dimension outperform multivariate two-sample tests, even when aggregated conservatively; and (iii) that harnessing predictions made by a domain-discriminating classifier enables the characterization of the shift's nature and malignancy.

7.2 Future Work

Our work suggests several open questions that might offer promising paths for future work:

(i) *Can we characterize shifts even better?* While the presented approaches for shift characterization and malignancy detection already provide valuable insights into the nature of the shift, we believe that future works should look into either (1) decreasing the amount of samples needed for a successful qualitative characterization of the shift; or (2) relaxing the requirement of obtaining labels for the top anomalous samples.

(ii) *How do we detect shifts in online data?* Since data often arrives in a continuous stream, adapting our detection scheme to deal with online data would be an important addition. By doing so, we would need to account for and exploit the high degree of correlation between adjacent time steps, known as multiple-hypothesis-testing over time. Recently, Howard et al. (2018) provided some interesting insights on how to design nonparametric, time-evolving confidence intervals, which are correct at every single time-step.

(iii) *How does the proposed detection scheme work in other domains?* As we have mostly explored a standard image classification setting for our experiments, it would be interesting to see how our method performs on problem classes in other machine learning domains, such as natural language processing or graphs.

References

Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-Of-The-Art in Artificial Neural Network Applications: A Survey. *Heliyon*, 4, 2018.

Dimitris Achlioptas. Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins. *Journal of Computer and System Sciences*, 66, 2003.

Alexander A Alemi, Ian Fischer, and Joshua V Dillon. Uncertainty in the Variational Information Bottleneck. *arXiv Preprint arXiv:1807.00906*, 2018.

Dana H Ballard. Modular Learning in Neural Networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 1987.

Oscar Beijbom, Peter J Edmunds, David I Kline, B Greg Mitchell, and David Kriegman. Automated Annotation of Coral Reef Survey Images. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility Theorems for Domain Adaptation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society*, 57, 1995.

Yoav Benjamini, Daniel Yekutieli, et al. The Control of the False Discovery Rate in Multiple Testing Under Dependency. *The Annals of Statistics*, 29, 2001.

Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

J Martin Bland and Douglas G Altman. Multiple Significance Tests: The Bonferroni Method. *BMJ*, 1995.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to End Learning for Self-Driving Cars. *arXiv Preprint arXiv:1604.07316*, 2016.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. LOF: Identifying Density-Based Local Outliers. In *ACM SIGMOD Record*, 2000.

Yee Seng Chan and Hwee Tou Ng. Word Sense Disambiguation with Distribution Estimation. In *International Joint Conference on Artificial intelligence (IJCAI)*, 2005.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 2009.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016.

Hyunsun Choi and Eric Jang. Generative Ensembles for Robust Anomaly Detection. *arXiv Preprint arXiv:1810.01392*, 2018.

Paul Covington, Jay Adams, and Emre Sargin. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016.

David Roxbee Cox. *Principles of Statistical Inference*. Cambridge University Press, 2006.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2011.

Ronald A Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 1936.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer Series in Statistics New York, 2001.

Hans-Otto Georgii. *Stochastik: Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Walter de Gruyter GmbH & Co KG, 2015.

Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2017.

Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity Search in High Dimensions via Hashing. In *Very Large Data Bases (VLDB)*, volume 99, 1999.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2014.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.

Priscilla E Greenwood and Michael S Nikulin. *A Guide to Chi-Squared Testing*, volume 280. John Wiley & Sons, 1996.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research (JMLR)*, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Nicholas A Heard and Patrick Rubin-Delanchy. Choosing Between Methods of Combining-Values. *Biometrika*, 2018.

James J Heckman. Sample Selection Bias as a Specification Error (With an Application to the Estimation of Labor Supply Functions), 1977.

Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-Of-Distribution Examples in Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas G Dietterich. Deep Anomaly Detection with Outlier Exposure. In *International Conference on Learning Representations (ICLR)*, 2019.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 29, 2012.

Yosef Hochberg. A Sharper Bonferroni Procedure for Multiple Tests of Significance.

Yosef Hochberg. Multiple Comparison Procedures. Technical report, John Wiley & Sons, 1987.

Sture Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 1979.

Steven R Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Uniform, Non-parametric, Non-Asymptotic Confidence Sequences. *arXiv Preprint arXiv:1810.08240*, 2018.

David H Hubel and Torsten N Wiesel. Receptive Fields of Single Neurones in the Cat's Striate Cortex. *The Journal of Physiology*, 148, 1959.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*, volume 112. Springer, 2013.

William B Johnson and Joram Lindenstrauss. Extensions of Lipschitz Mappings into a Hilbert Space. *Contemporary Mathematics*, 26, 1984.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv Preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv Preprint arXiv:1312.6114*, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

Paras Lakhani and Baskaran Sundaram. Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks. *Radiology*, 284, 2017.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86, 1998.

Daniel D Lee and H Sebastian Seung. Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training Confidence-Calibrated Classifiers for Detecting Out-Of-Distribution Samples. In *International Conference on Learning Representations (ICLR)*, 2018.

Erich L Lehmann and Joseph P Romano. *Testing Statistical Hypotheses*. Springer Science & Business Media, 2006.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

Ping Li, Trevor J Hastie, and Kenneth W Church. Very Sparse Random Projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2006.

Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the Reliability of Out-Of-Distribution Image Detection in Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and Correcting for Label Shift with Black Box Predictors. In *International Conference on Machine Learning (ICML)*, 2018.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *International Conference on Data Mining (ICDM)*, 2008.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer Feature Learning with Joint Distribution Adaptation. In *International Conference on Computer Vision (ICCV)*, 2013.

Thomas M Loughin. A Systematic Comparison of Methods for Combining p-Values from Independent Tests. *Computational Statistics & Data Analysis*, 2004.

Alireza Makhzani and Brendan Frey. K-Sparse Autoencoders. *arXiv Preprint arXiv:1312.5663*, 2013.

Charles F Manski and Steven R Lerman. The Estimation of Choice Probabilities from Choice Based Samples. *Econometrica: Journal of the Econometric Society*, 1977.

Markos Markou and Sameer Singh. Novelty Detection: A Review: Part 1: Statistical Approaches. *Signal Processing*, 2003.

Frank J Massey Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American statistical Association*, 46, 1951.

James Mercer. Functions of Positive and Negative Type, and Their Connection the Theory of Integral Equations. *Philosophical Transactions*, 209, 1909.

Tom Mitchell. *Machine Learning*. 1997.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018.

Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia Object Image Library (COIL-100). 1996.

Karl Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2, 1901.

Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry A Wasserman. On the Decreasing Power of Kernel and Distance Based Nonparametric Hypothesis Tests in High Dimensions. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Aaditya Ramdas, Aarti Singh, and Larry Wasserman. Classification Accuracy as a Proxy for Two Sample Testing. *arXiv Preprint arXiv:1602.02210*, 2016.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 1951.

Paul R Rosenbaum and Donald B Rubin. The Central Role of the Propensity Score in Observational Studies for Causal Effects. *Biometrika*, 1983.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning Representations by Back-Propagating Errors. *Cognitive Modeling*, 5, 1988.

G Rupert Jr et al. *Simultaneous Statistical Inference*. Springer Science & Business Media, 2012.

Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery. In *International Conference on Information Processing in Medical Imaging*, 2017.

Bernhard Schölkopf and Alexander J Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support Vector Method for Novelty Detection. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On Causal and Anticausal Learning. In *International Conference on Machine Learning (ICML)*, 2012.

D Sculley, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. Machine Learning: The High-Interest Credit Card of Technical Debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

Alireza Shafaei, Mark Schmidt, and James J Little. Does Your Model Know the Digit 6 Is Not a Cat? A Less Biased Evaluation of Outlier Detectors. *arXiv Preprint arXiv:1809.04729*, 2018.

Gabi Shalev, Yossi Adi, and Joseph Keshet. Out-Of-Distribution Detection Using Multiple Semantic Label Representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Hidetoshi Shimodaira. Improving Predictive Inference Under Covariate Shift by Weighting the Log-Likelihood Function. *Journal of Statistical Planning and Inference*, 2000.

Zbyněk Šidák. Rectangular Confidence Regions for the Means of Multivariate Normal Distributions. *Journal of the American Statistical Association*, 62, 1967.

R John Simes. An Improved Bonferroni Procedure for Multiple Tests of Significance. *Biometrika*, 1986.

Zak Stone, Todd Zickler, and Trevor Darrell. Autotagging Facebook: Social Network Context Improves Photo Annotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 2008.

Amos Storkey. When Training and Test Sets Are Different: Characterizing Learning Transfer. *Dataset Shift in Machine Learning*, 2009.

Masashi Sugiyama, Neil D Lawrence, Anton Schwaighofer, et al. *Dataset Shift in Machine Learning*. The MIT Press, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. *arXiv Preprint arXiv:1312.6199*, 2013.

Charles Truong, Laurent Oudre, and Nicolas Vayatis. A Review of Change Point Detection Methods. *arXiv Preprint arXiv:1801.00718*, 2018.

John W Tukey et al. Comparing Individual Means in the Analysis of Variance. *Biometrics*, 5, 1949.

Vladimir Vovk and Ruodu Wang. Combining p-Values via Averaging. *arXiv Preprint arXiv:1212.4966*, 2018.

Dmitri V Zaykin, Lev A Zhivotovsky, Peter H Westfall, and Bruce S Weir. Truncated Product Method for Combining p-Values. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society*, 2002.

Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv Preprint arXiv:1212.5701*, 2012.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain Adaptation Under Target and Conditional Shift. In *International Conference on Machine Learning (ICML)*, 2013.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial Attacks on Neural Networks for Graph Data. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.

Appendix A

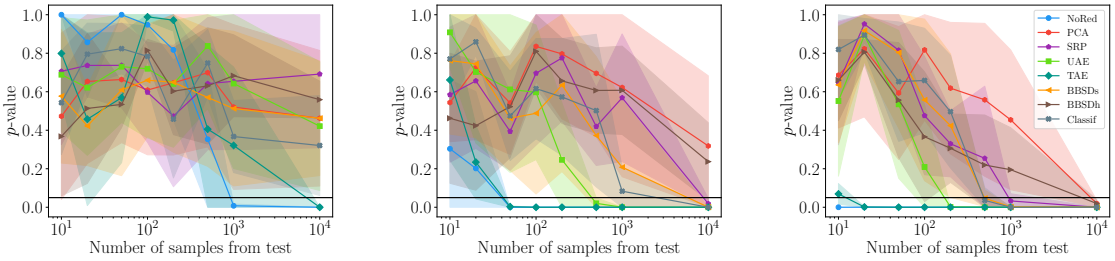
Detailed Shift Detection Results

Our complete shift detection results in which we evaluate different kinds of shifts on MNIST and CIFAR-10 using the proposed methods are documented below. In addition to our artificially generated shifts, we also evaluated our testing procedure on the original splits provided by MNIST, Fashion MNIST, CIFAR-10, and SVHN, as well as on the outlined domain adaptation datasets.

