Estimating and Explaining Model Performance When Both Covariates and Labels Shift

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deployed machine learning (ML) models often encounter new user data that differs from their training data. Therefore, estimating how well a given model might perform on the new data is an important step toward reliable ML applications. This is very challenging, however, as the data distribution can change in flexible ways, and we may not have any labels on the new data, which is often the case in monitoring settings. In this paper, we propose a new distribution shift model, Sparse Joint Shift (SJS), which considers the joint shift of both labels and a few features. This unifies and generalizes several existing shift models including label shift and sparse covariate shift, where only marginal feature or label distribution shifts are considered. We describe mathematical conditions under which SJS is identifiable. We further propose SEES, an algorithmic framework to characterize the distribution shift under SJS and to estimate a model's performance on new data without any labels. We conduct extensive experiments on several real-world datasets with various ML models. Across different datasets and distribution shifts, SEES achieves significant (up to an order of magnitude) shift estimation error improvements over existing approaches.

1 Introduction

Deployed machine learning (ML) models often face new data different from their training data. For example, mismatch of deployment-development data in geographical locations [20], demographic features [15], and label balance [19] is widely observed and known to affect model performance. Thus, estimating and explaining how a model's performance changes on the new data is an important step toward reliable ML applications.

Estimating and explaining performance shift is challenging for several reasons, however. One major challenge is that the data distribution might shift in flexible ways. Another obstacle is that we often do not have labels on the new data, especially in ML monitoring applications. Without any assumption on the distribution shift, it's impossible to estimate how well the model would perform on the unlabeled new data. Previous work often assumes (i) label shift [21], where feature distributions conditional on the labels are fixed, or (ii) covariate shift [30], where label distributions conditional on features stay the same. However, we often do not know whether the real data shift is limited to label or covariate shift, and naively applying estimation methods designed for one shift may produce inaccurate assessments [24]. Moreover, labels and features may shift simultaneously in practice, invalidating these common assumptions.

To tackle the above challenges, here we propose a new distribution shift model, Sparse Joint Shift (SJS), to consider the joint shift of both labels and a few features. SJS assumes labels and a few features shift, but the remaining features' distribution conditional on the shifted features and labels is fixed. This unifies and generalizes sparse covariate shift and label shift: both of them are SJS,

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

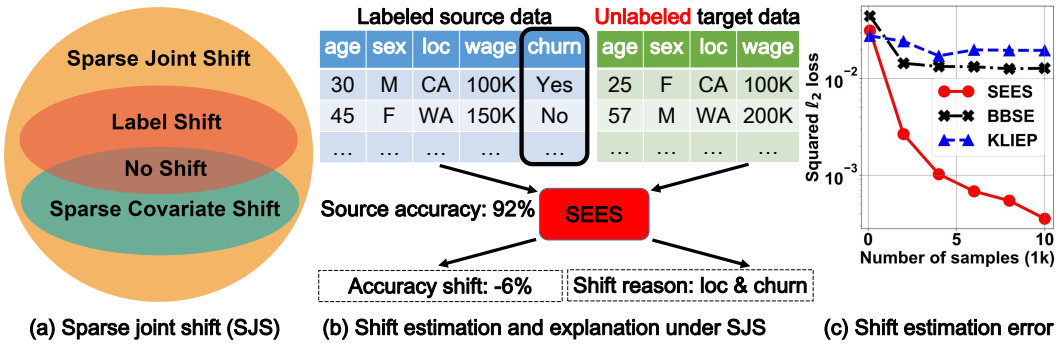


Figure 1: Overview of sparse joint shift (SJS). (a) Both label shift and sparse covariate shift are SJS, but SJS contains additional shifts as well. (b) illustrates SEES, a framework for performance shift estimation and explanation under SJS. Given labeled source and unlabeled target data, SEES exploits the joint shift modeled by SJS to estimate the model performance change and explain which factors drive the shift. In this example, the goal is to predict *churn*. (c) SEES significantly reduces the shift estimation error over existing methods when both labels and covariates shift.

but some SJS is not label or sparse covariate shift (Figure 1). We describe mathematical conditions under which SJS is provably identifiable: if the non-shifted features are weakly correlated, then the marginal feature distribution uniquely determines the joint distribution under SJS. This makes it possible to quantify the shift and estimate model performance on new data without any labels.

When SJS occurs: a motivating example. Consider Alice, a data scientist who built an ML model for customer churn prediction. Two years later, churn rate rose in some states but dropped in others, while the distribution of other features given label (churn) and location (state) remained. This shift in customer distribution is a natural SJS and challenging to estimate without labels on the new data.

Furthermore, we propose SEES, an algorithmic framework for performance shift estimation and explanation under SJS. SEES exploits correlation shifts between features and labels modeled by SJS to improve performance estimation accuracy over existing methods, such as BBSE [21] for label shift and KLIEP [30] for covariate shift (Figure 1). It uses the identified shifted features and labels as a natural explanation of the performance shift. We present an extensive empirical evaluation on real world datasets with both simulated SJS and real shifts. Our experiments validate the effectiveness of SEES: the estimation error of SEES is often an order of magnitude smaller than that of previous state-of-the-art methods. All together, **our contributions are**:

1. We formulate and study sparse joint shift (SJS), a new distribution shift model that considers the joint shift of both labels and features. We show how it unifies and generalizes existing shift models and when it is identifiable with unlabeled data.
2. We propose SEES, a general framework for performance shift estimation and explanation under SJS. We design efficient substantiations of SEES for different data types.
3. We provide comprehensive empirical analysis of our methods. On real world datasets with natural distribution shifts, we found SEES leads to up to 66% performance estimation error reduction over standard approaches.