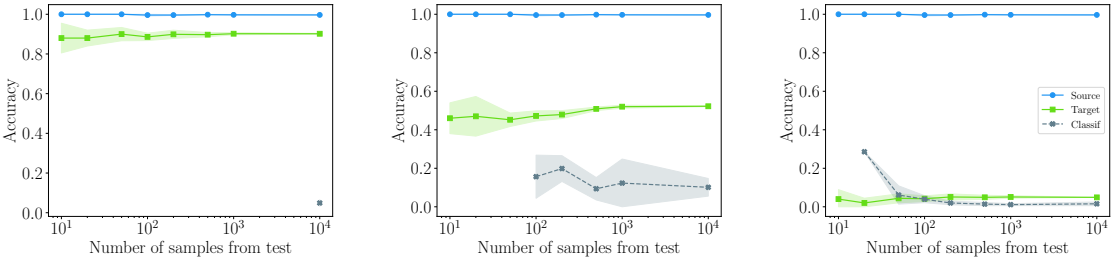
A.1 Artificially Generated Shifts

A.1.1 MNIST

Adversarial Shift



(a) 10% adversarial samples. (b) 50% adversarial samples. (c) 100% adversarial samples.

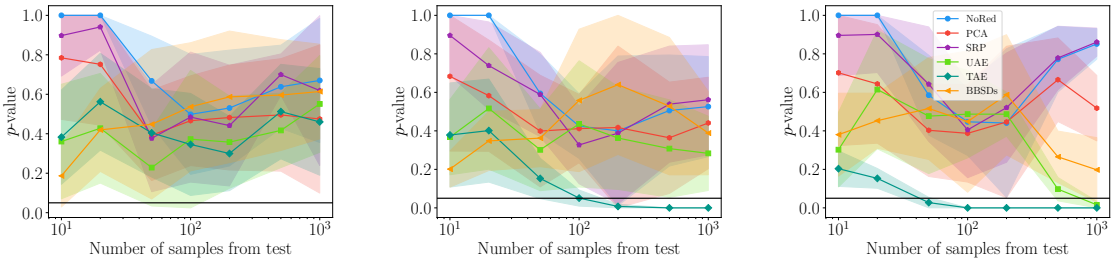


(d) 10% adversarial samples. (e) 50% adversarial samples. (f) 100% adversarial samples.

(g) Top different samples.

(h) Top similar samples.

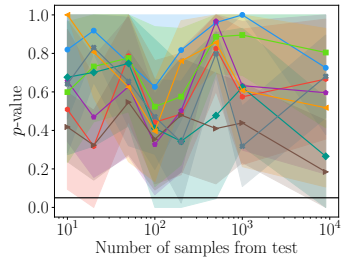
Fig. A.1 MNIST adversarial shift, univariate two-sample tests + Bonferroni aggregation.



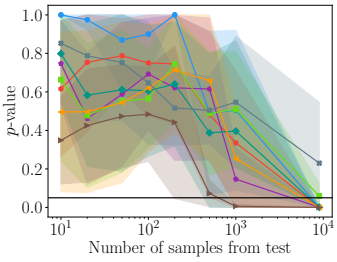
(a) 10% adversarial samples. (b) 50% adversarial samples. (c) 100% adversarial samples.

Fig. A.2 MNIST adversarial shift, multivariate two-sample tests.

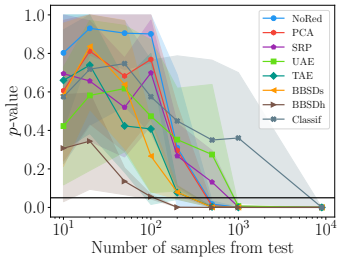
Knock-Out Shift



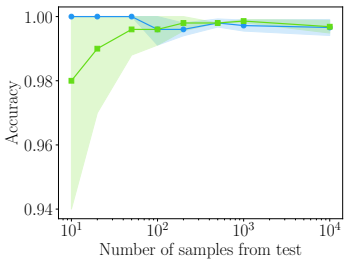
(a) Knock out 10% of class 0.



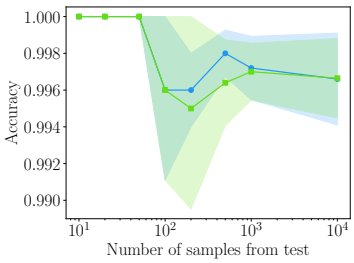
(b) Knock out 50% of class 0.



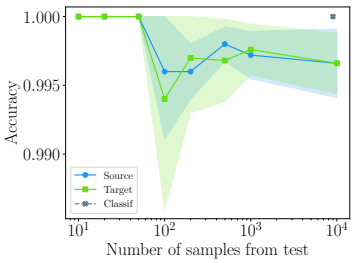
(c) Knock out 100% of class 0.



(d) Knock out 10% of class 0.



(e) Knock out 50% of class 0.

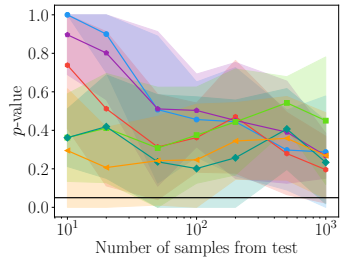


(f) Knock out 100% of class 0.

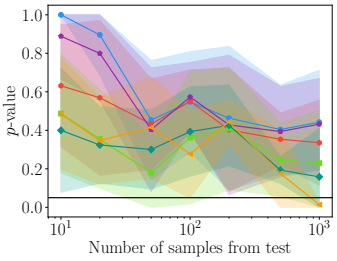
(g) Top different samples.

(h) Top similar samples.

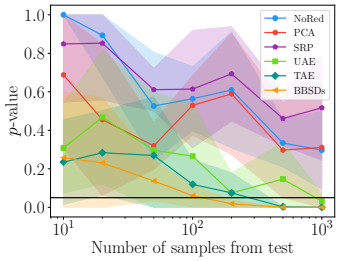
Fig. A.3 MNIST knock-out shift, univariate two-sample tests + Bonferroni aggregation.



(a) Knock out 10% of class 0.



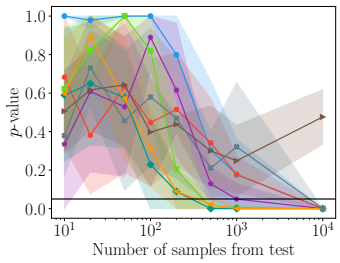
(b) Knock out 50% of class 0.



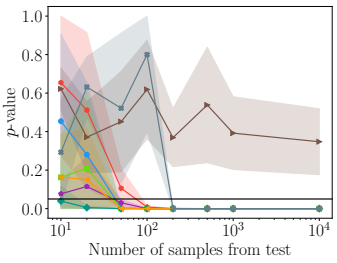
(c) Knock out 100% of class 0.

Fig. A.4 MNIST knock-out shift, multivariate two-sample tests.

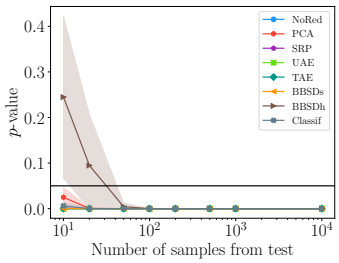
Large Gaussian Noise Shift



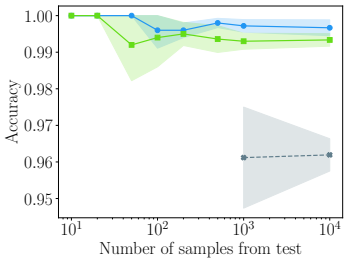
(a) 10% perturbed samples.



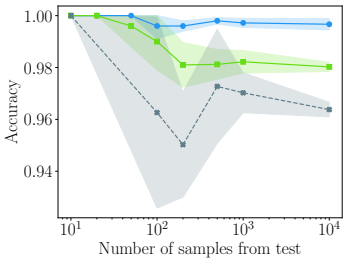
(b) 50% perturbed samples.



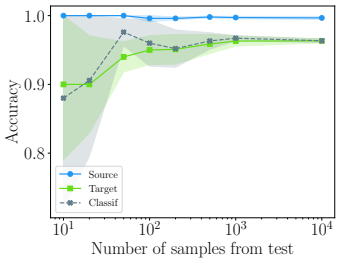
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

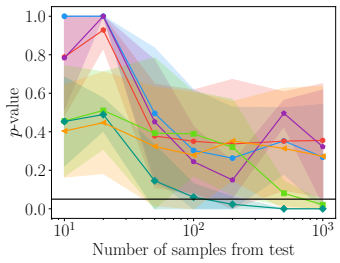


(f) 100% perturbed samples.

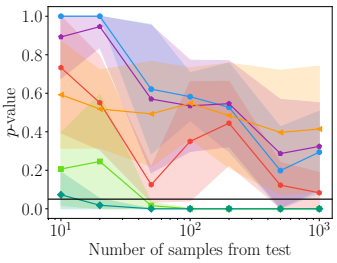
(g) Top different samples.

(h) Top similar samples.

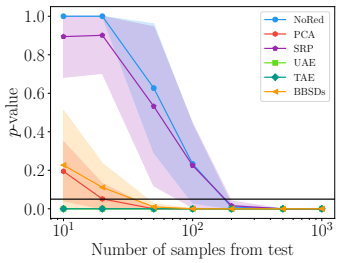
Fig. A.5 MNIST large Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



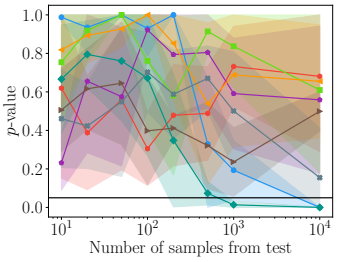
(b) 50% perturbed samples.



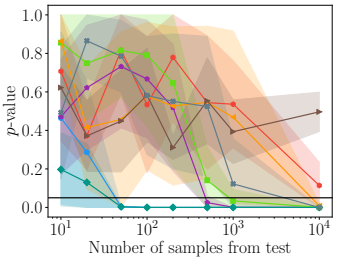
(c) 100% perturbed samples.

Fig. A.6 MNIST large Gaussian noise shift, multivariate two-sample tests.

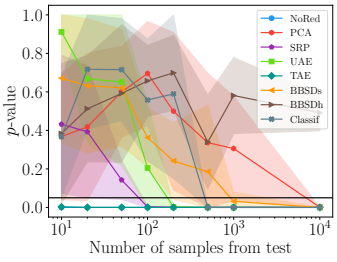
Medium Gaussian Noise Shift



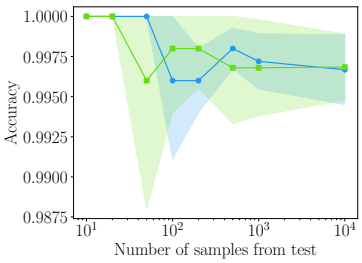
(a) 10% perturbed samples.



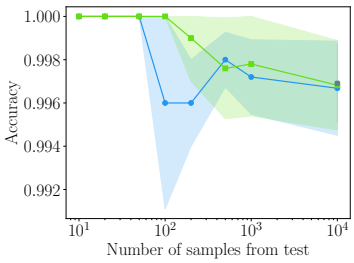
(b) 50% perturbed samples.



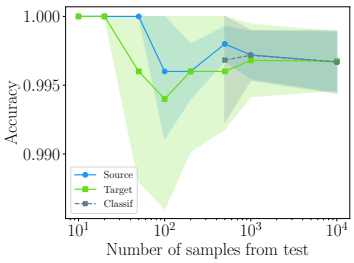
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

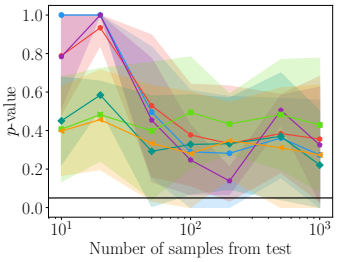


(f) 100% perturbed samples.