Related Work. **Label shift:** Label shift has been observed and studied in various domains, such as epidemiologic [5], economics [22], and data mining [13]. Recently, there is an increasing interest in model evaluation and development under under label shift. For example, [21, 3] study how to quantify label shift and evaluate an ML model's performance under label shift accordingly. [38] gives an algorithm for active learning under label shift. Online adaptation to label shift [35] is also shown to be feasible. Label shift can be viewed as a subset of our proposed shift model SJS. Although simpler and easier to estimate, label shift does not capture the joint shift of labels and features. Thus, methods developed for label shift may not work well under SJS.

Covariate shift: Covariate shift [28, 36, 23] is perhaps the most widely adopted assumption in data distribution shift. Since studied in the seminal work [28], various methods have been proposed to estimate covariate shift, including KLIEP [31], KMM [14], and IWCV [29]. Adaptation to covaraite shift has been found useful in many applications, such as spam filtering [4], emotion recognition [18] and human activity detection [17]. More recently, covariate shift adaptation is jointly optimized with model robustness [27], fairness [26], and conformal prediction [33].

Unsupervised model performance evaluation: Model performance evaluation without labels has received relatively limited attention. Domain-specific models' performance can be estimated via certain statistics, such as confidence score [16], rotation prediction [9] and feature statistics of the datasets sampled from a meta-dataset [10] for image recognition. General model evaluation often relies on different assumptions and accessibility. [7] assumes covariate shift and requires users to provide an approximation (slice) of the shifted features, while [6] needs white-box access to the ML models to train an ensemble as a reference. [12] assumes the label distributions are known, while [8] needs a feature independence structure given the labels. When a small number of labels can be obtained, [34] proposes an active model evaluation approach. To our knowledge, this is the first paper that explicitly models the joint shift of both labels and a few features with provable identification guarantees. Moreover, we do not require access to side information such as model design or metadata.

2 Preliminaries and Problem Statement

We start by giving the preliminaries and the problem of estimating and explaining performance shift.

Prediction tasks and ML models. In this paper we consider the standard classification task: given a d-dimensional feature vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ from the feature space $\mathcal{X}$, the goal is to predict its associated label $y \in \mathcal{Y}$ in the label space $\mathcal{Y}$. Let $f(\cdot) : \mathbb{R}^d \mapsto \mathcal{Y}$ denote an ML model designed for such a task. For simplicity, we assume that $\mathcal{Y} = \{1, 2, \cdots, L\}$. Given the model's prediction $f(\boldsymbol{x})$ and its true label y, its performance is quantified by some loss function $\ell(\cdot, \cdot)$. A popular choice is the standard 0-1 loss: $\ell(a, b) = \mathbb{1}_{a=b}$, which we focus on, but other losses are also applicable.

Joint distribution shift. The training and inference data for ML often come from two different distributions, referred to as source domain and target domain. Here, we consider the general case when the joint distribution vary across the source and target domains, and call this *joint distribution shift*. Formally, let $\mathbb{P}_s, \mathbb{P}_t : \mathcal{X} \times \mathcal{Y} \mapsto [0, 1]$ denote the source and target domains, and p_s, p_t be their probability density (or mass) functions. Then joint distribution shift means $p_s(\boldsymbol{x}, y) \neq p_t(\boldsymbol{x}, y)$.

Problem statement. Suppose we are given a labeled dataset $D_s \triangleq \{(\boldsymbol{x}^{s,i}, y^{s,i})\}_{i=1}^{n_s}$ from the source distribution $\mathbb{P}_s$, an unlabeled dataset $D_t \triangleq \{(\boldsymbol{x}^{t,i})\}_{i=1}^{n_t}$ from the target distribution $\mathbb{P}_t$, and an ML model $f(\cdot)$ predicting the associated label given any feature vector $\boldsymbol{x}$. Our goal is to estimate how much performance changes from the source domain to the target domain. More formally, we aim at estimating the performance shift $\Delta \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}_t}[\ell(f(\boldsymbol{x}), y)] - \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}_s}[\ell(f(\boldsymbol{x}), y)]$. This is challenging as we do not observe labels on the target domain. In many applications, attributing the performance shift to certain features is also desired. Thus, we are also interested in identifying a set of features to explain the performance shift.

3 SJS: A Tractable Unification of Label Shift and Sparse Covariate Shift

At a first glance, estimating the performance shift under joint distribution shift without observing labels from target domain seems hopeless: if the marginal feature distributions are identical for both domains, then observing the features alone should give 0 as the estimated performance shift. However, the label distribution given any feature on the target domain is arbitrary, and thus the estimated shift can be arbitrarily bad. In other words, joint distribution shift is not identifiable with no target labels.

To mitigate nonidentifiability, it's necessary to make additional assumptions. The most popular assumptions in literature are label shift [21] and covariate shift [28]. Label shift assumes that only label distribution may change, but the feature distribution given a label remains, i.e., $p_s(\boldsymbol{x}|y) = p_t(\boldsymbol{x}|y)$. On the other hand, covariate shift assumes that feature distribution can shift, but the label distribution given the features is fixed, i.e., $p_s(y|\boldsymbol{x}) = p_t(y|\boldsymbol{x})$. However, those assumptions disallow

simultaneous changes of both features and labels, which often happen in real-world data [20, 25, 32]. To enable joint feature and label estimation which is tractable, we introduce a subclass of joint distribution shift, *Sparse Joint Shift* (SJS), as follows.

Definition 1 (Sparse Joint Shift (SJS))**.** *Suppose for some integer $s \leq d$ and an index set $\mathcal{I} \subset [d]$ with size at most s ($|\mathcal{I}| \leq s$), $p_s(\boldsymbol{x}_{\mathcal{I}^c}|\boldsymbol{x}_{\mathcal{I}}, y) = p_t(\boldsymbol{x}_{\mathcal{I}^c}|\boldsymbol{x}_{\mathcal{I}}, y)$. Then we say the source and target pair (p_s, p_t) is under s-Sparse Joint Shift, or s-*SJS*. Here, $\mathcal{I}^c \triangleq [d] - \mathcal{I}$. We call $\mathcal{I}$ the shift index set.*

Roughly speaking, SJS allows both labels and a few features to shift, but assumes the remaining features' conditional distribution to stay the same. Next, we will study when this assumption allows tractable performance shift estimation. All proofs are left to the appendix.

3.1 When is sparse joint shift identifiable?

Recall that additional assumptions are needed because the general joint distribution shift is not identifiable. However, when $s = d$, s-SJS simply becomes general joint distribution shift. Thus, it is worthy understanding when s-SJS resolves the identifiability issue. To do so, let us first formally introduce the notation of identifiability.

Definition 2 (Identifiable)**.** *Suppose the source-target tuple (p_s, p_t) is under s-*SJS*. (p_s, p_t) is identifiable if and only if for any alternative distribution $p_a(\boldsymbol{x}, Y)$, if $p_a(\boldsymbol{x}) = p_t(\boldsymbol{x})$ and $\exists \mathcal{J} \subset [d], |\mathcal{J}| \leq s$, such that $p_a(\boldsymbol{x}_{\mathcal{J}^c}|\boldsymbol{x}_{\mathcal{J}}, y) = p_s(\boldsymbol{x}_{\mathcal{J}^c}|\boldsymbol{x}_{\mathcal{J}}, y)$, then $p_a(\boldsymbol{x}, y) = p_t(\boldsymbol{x}, y)$.*

The identifiability can be easily interpreted in words: If a joint feature and label distribution matches the target feature distribution and satisfies the s-SJS requirement together with the source distribution, it has to be the target distribution. The following statement shows when (p_s, p_t) is identifiable.

Theorem 1. *Suppose (p_s, p_t) is under s-*SJS*. Assume for any set $\mathcal{J} \subset [d], |\mathcal{J}| \leq s$ and any fixed $\bar{\boldsymbol{x}} \in \mathcal{X}$, the probability density (or mass) functions $\{p_s(\boldsymbol{x}_{\mathcal{J}^c \cap \mathcal{I}^c}, \boldsymbol{x}_{\mathcal{J} \cup \mathcal{I}} = \bar{\boldsymbol{x}}_{\mathcal{J} \cup \mathcal{I}}, y = i)\}_{i=1}^{L}$ are linearly independent. Then (p_s, p_t) is identifiable.*

This statement sheds light on why uniquely identifying the target distribution without target label is feasible under sparse joint shift. Roughly speaking, s-SJS requires that given the shifted features and labels, the remaining features' distribution remains the same on both domains. If those remaining features are different enough (linear independence), they can uniquely determine the distribution of the shifted features and labels. We stress that the linear independence is necessary: if it does not hold, then for any s, we can always find some source-target pair (p_s, p_t) which is not identifiable. Linear independence implicitly requires sparsity: if $s > d/2$, then $\mathcal{J}^c \cap \mathcal{I}^c$ can be empty and the linear independence does not hold. In other words, the sparsity is necessary for the shift to be identifiable.

3.2 How does SJS relate to label shift and covariate shift?

A natural question is how does SJS relates to standard label shift and covariate shift. The following statement formally gives the answer.

Theorem 2. *If (p_s, p_t) is under label shift, then it is also under* 0-SJS*. If (p_s, p_t) is under sparse covariate shift, i.e., $p_s(\boldsymbol{x}_{\mathcal{I}^c}|\boldsymbol{x}_{\mathcal{I}}) = p_t(\boldsymbol{x}_{\mathcal{I}^c}|\boldsymbol{x}_{\mathcal{I}})$ for some index set $\mathcal{I}$ with size $s < d$, then it is also under s-*SJS*. In addition, there exists (p_s, p_t) under s-*SJS *such that it is under neither label shift or covariate shift.*

There are several takeaways. First, label shift implies SJS without additional requirements. In fact, as certain distribution pairs are under SJS but not label shift, SJS is strictly more general than label shift. Second, SJS also includes sparse covariate shift. When $s = d$, SJS completely unifies both label shift and covariate shift, though it is not identifiable. Identifiable SJS, on the other hand, unifies label shift and sparse covariate shift. Finally, SJS also allows shifts not covered by label shift and covaraite shift: the correlation between label and (a set of) features can be shifted.