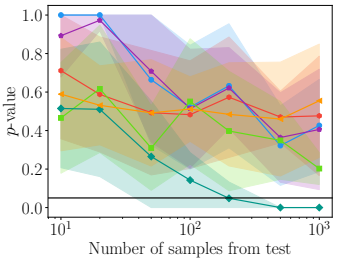
(g) Top different samples.

(h) Top similar samples.

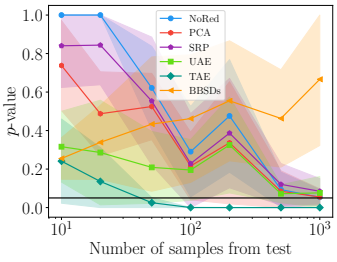
Fig. A.7 MNIST medium Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



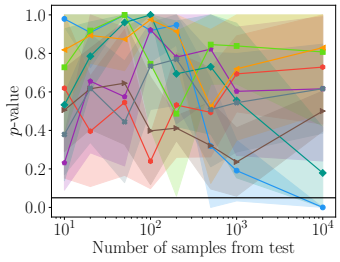
(b) 50% perturbed samples.



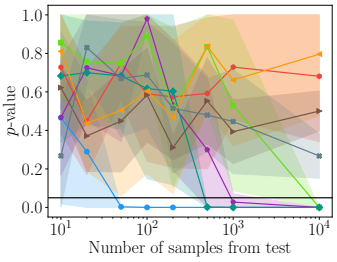
(c) 100% perturbed samples.

Fig. A.8 MNIST medium Gaussian noise shift, multivariate two-sample tests.

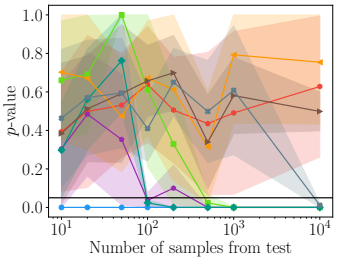
Small Gaussian Noise Shift



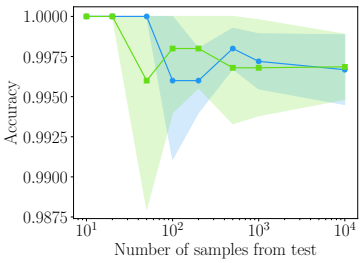
(a) 10% perturbed samples.



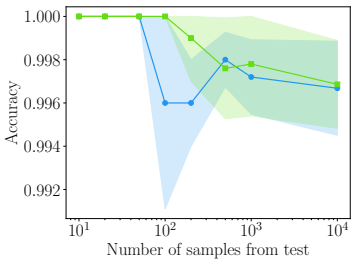
(b) 50% perturbed samples.



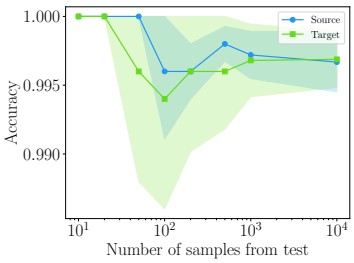
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

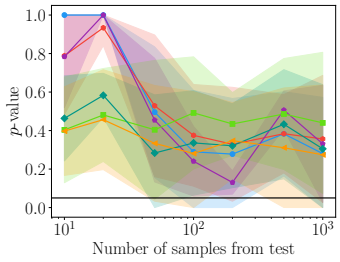


(f) 100% perturbed samples.

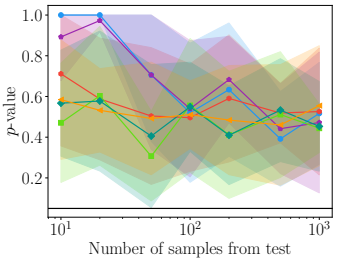
(g) Top different samples.

(h) Top similar samples.

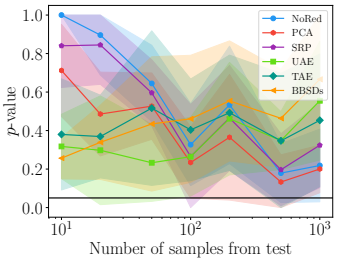
Fig. A.9 MNIST small Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



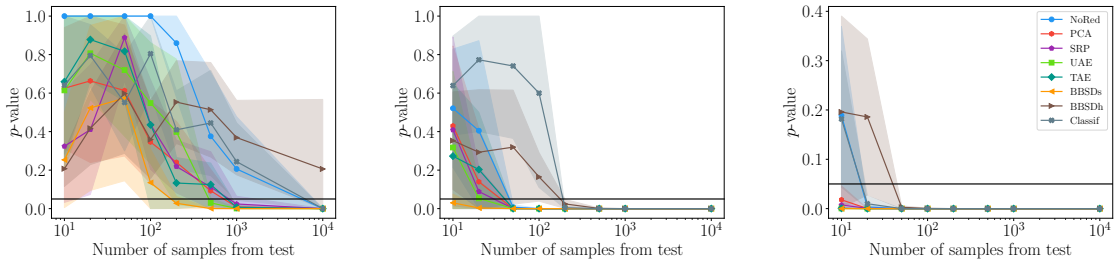
(b) 50% perturbed samples.



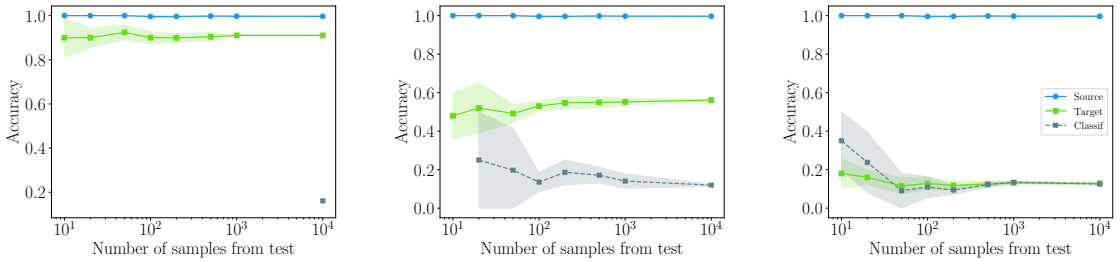
(c) 100% perturbed samples.

Fig. A.10 MNIST small Gaussian noise shift, multivariate two-sample tests.

Large Image Shift



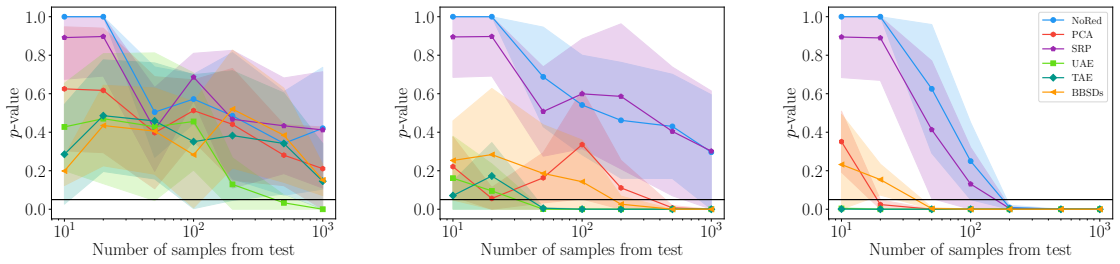
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.



(d) 10% perturbed samples. (e) 50% perturbed samples. (f) 100% perturbed samples.

(g) Top different samples. (h) Top similar samples.

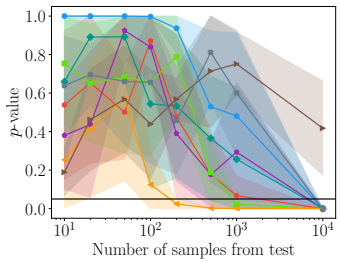
Fig. A.11 MNIST large image shift, univariate two-sample tests + Bonferroni aggregation.



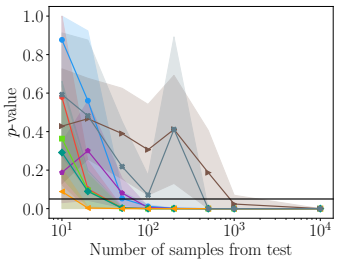
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.

Fig. A.12 MNIST large image shift, multivariate two-sample tests.

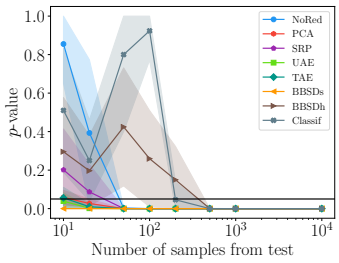
Medium Image Shift



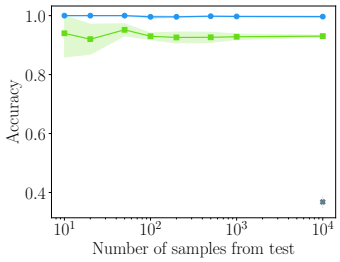
(a) 10% perturbed samples.



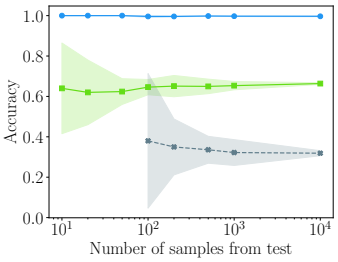
(b) 50% perturbed samples.



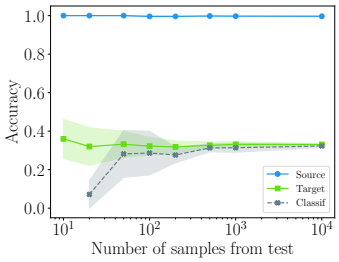
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

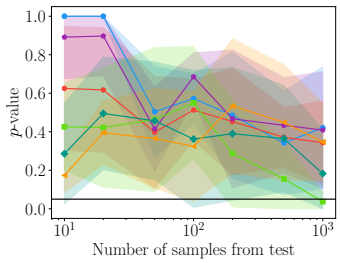


(f) 100% perturbed samples.

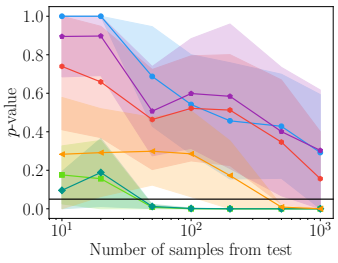
(g) Top different samples.

(h) Top similar samples.

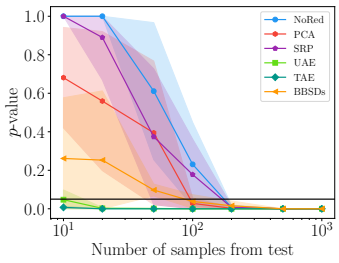
Fig. A.13 MNIST medium image shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



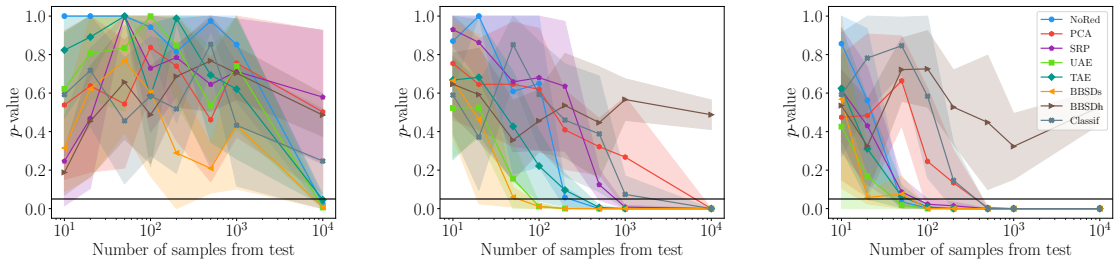
(b) 50% perturbed samples.