4 Shift Estimation and Explanation under Sparse Joint Shift

Now we present SEES, an algorithmic framework to estimate and explain the performance shift Δ when the source and target domain is under s-SJS. As shown in Figure 2, it consists of three components: a sparsity-aware density matcher, an empirical gap calculator, and a shifted feature

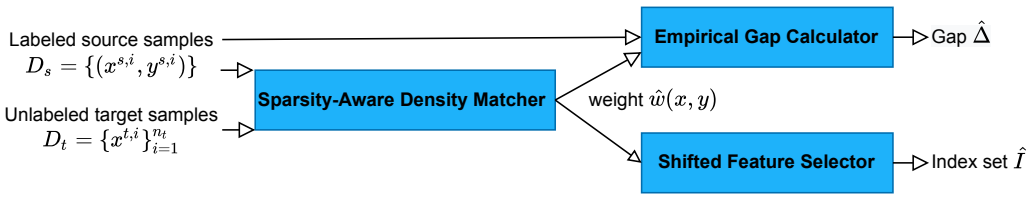


Figure 2: How SEES works. Given labeled source and unlabeled target data, SEES uses a sparsity-aware density matcher to learn a weight function $\hat{w}(\boldsymbol{x}, y)$. Next, an empirical gap calculator computes the performance gap $\hat{\Delta}$ by weighing the source samples with the learned $\hat{w}(\boldsymbol{x}, y)$. The shifted feature selector extracts the features $\hat{I}$ on which the weight function depends heavily.

selector. Given the labeled source samples and unlabeled target samples, we first adopt the sparsity-aware density matcher to obtain an estimated ratio of the target and source density functions, denoted by $\hat{w}(\boldsymbol{x}, y)$. Next, the empirical gap calculator is responsible to estimate the performance shift Δ via appropriately reweighting source samples with ratio $\hat{w}(\boldsymbol{x}, y)$. Finally, the shifted feature selector picks a set of features as the explanation for the shift. We explain each component as follows.

4.1 Sparsity-aware density matcher

A key component of SEES is the sparsity-aware density matcher. Here we want to find some weight function $\hat{w}(\boldsymbol{x}, y)$ to induce an estimated target distribution $\hat{p}_t(\boldsymbol{x}, y) \triangleq \hat{w}(\boldsymbol{x}, y) \cdot p_s(\boldsymbol{x}, y)$. Our goals include (i) a small distance between the estimated target distribution and the true target distribution, (ii) s-SJS between the source and the estimated target distributions, and (iii) flexible parameterization of the weight function. To achieve those goals, we propose the following optimization framework

$$\begin{aligned} &\min_{w(\boldsymbol{x},y)\in\mathcal{W}} D(p_t(\boldsymbol{x}), \hat{p}_t(\boldsymbol{x})) \\ &\quad \text{s.t. } \hat{p}_t(\boldsymbol{x}) = \sum_{y=1}^{L} w(\boldsymbol{x}, y) \cdot p_s(\boldsymbol{x}, y), \text{ and } w(\boldsymbol{x}, y) \text{ depends on at most } s \text{ features of } \boldsymbol{x}. \end{aligned} \tag{4.1}$$

Here, $D(\cdot, \cdot)$ is some distance metric that measures the difference between two density functions. We minimize the distance between the induced feature density $\hat{p}_t(\boldsymbol{x})$ and the target feature density $p_t(\boldsymbol{x})$. The minimization is not over joint label and feature distributions since target labels are not available. The induced feature density function can be easily derived from source density function and the weight function, encoded in the first constraint. s-SJS is enforced by the second constraint: s-SJS means given s features and labels, the distributions of remaining features are fixed across source and the induced domain, which holds if and only if their density ratio $w(\boldsymbol{x}, y)$ only depends on those s features. $\mathcal{W}$ represents the set of all feasible weight functions. Different parameterization can be easily realized by adopting different $\mathcal{W}$. Assume access to density functions $p_s(\boldsymbol{x}, y)$ and $p_t(\boldsymbol{x})$, and a weight function set $\mathcal{W}$ containing the true weight $w^*(\boldsymbol{x}, y) \triangleq \frac{p_t(\boldsymbol{x},y)}{p_s(\boldsymbol{x},y)}$. One can easily show the above optimization returns the true weight function $w^*(\boldsymbol{x}, y)$ for identifiable s-SJS.

One benefit of the above framework is the flexibility of concrete instantiations. Different design choices, including the distance metric $D(\cdot, \cdot)$ and the weight parameterization space $\mathcal{W}$, can fit different feature types, incorporate domain knowledge, and tradeoff different sample and computational complexity. We give two instantiations of the above optimization: SEES-c for continuous features, and SEES-d for discrete features.

SEES-c: SEES for continuous features. For continuous features, we use KL-divergence $D_{KL}(\cdot, \cdot)$ as the distance metric, and initialize the parameterization space $\mathcal{W}$ by linear combinations of K fixed basis functions, $\phi_1(\boldsymbol{x}, y), \phi_2(\boldsymbol{x}, y), \cdots, \phi_K(\boldsymbol{x}, y)$. That is to say, $\mathcal{W} = \{w(\boldsymbol{x}, y) | w(\boldsymbol{x}, y) = \sum_{k=1}^{K} a_{k,y}\phi_k(\boldsymbol{x}, y), a_{k,y} \geq 0, \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}_s}\left[\sum_{k=1}^{K} a_{k,y}\phi_k(\boldsymbol{x}, y)\right] = 1, \alpha_{k,y} \geq 0\}$. The last two constraints ensure $w(\boldsymbol{x}, y) \cdot p_s(\boldsymbol{x}, y)$ is a valid probability density. Those basis functions encode users' prior knowledge about the shift. A simple choice, for example, is linear functions: setting $K = d$ and $\phi_k(\boldsymbol{x}, y) = \boldsymbol{x}_k$. To model the dependence on different features, let e_i denote all indexes k such

that $\phi_k(\cdot)$ depends on feature $\boldsymbol{x}_i$, and introduce a vector $\boldsymbol{\beta} \in \mathbb{R}^d$ such that $\boldsymbol{\beta}_i \triangleq \sqrt{\sum_{k \in e_i} \sum_{y=1}^{L} a_{k,y}^2}$. The feature dependence requirement in Problem 4.1 is equivalent to sparsity constraint $\|\boldsymbol{\beta}\|_0 \leq s$. We can relax the 0-norm by 1-norm and obtain one instantiation of Problem 4.1 as

$$\max_{a_{1,1}, a_{1,2}, \cdots, a_{K,L}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_t} \left[\log \sum_{y=1}^{L} p_s(y|\boldsymbol{x}) \sum_{k=1}^{K} a_{k,y} \phi_k(\boldsymbol{x}, y) \right] + \eta \sum_{i=1}^{d} \sqrt{\sum_{k \in e_i} \sum_{y=1}^{L} a_{k,y}^2}$$
$$s.t.\ \mathbb{E}_{(\boldsymbol{x},y) \sim \mathbb{P}_s} \left[\sum_{k=1}^{K} a_{k,y} \phi_k(\boldsymbol{x}, y) \right] = 1, \alpha_{k,y} \geq 0$$

where $\eta > 0$ controls the trade-off between sparsity and the KL distance. One benefit of this instantiation is computational efficiency: the constraint is linear in the optimization variables, and the objective is convex. Thus, the problem is convex and can be efficiently solved. The label distribution given feature $p_s(y|\boldsymbol{x})$ is unknown but can be approximated by the ML model $f(\cdot)$ trained on the source domain. Given finite samples, the expectations can be replaced by their empirical estimation.

SEES-d: SEES for discrete features. Another interesting instantiation exists for discrete features. With no prior knowledge, we parameterize $\mathcal{W}$ to include all possible s-SJS: specifically, $\mathcal{W}$ contains all tuple $(J, w_J(\boldsymbol{x}_J, y))$, where index set $J \subset [d], |J| = s$ represents the shifted features, and weight function $w_J(\boldsymbol{x}_J, y)$ only depends on $\boldsymbol{x}_J$ and y.

Features only take finite values, so we can view the density (mass) functions as vectors with finite dimensions. Thus, we adopt the squared ℓ_2 distance, i.e., $D(\boldsymbol{z}, \boldsymbol{z}') = \sum_{i=1}^{|\boldsymbol{z}|} (\boldsymbol{z}_i - \boldsymbol{z}'_i)^2$ to measure distance. However, naively measuring ℓ_2 distance between $p_t(\boldsymbol{x})$ and $\hat{p}_t(\boldsymbol{x})$ leads to a computational complexity exponential in d. Instead, we measure the distance on a set of marginal densities: given an index set J, for every index set with size $2s$ that contains J, denoted by κ, we measure the squared ℓ_2 distance between $p_t(\boldsymbol{x}_\kappa, f(\boldsymbol{x}))$ and $\hat{p}_t(\boldsymbol{x}_\kappa, f(\boldsymbol{x}))$, and then aggregate over κ. This design leads to the following instantiation of Problem 4.1

$$\min_{J, w_J(\boldsymbol{x}, y)} \sum_{\kappa: J \subseteq \kappa, |\kappa| = 2s} \sum_{\bar{f}=1}^{L} \sum_{\bar{\boldsymbol{x}}_\kappa \in \mathcal{X}_\kappa} \|p_t(\bar{\boldsymbol{x}}_\kappa, \bar{f}) - \sum_{\bar{y}=1}^{L} w_J(\bar{\boldsymbol{x}}_\kappa, \bar{y}) \cdot p_s(\bar{\boldsymbol{x}}_\kappa, \bar{f}, \bar{y})\|_2^2,\ s.t. \|J\| = s \quad (4.2)$$

where $p_s(\bar{\boldsymbol{x}}_\kappa, \bar{f})$ and $p_t(\bar{\boldsymbol{x}}_\kappa, \bar{f}, \bar{y})$ are short for $p_s(\boldsymbol{x}_\kappa = \bar{\boldsymbol{x}}_\kappa, f(\boldsymbol{x}) = \bar{f})$ and $p_t(\boldsymbol{x}_\kappa = \bar{\boldsymbol{x}}_\kappa, f(\boldsymbol{x}) = \bar{f}, y = \bar{y})$, respectively.

Compared to the naive approach, the above formulation is much more computationally efficient: the number of parameters in the above objective is only polynomial in the feature dimension d. For fixed J, the problem is simply a linear regression over the weight $w_J(\boldsymbol{x}_J, y)$ and thus can be efficiently solved. In practice, one can estimate $p_s(\bar{\boldsymbol{x}}_\kappa, \bar{f})$ and $p_t(\bar{\boldsymbol{x}}_\kappa, \bar{f}, \bar{y})$ via labeled source and unlabeled target samples, and then solve the empirical version of the above problem. Compared to using KL-divergence, solving the empirical version produces the correct shifted index set and a weight function close to the true weight $w^*(\boldsymbol{x}, y)$ (under mild conditions). This is formally stated as follows.