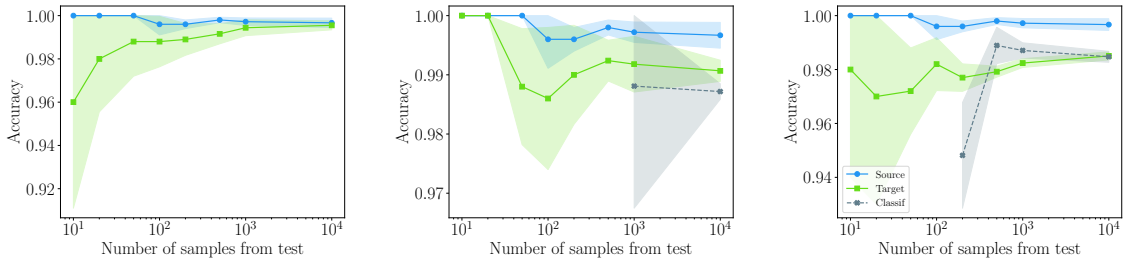
(c) 100% perturbed samples.

Fig. A.14 MNIST medium image shift, multivariate two-sample tests.

Small Image Shift



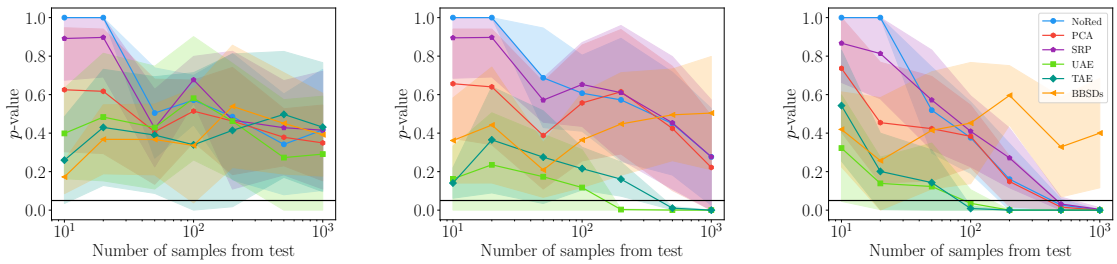
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.



(d) 10% perturbed samples. (e) 50% perturbed samples. (f) 100% perturbed samples.

(g) Top different samples. (h) Top similar samples.

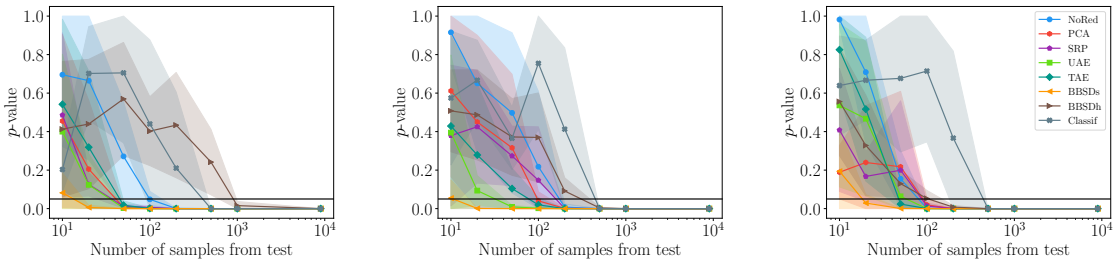
Fig. A.15 MNIST small image shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.

Fig. A.16 MNIST small image shift, multivariate two-sample tests.

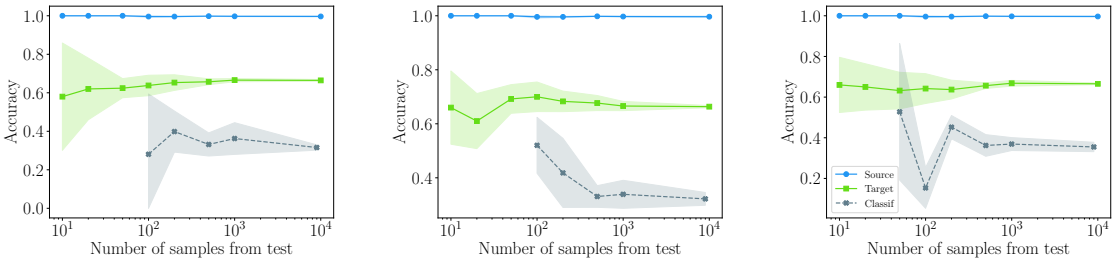
Medium Image Shift + Knock-Out Shift



(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

(c) Knock out 100% of class 0.



(d) Knock out 10% of class 0.

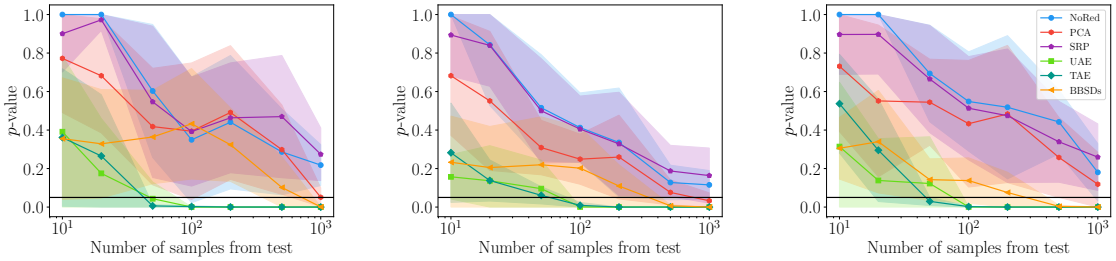
(e) Knock out 50% of class 0.

(f) Knock out 100% of class 0.

(g) Top different samples.

(h) Top similar samples.

Fig. A.17 MNIST medium image shift (50%, fixed) plus knock-out shift (variable), univariate two-sample tests + Bonferroni aggregation.



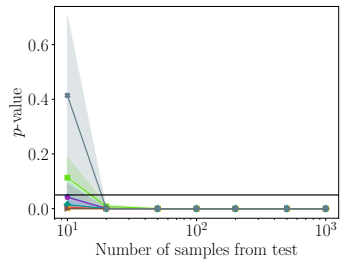
(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

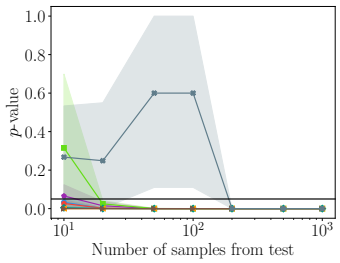
(c) Knock out 100% of class 0.

Fig. A.18 MNIST medium image shift (50%, fixed) plus knock-out shift (variable), multivariate two-sample tests.

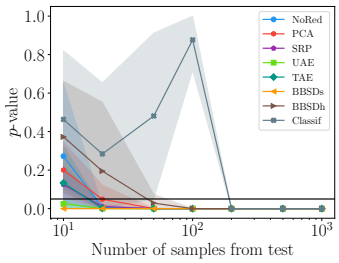
Only-Zero Shift + Medium Image Shift



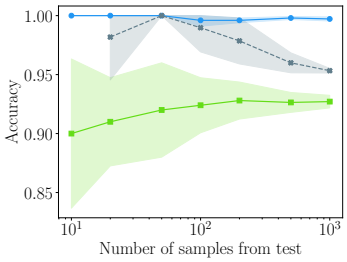
(a) 10% perturbed samples.



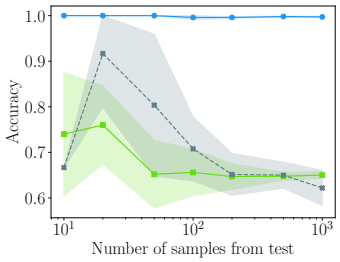
(b) 50% perturbed samples.



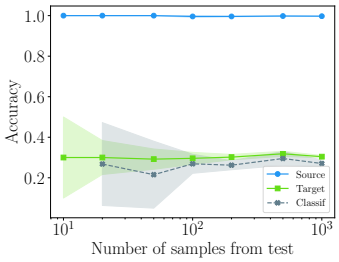
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

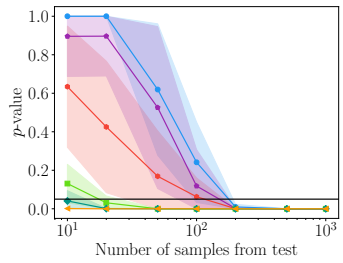


(f) 100% perturbed samples.

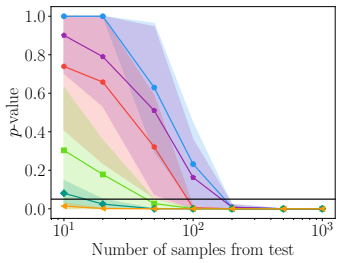
(g) Top different samples.

(h) Top similar samples.

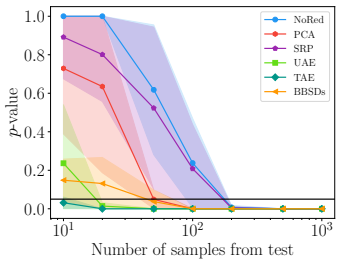
Fig. A.19 MNIST only-zero shift (fixed) plus medium image shift (variable), univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



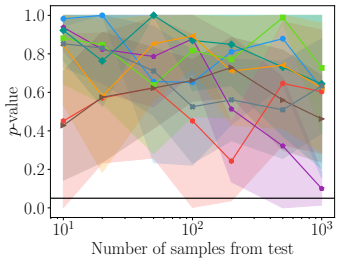
(b) 50% perturbed samples.



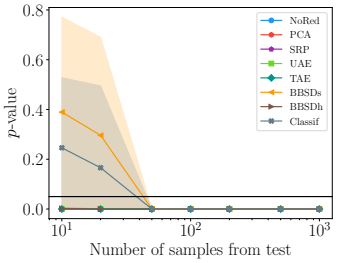
(c) 100% perturbed samples.

Fig. A.20 MNIST only-zero shift (fixed) plus medium image shift (variable), multivariate two-sample tests.

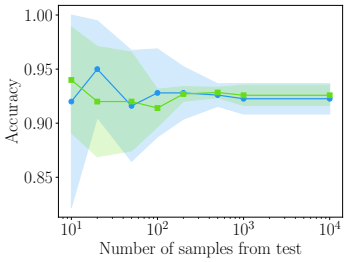
A.1.2 Domain Adaptation MNIST to USPS



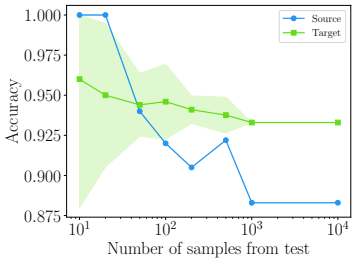
(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.



(c) Randomly shuffled dataset with same split proportions as original dataset.

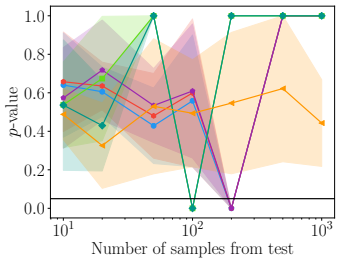


(d) Original split.

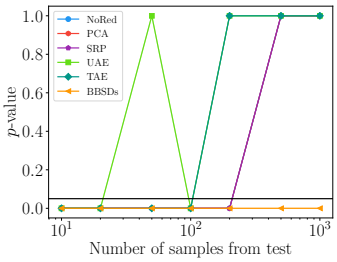
(e) Top different samples.

(f) Top similar samples.

Fig. A.21 MNIST to USPS domain adaptation, univariate two-sample tests + Bonferroni aggregation.



(a) Randomly shuffled dataset with same split proportions as original dataset.

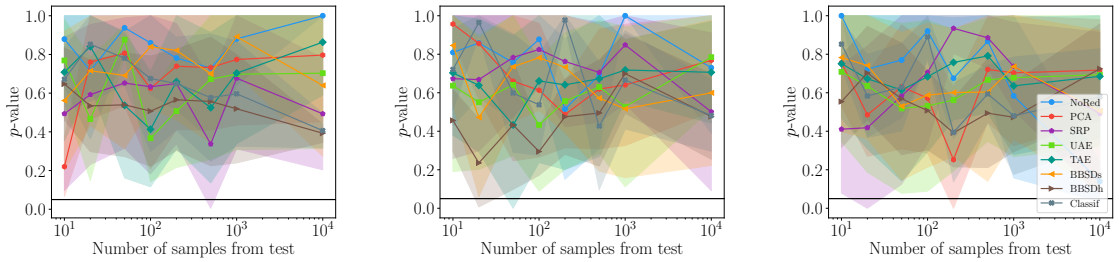


(b) Original split.

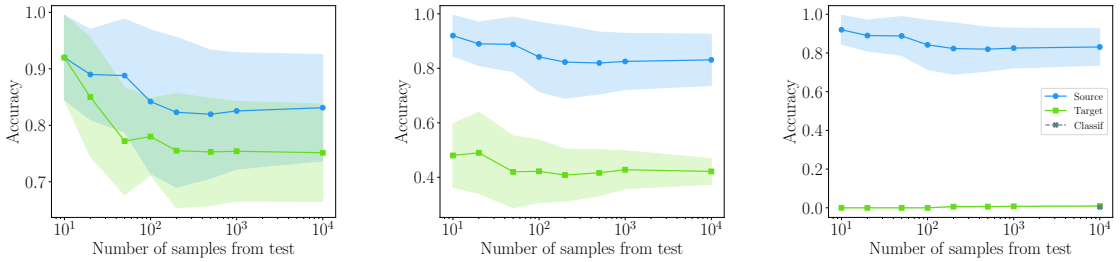
Fig. A.22 MNIST to USPS domain adaptation, multivariate two-sample tests.

A.1.3 CIFAR-10

Adversarial Shift



(a) 10% adversarial samples. (b) 50% adversarial samples. (c) 100% adversarial samples.



(d) 10% adversarial samples. (e) 50% adversarial samples. (f) 100% adversarial samples.

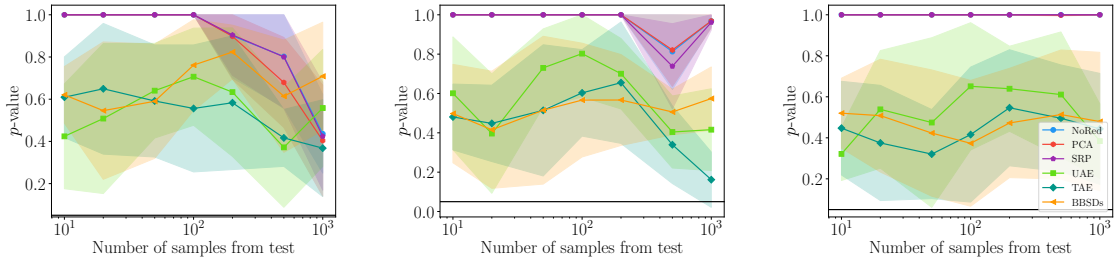
No samples available as Classif did not detect a shift.

(g) Top different samples.

No samples available as Classif did not detect a shift.

(h) Top similar samples.

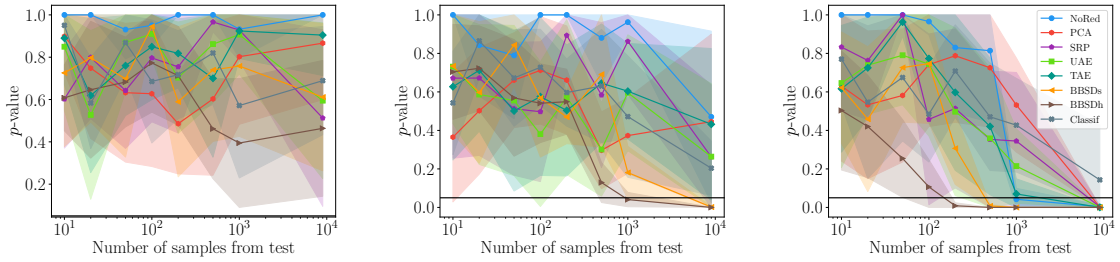
Fig. A.23 CIFAR-10 adversarial shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% adversarial samples. (b) 50% adversarial samples. (c) 100% adversarial samples.

Fig. A.24 CIFAR-10 adversarial shift, multivariate two-sample tests.

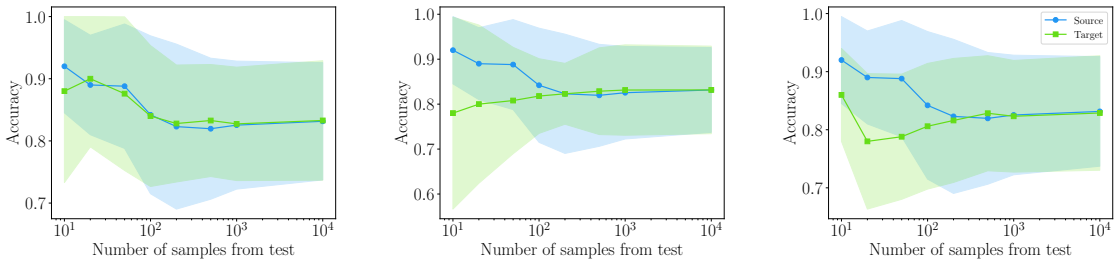
Knock-Out Shift



(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

(c) Knock out 100% of class 0.



(d) Knock out 10% of class 0.

(e) Knock out 50% of class 0.

(f) Knock out 100% of class 0.

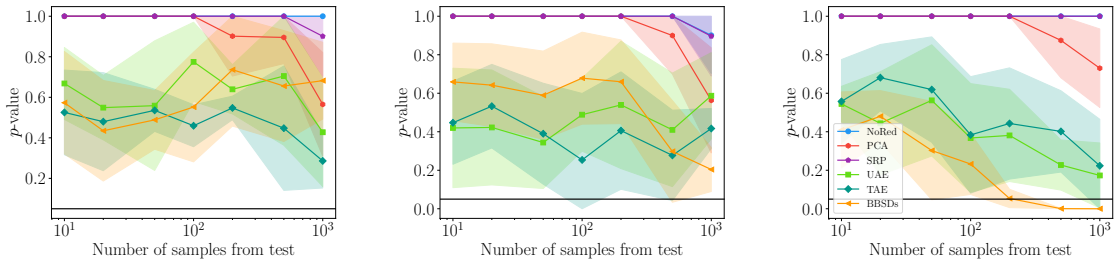
No samples available as Classif did not detect a shift.

(g) Top different samples.

No samples available as Classif did not detect a shift.

(h) Top similar samples.

Fig. A.25 CIFAR-10 knock-out shift, univariate two-sample tests + Bonferroni aggregation.



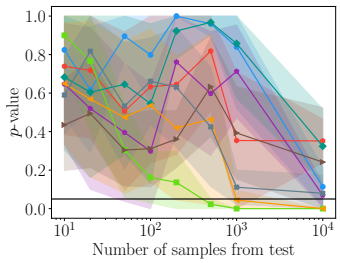
(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

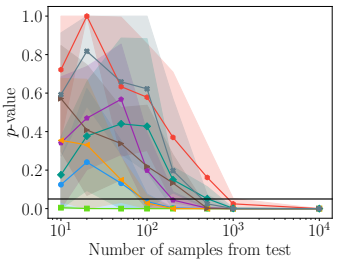
(c) Knock out 100% of class 0.

Fig. A.26 CIFAR-10 knock-out shift, multivariate two-sample tests.

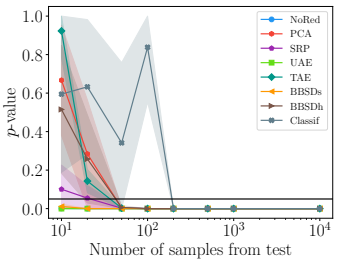
Large Gaussian Noise Shift



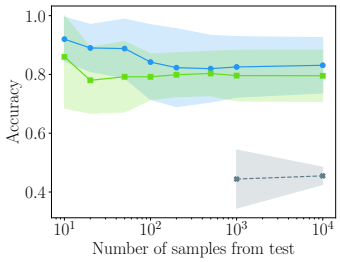
(a) 10% perturbed samples.



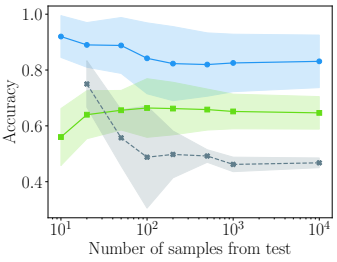
(b) 50% perturbed samples.



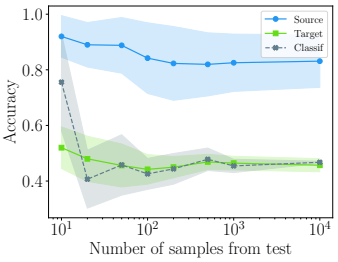
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

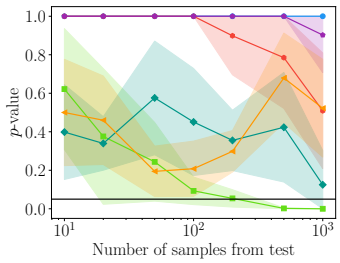


(f) 100% perturbed samples.

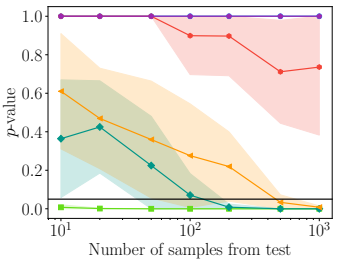
(g) Top different samples.

(h) Top similar samples.