Theorem 3. *Consider when all features are discrete, i.e., for each i, $\boldsymbol{x}_i \in \{1, 2, \cdots, v\}$. Suppose (i) the source and target are under exact s-SJS, (ii) for any set $\mathcal{J} \subset [d], |\mathcal{J}| \leq s$ and any $\bar{\boldsymbol{x}} \in \mathcal{X}$, the marginal probability density (or mass) functions $\{p_s(f(\boldsymbol{x}), \boldsymbol{x}_{\mathcal{J} \cup \mathcal{I}} = \bar{\boldsymbol{x}}_{\mathcal{J} \cup \mathcal{I}}, y = i)\}_{i=1}^{d}$ are linearly independent, and (iii) $w(\boldsymbol{x}, y)$ is bounded by a constant M. Then there exists some constant c (independent of d, n_s and n_t), such that if $\sqrt{\frac{1}{2n_s}} + LM\sqrt{\frac{1}{2n_t}} < c/\sqrt{2s \log d + s \log v + 2 \log L + \log 1/\delta}$, then with probability $1 - \delta$, (i) the index set $\hat{J}$ learned by Problem 4.2 matches the true shift index set I, and (ii) the produced weights $w_{\hat{J}}(\boldsymbol{x}_{\hat{J}}, y)$ satisfies $\left| w_{\hat{J}}(\boldsymbol{x}_{\hat{J}}, y) - w^*(\boldsymbol{x}, y) \right| \leq O\left(\sqrt{2s \log d + s \log v + 2 \log L + \log 1/\delta} \left(\sqrt{\frac{1}{2n_s}} + LM\sqrt{\frac{1}{2n_t}} \right) \right)$.*

Roughly speaking, this statement ensures that, when source and target sample sizes are large enough, with high probability, the true shift index set can be identified with finite samples, and the error rate of the learned weight function is approximately the inverse of sample sizes' square root.

Comparisons of SEES-c and SEES-d. SEES-d enjoys mathematical guarantees, but SEES-c can be computationally more efficient. In practice, we can discretize continuous features to use SEES-d.

4.2 Empirical gap calculator and shifted feature selector

Now we explain how the other two compoents of SEES work. The empirical gap calculator computes the performance shift $\hat{\Delta}$ via three steps. First, it estimates the source performance by $\frac{1}{n_s}\sum_{i=1}^{n_s}\ell(\boldsymbol{x}^{s,i}, y^{s,i})$. Next, it estimates the performance on the induced target distribution. Note that the performance on the induced target domain is $\int \hat{p}_t(\boldsymbol{x}, y)\ell(\boldsymbol{x}, y)d\boldsymbol{x}dy = \int \hat{w}(\boldsymbol{x}, y)p_s(\boldsymbol{x}, y)\ell(\boldsymbol{x}, y)d\boldsymbol{x}dy = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathbb{P}_s}[\hat{w}(x, y)\ell(\boldsymbol{x}, y)]$. Thus, we use the weighted loss on the source samples $\frac{1}{n_s}\sum_{i=1}^{n_s}\hat{w}(\boldsymbol{x}^{s,i}, y^{s,i})\ell(\boldsymbol{x}^{s,i}, y^{s,i})$ as the estimation. Finally, their difference, i.e., $\hat{\Delta} = \frac{1}{n_s}\sum_{i=1}^{n_s}(\hat{w}(\boldsymbol{x}^{s,i}, y^{s,i}) - 1)\ell(\boldsymbol{x}^{s,i}, y^{s,i})$ is returned as the estimated performance shift.

The shifted feature selector picks a set of features as the shift explanation. For discrete data, the weight function $\hat{w}(\boldsymbol{x}, y)$ learned by the density matcher's instantiation is parameterised as a shifted index $\hat{J}$ and the corresponding weight $\hat{w}_{\hat{J}}(\boldsymbol{x}_{\hat{J}}, y)$. Thus, a natural choice is to return $\hat{I} = \hat{J}$ as the explanation. For continuous data, the weight function is $\hat{w}(\boldsymbol{x}, y) = \sum_{k=1}^{K}\hat{a}_{k,y}\phi_k(\boldsymbol{x}, y)$, where $\hat{a}_{k,y}$ is learned by the corresponding instantiation. Recall that e_i denotes all basis functions that depends on feature i. Then $\hat{\boldsymbol{\beta}}_i \triangleq \sqrt{\sum_{k\in e_i}\sum_{y=1}^{L}a_{k,y}^2}$ can be viewed as the total contribution of feature i. Thus, a simple choice is to pick features with the s largest contributions. Formally, we use $\hat{I} = \{i|\hat{\boldsymbol{\beta}}_i > \hat{\boldsymbol{\beta}}_{(d-s)}\}$, where $\hat{\boldsymbol{\beta}}_{(d-s)}$ is the $d-s$ smallest value in $\hat{\boldsymbol{\beta}}$.