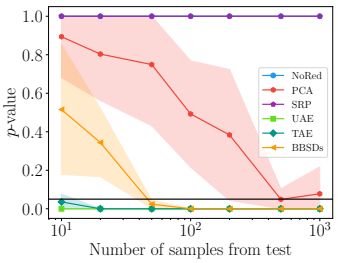
Fig. A.27 CIFAR-10 large Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



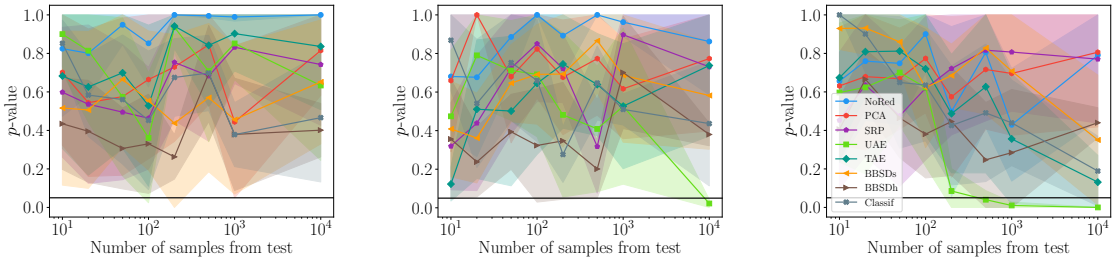
(b) 50% perturbed samples.



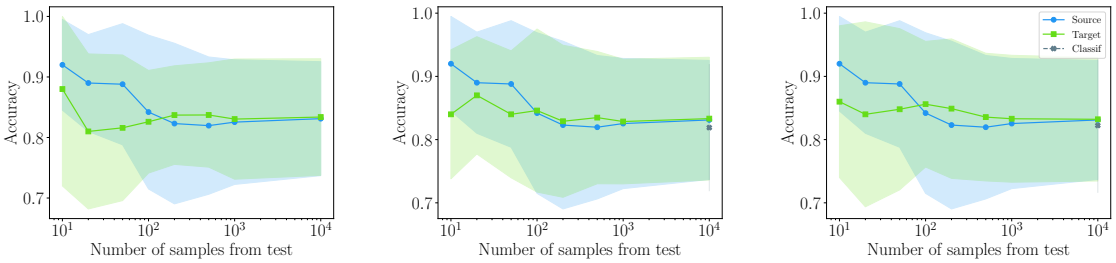
(c) 100% perturbed samples.

Fig. A.28 CIFAR-10 large Gaussian noise shift, multivariate two-sample tests.

Medium Gaussian Noise Shift



(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.



(d) 10% perturbed samples. (e) 50% perturbed samples. (f) 100% perturbed samples.

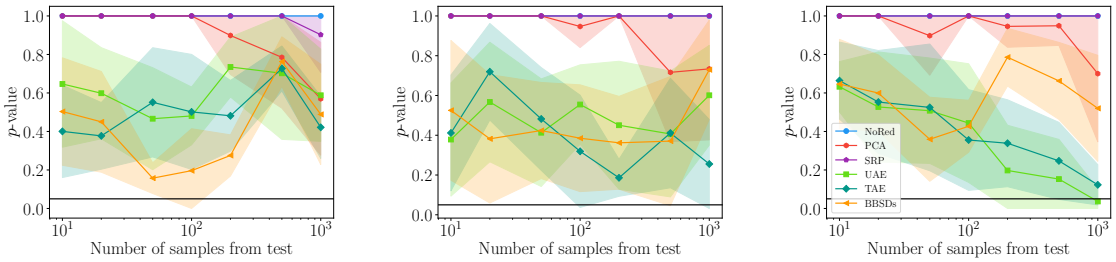
No samples available as Classif did not detect a shift.

(g) Top different samples.

No samples available as Classif did not detect a shift.

(h) Top similar samples.

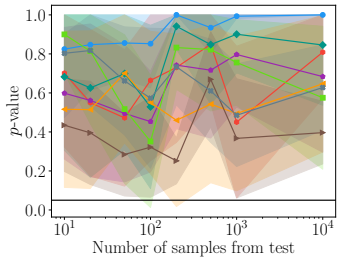
Fig. A.29 CIFAR-10 medium Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



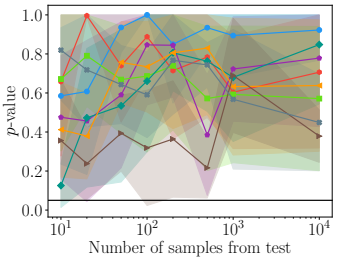
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.

Fig. A.30 CIFAR-10 medium Gaussian noise shift, multivariate two-sample tests.

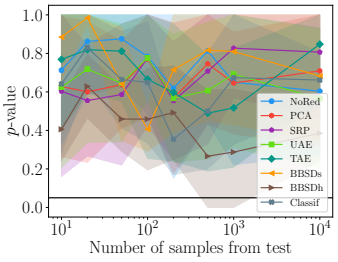
Small Gaussian Noise Shift



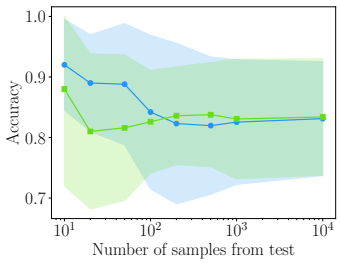
(a) 10% perturbed samples.



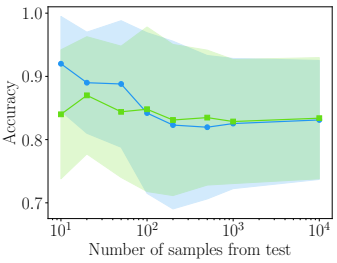
(b) 50% perturbed samples.



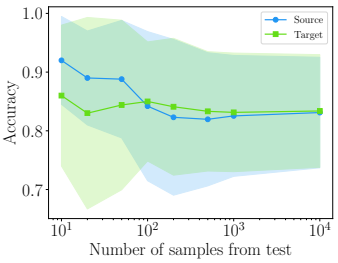
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.



(f) 100% perturbed samples.

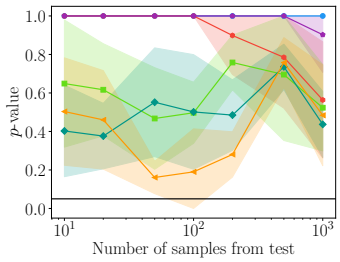
No samples available as Classif did not detect a shift.

(g) Top different samples.

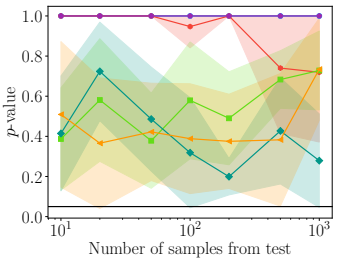
No samples available as Classif did not detect a shift.

(h) Top similar samples.

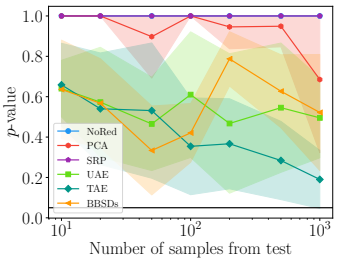
Fig. A.31 CIFAR-10 small Gaussian noise shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



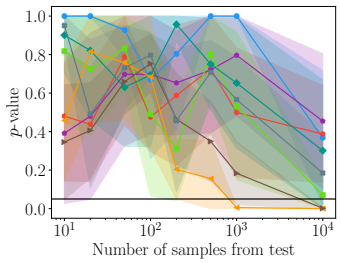
(b) 50% perturbed samples.



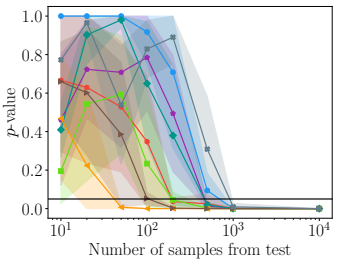
(c) 100% perturbed samples.

Fig. A.32 CIFAR-10 small Gaussian noise shift, multivariate two-sample tests.

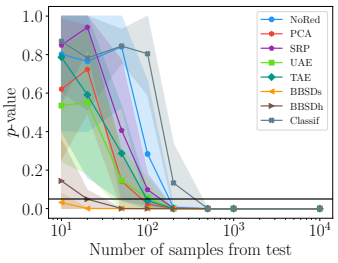
Large Image Shift



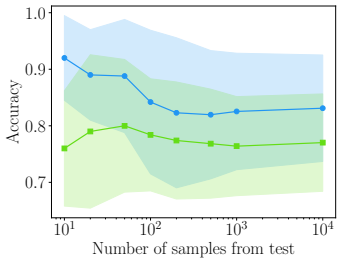
(a) 10% perturbed samples.



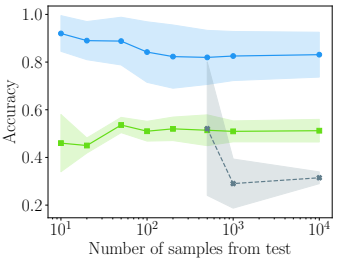
(b) 50% perturbed samples.



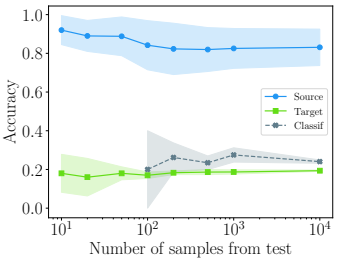
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

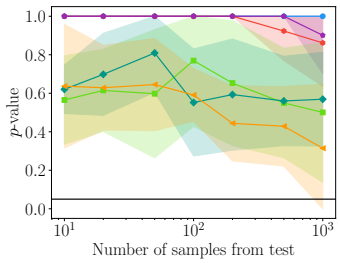


(f) 100% perturbed samples.

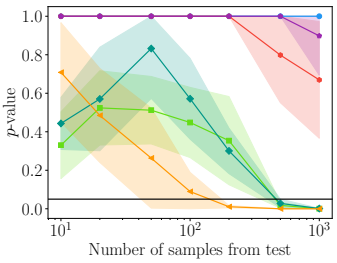
(g) Top different samples.

(h) Top similar samples.

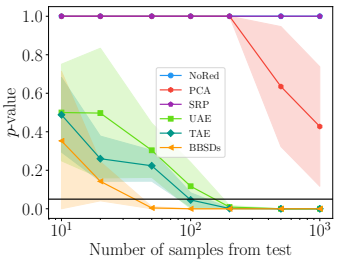
Fig. A.33 CIFAR-10 large image shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



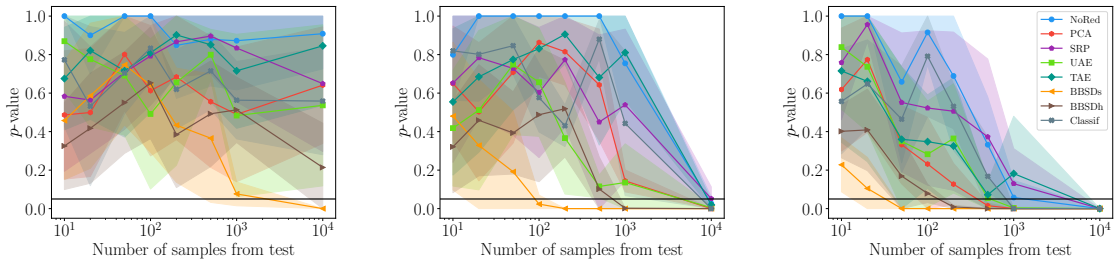
(b) 50% perturbed samples.



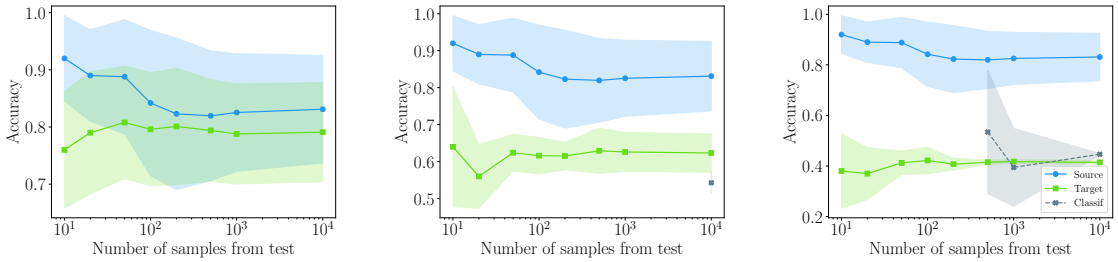
(c) 100% perturbed samples.

Fig. A.34 CIFAR-10 large image shift, multivariate two-sample tests.

Medium Image Shift



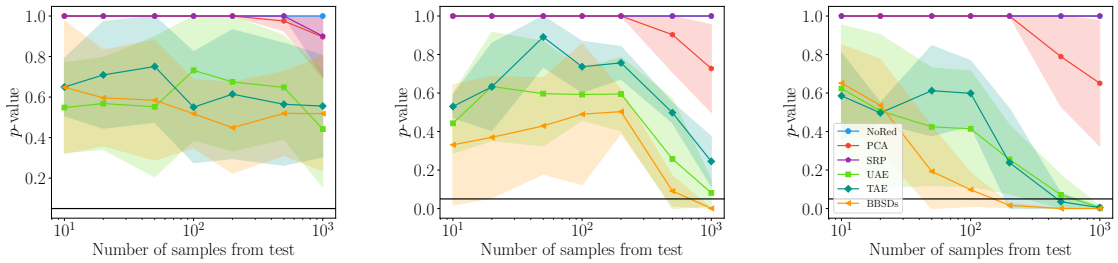
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.



(d) 10% perturbed samples. (e) 50% perturbed samples. (f) 100% perturbed samples.

(g) Top different samples. (h) Top similar samples.

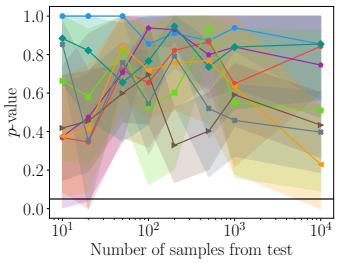
Fig. A.35 CIFAR-10 medium image shift, univariate two-sample tests + Bonferroni aggregation.



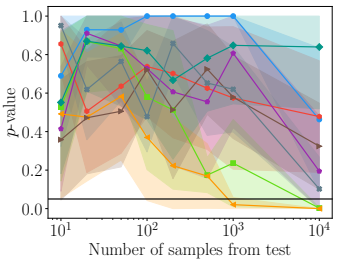
(a) 10% perturbed samples. (b) 50% perturbed samples. (c) 100% perturbed samples.

Fig. A.36 CIFAR-10 medium image shift, multivariate two-sample tests.

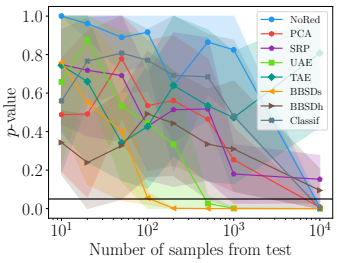
Small Image Shift



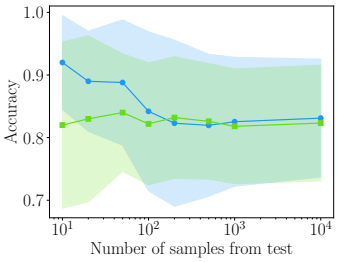
(a) 10% perturbed samples.



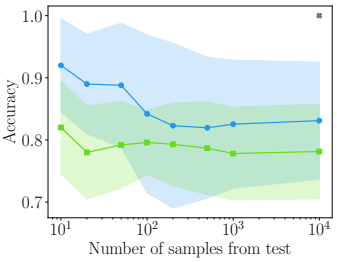
(b) 50% perturbed samples.



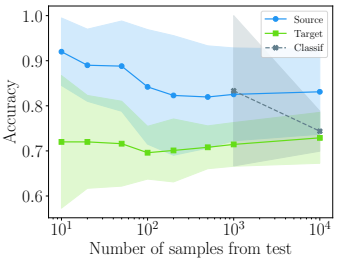
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

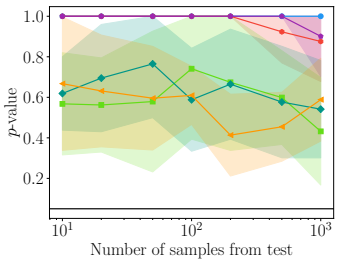


(f) 100% perturbed samples.

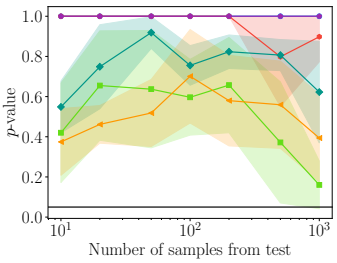
(g) Top different samples.

(h) Top similar samples.

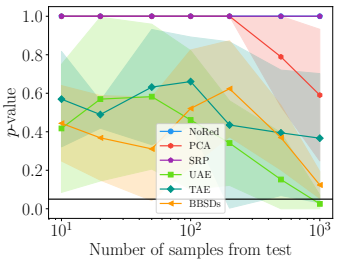
Fig. A.37 CIFAR-10 small image shift, univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



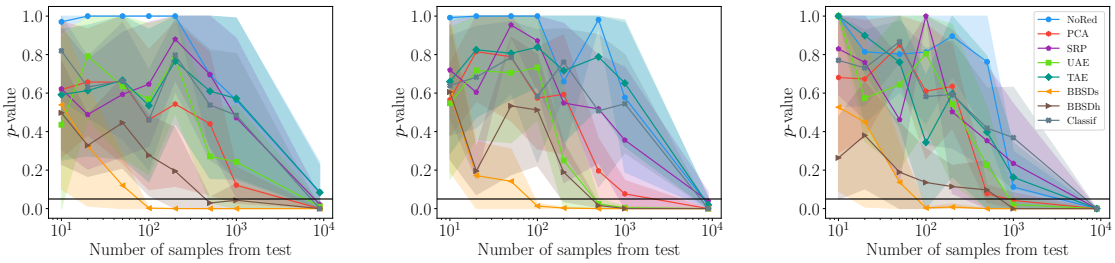
(b) 50% perturbed samples.



(c) 100% perturbed samples.

Fig. A.38 CIFAR-10 small image shift, multivariate two-sample tests.

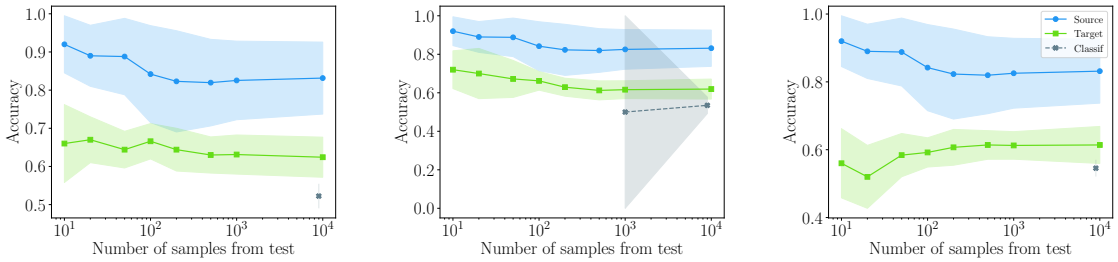
Medium Image Shift + Knock-Out Shift



(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

(c) Knock out 100% of class 0.



(d) Knock out 10% of class 0.

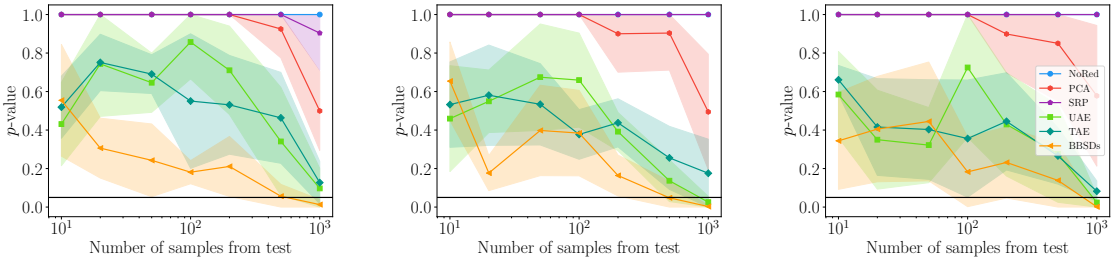
(e) Knock out 50% of class 0.

(f) Knock out 100% of class 0.

(g) Top different samples.

(h) Top similar samples.

Fig. A.39 CIFAR-10 medium image shift (50%, fixed) plus knock-out shift (variable), univariate two-sample tests + Bonferroni aggregation.



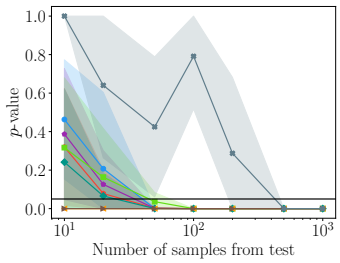
(a) Knock out 10% of class 0.

(b) Knock out 50% of class 0.

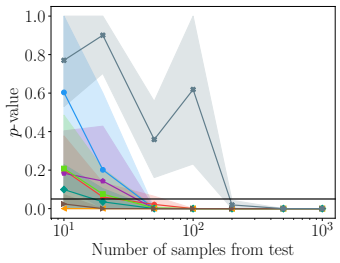
(c) Knock out 100% of class 0.

Fig. A.40 CIFAR-10 medium image shift (50%, fixed) plus knock-out shift (variable), multivariate two-sample tests.

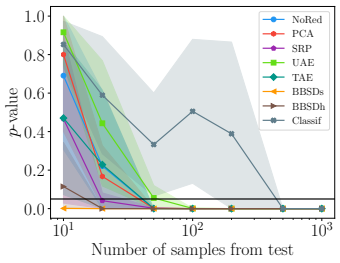
Only Zero Shift + Medium Image Shift



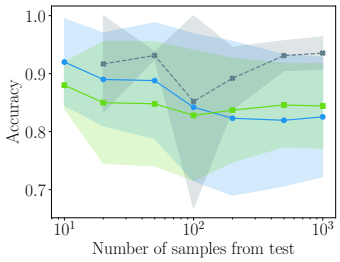
(a) 10% perturbed samples.



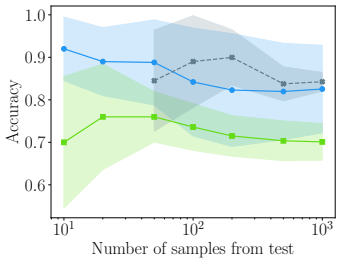
(b) 50% perturbed samples.



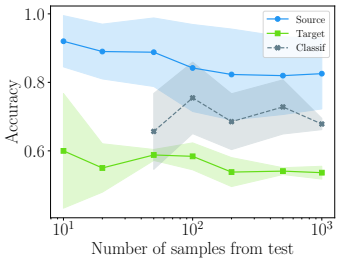
(c) 100% perturbed samples.