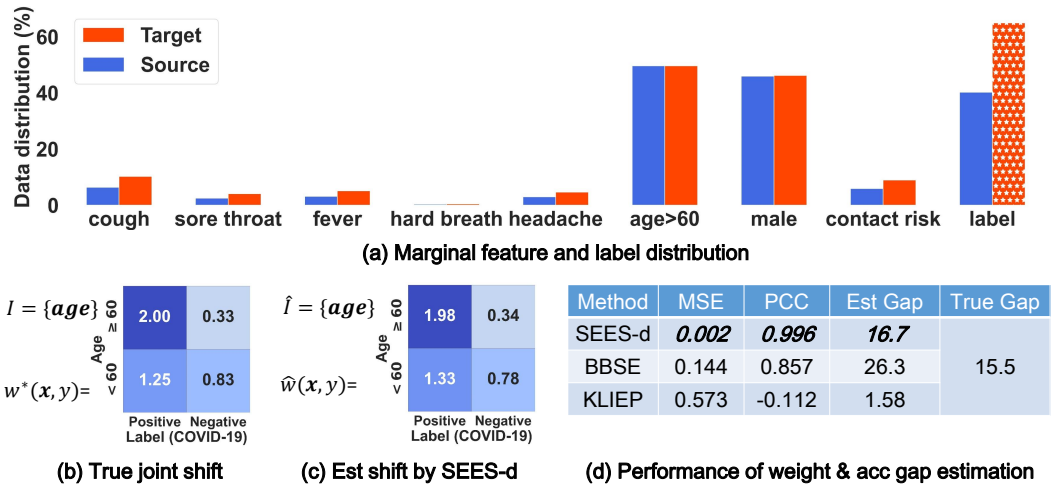
5 Experiments

In this section, we study the performance of SEES on several real world datasets with synthetic and natural distribution shifts. Our goal is four-fold: (i) understand when and how SEES estimates the performance shift, (ii) evaluate the trade-offs between the estimation performance reached by SEES and the required dataset sizes, (iii) explore the effects of shift sparsity on SEES' performance, and (iv) validate the effectiveness of SEES on datasets with real world distribution shifts.

Datasets, ML models and baselines. Six datasets are used for evaluation purposes. we first simulate various SJS on BANKCHURN [1], COVID-19 [2], and CREDIT [37] to systematically understand the performance of SEES. Next, we apply SEES on EMPLOY, INCOME, and INSURANCE [11] with real world distribution shifts and perform an in-depth analysis. We use a gradient boosting tree model as the ML model, and results for more models can be found in the Appendix. For comparison, we adopt two state-of-the-art methods for comparison: BBSE [21] for label shift and KLIEP [30] for covariate shift. More details on the experiments can be found in the Appendix.

Case study. We start with a case study on the dataset COVID-19. The task is to predict whether a person tests positive for COVID-19. We simulate a joint shift of the feature aged and label. Specifically, both source and target data contain 5000 young and aged individuals. The positive rate is 40% for both young and aged group from the source. In the target data, we raise the positive rate to 80% for aged group and 50% for young group. This simulates a shift due to a COVID variant more harmful to the elder than its ancestor. We adopt SEES-d as all features are categorical.

Figure 3 summarizes this case study. First note that identifying which feature is shifted is not obvious. As shown in Figure 3(a), the marginal distribution of most features except age and gender has changed from the source to the target. The actual joint shift (Figure 3(b)) is, on the other hand, due to age group and the labels. Identifying shifted age is challenging as the label on the target (last red bar in Figure 3(a)) also shifts but cannot be observed. On the other hand, SEES-d correctly identifies the shifted feature age, and produces a weight function close to the true weight (Figure 3(b) and (c)). This is primarily because SEES-d explicitly exploits the joint shift modeled by SJS. In fact, SEES-d's performance is significantly better than existing methods. As shown in Figure 3(d), the mean square error (MSE) between the true weights and learned weights is only 0.002 when adopting SEES-d, but 0.144 and 0.573 when using BBSE and KLIEP, respectively. The Pearson correlation coefficient (PCC) between the true weights and weights learned by SEES-d is 0.996, indicating a strong correlation. The weight estimation performance directly affects how precise the estimated



Method	MSE	PCC	Est Gap	True Gap
SEES-d	*0.002*	*0.996*	*16.7*	15.5
BBSE	0.144	0.857	26.3	
KLIEP	0.573	-0.112	1.58	

Figure 3: A case study on the COVID-19 dataset. (a) The marginal distribution of labels and all features. The label on the target domain (the last red bar) is not observable. (b) The actual joint shift between source and target data. (c) The mean square error (MSE), pearson correlation coefficient (PCC) between learned and true weights, and the the estimated accuracy gap. Overall, SEES-d significantly improves estimation performance over existing methods.