(d) 10% perturbed samples.



(e) 50% perturbed samples.

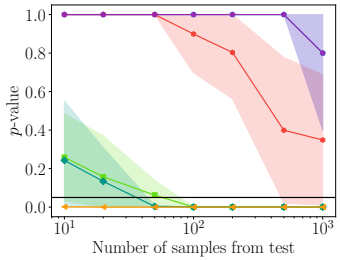


(f) 100% perturbed samples.

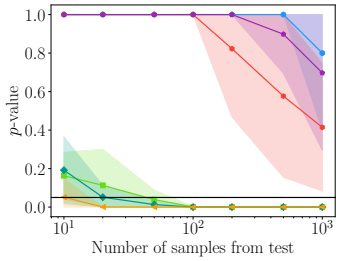
(g) Top different samples.

(h) Top similar samples.

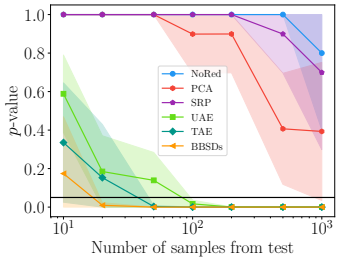
Fig. A.41 CIFAR-10 only-zero shift (fixed) plus medium image shift (variable), univariate two-sample tests + Bonferroni aggregation.



(a) 10% perturbed samples.



(b) 50% perturbed samples.

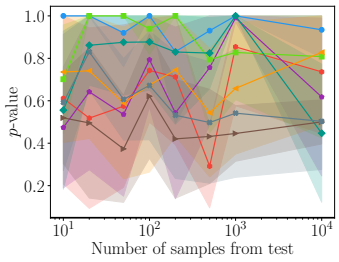


(c) 100% perturbed samples.

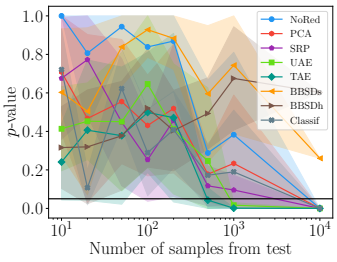
Fig. A.42 CIFAR-10 only-zero shift (fixed) plus medium image shift (variable), multivariate two-sample tests.

A.2 Original Splits

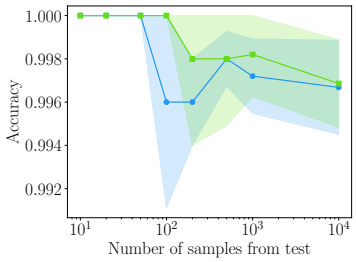
A.2.1 MNIST



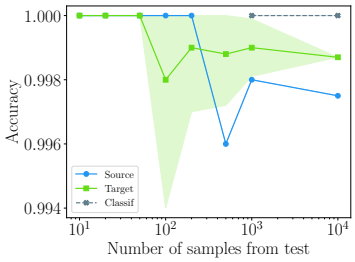
(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.



(c) Randomly shuffled dataset with same split proportions as original dataset.

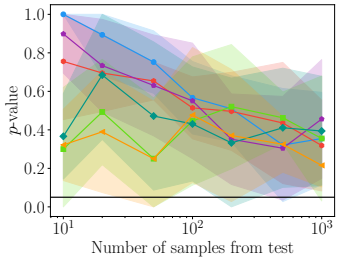


(d) Original split.

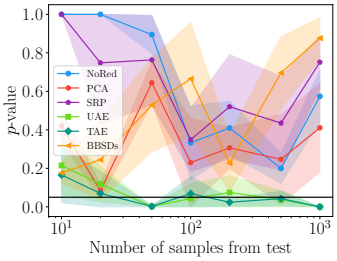
(e) Top different samples.

(f) Top similar samples.

Fig. A.43 MNIST randomized and original split, univariate two-sample tests + Bonferroni aggregation.



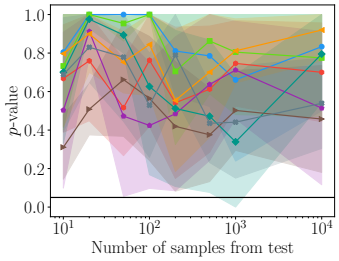
(a) Randomly shuffled dataset with same split proportions as original dataset.



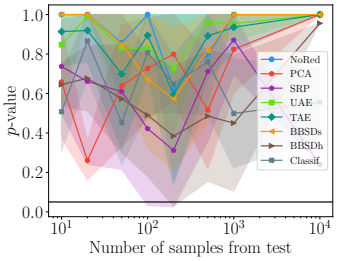
(b) Original split.

Fig. A.44 MNIST randomized and original split, multivariate two-sample tests.

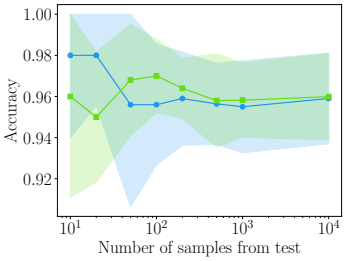
A.2.2 Fashion MNIST



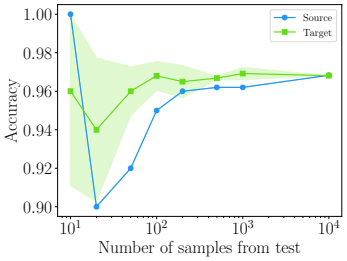
(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.



(c) Randomly shuffled dataset with same split proportions as original dataset.



(d) Original split.

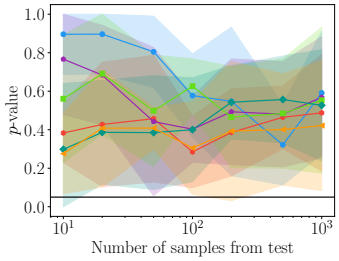
No samples available as Classif did not detect a shift.

(e) Top different samples.

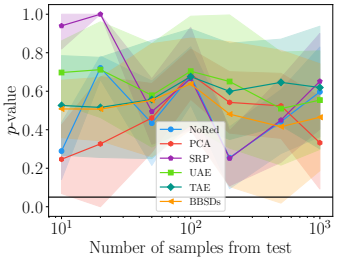
No samples available as Classif did not detect a shift.

(f) Top similar samples.

Fig. A.45 Fashion MNIST randomized and original split, univariate two-sample tests + Bonferroni aggregation.



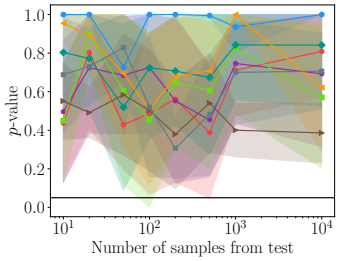
(a) Randomly shuffled dataset with same split proportions as original dataset.



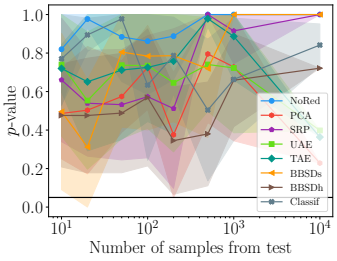
(b) Original split.

Fig. A.46 Fashion MNIST randomized and original split, multivariate two-sample tests.

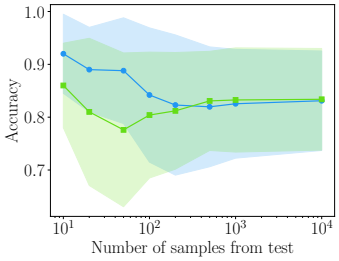
A.2.3 CIFAR-10



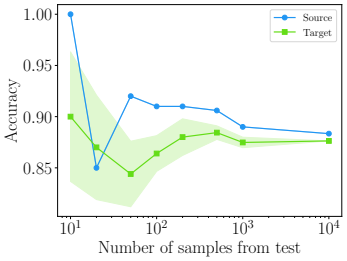
(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.



(c) Randomly shuffled dataset with same split proportions as original dataset.



(d) Original split.

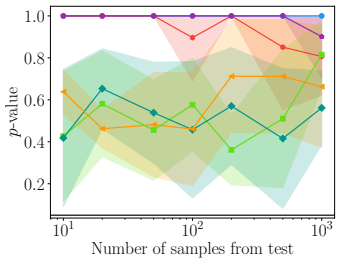
No samples available as Classif did not detect a shift.

(e) Top different samples.

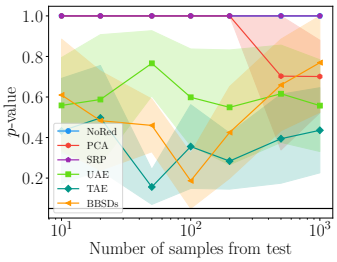
No samples available as Classif did not detect a shift.

(f) Top similar samples.

Fig. A.47 CIFAR-10 randomized and original split, univariate two-sample tests + Bonferroni aggregation.



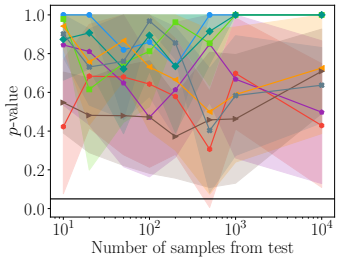
(a) Randomly shuffled dataset with same split proportions as original dataset.



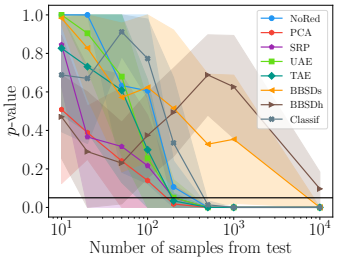
(b) Original split.

Fig. A.48 CIFAR-10 randomized and original split, multivariate two-sample tests.

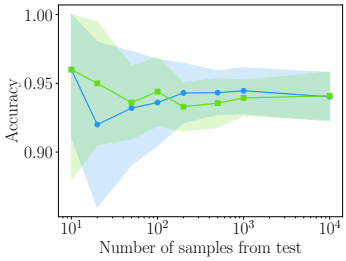
A.2.4 SVHN



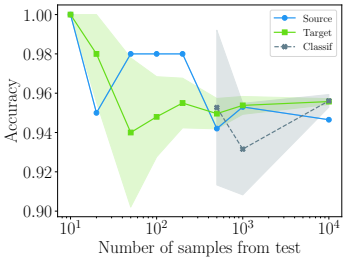
(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.



(c) Randomly shuffled dataset with same split proportions as original dataset.

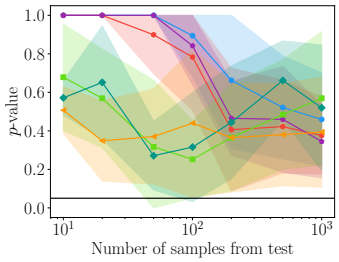


(d) Original split.

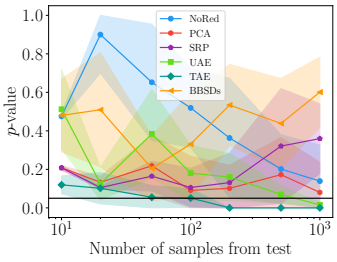
(e) Top different samples.

(f) Top similar samples.

Fig. A.49 SVHN randomized and original split, univariate two-sample tests + Bonferroni aggregation.



(a) Randomly shuffled dataset with same split proportions as original dataset.



(b) Original split.

Fig. A.50 SVHN randomized and original split, multivariate two-sample tests.