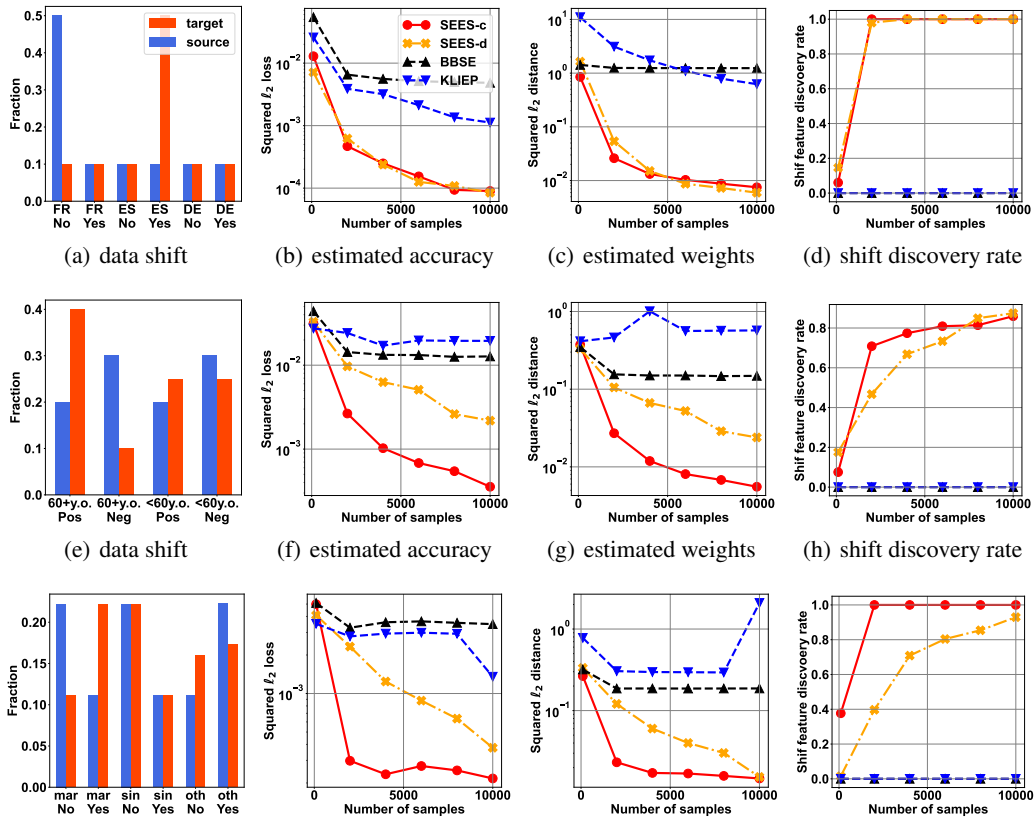
accuracy gap is. The estimated gap $\hat{\Delta}$ of SEES-d is 16.7%, which is close to the true gap (15.5%). By contrast, BBSE tends to overestimates the gap (26.3$) while KLIEP underestimates it (1.58%).

Trade-offs between estimation performance and sample size. Next, we study the trade-offs between estimation performance and the number of available samples. For simplicity, we simulate various sparse joint shifts with $s = 1$ via (i) specifying marginal distribution of the shifted feature and labels first, and (ii) then drawing random samples from the original dataset conditional on values of specified labels and shifted feature. Same number of samples are allocated to both source and target datasets. Figure 4 shows the simulated data shift (column 1), the squared ℓ_2 loss of accuracy estimation (column 2), weight estimation (column 3), and the shifted feature discovery rate (column 4) for three datasets. Overall, we observe that the estimation error of SEES-c and SEES-d diminishes as the number of samples increases, while that of BBSE and KLIEP is almost flat. This is becausse BBSE and KLIEP ignore the change of correlation between features and labels. In addition, SEES-c SEES-d can often identify the shifted feature correctly given large number of samples, while BBSE and KLIEP cannot. Interestingly, while no theoretical guarantee is known for SEES-d, its estimation performance often matches or even surpasses the instantiation designed for discrete data.

Table 1: Root mean square error of estimated accuracy gap (%) under real shifts for a gradient boosting model. The numbers are averaged over all source-target pairs in each dataset. Results for other models (e.g., a neural network) can be found in the Appendix. For each dataset and ML model, SEES provides significant estimation error reduction over baselines.

EMPLOY				INCOME				INSURANCE			
SEES-c	SEES-d	BBSE	KLIEP	SEES-c	SEES-d	BBSE	KLIEP	SEES-c	SEES-d	BBSE	KLIEP
2.90	3.00	5.20	5.20	**1.90**	2.40	3.00	3.40	**1.70**	2.20	2.10	5.00

Effects of shift sparsity. We have focused on 1-SJS for simplicity, but how the sparsity s affects the estimation performance remains unknown. To answer this, we fix the number of samples to be 10,000, and then measure the performance of all compared methods for joint distribution of label and different number of features. Overall, we observe that the estimation error often grows as more features shift jointly with the labels. This is because more shifted features often implies higher complexity in the distribution shifts and thus more parameters to estimate. It is worthy noting that 0-SJS degenerates to label shift, and the performance of SEES-c and SEES-d is slightly worse



(a) data shift (b) estimated accuracy (c) estimated weights (d) shift discovery rate

(e) data shift (f) estimated accuracy (g) estimated weights (h) shift discovery rate

(i) data shift (j) estimated accuracy (k) estimated weights (l) shift discovery rate

Figure 4: Trade-offs between shift estimation and sample size. We vary the shifted features (first column), and measure performance of the estimated accuracy shift (second column), estimated weights (third column), and how often the true shifted features are discovered (last column). The first, second, and third row corresponds to dataset BANKCHURN, COVID-19, and CREDIT, respectively. Overall, both SEES-c and SEES-d consistently outperforms existing estimation approaches.

than BBSE (designed for label shift) under 0-SJS. This is a trade-off for covering more general distribution shifts. More details can be found in the appendix.

Accuracy gap estimation on real shifts. Finally we validate the effectiveness of SEES on accuracy estimation with real world shifts. EMPLOY and INCOME are partitioned by geography (states) and INSURANCE is partitioned by time (year). For each partition pair, we train a gradient boosting model on one and estimate its performance on the other. Table 1 shows the estimation error averaged over all partition pairs for each dataset. SEES consistently outperforms BBSE and KLIEP, and reduces the estimation error by up to 66% (1-1.7/5.0). We also evaluate other models (including a neural network) and observe similar results. More results can be found in the Appendix.

6 Conclusion

In this paper, we propose Sparse Joint Shift (SJS), a new distribution shift model that accounts for both label and covariate shifts. We show how SJS unifies and generalizes existing distribution shift models and remains identifiable under reasonable assumptions. We develop SEES, an algorithmic framework for unsupervised model performance estimation and explanation under SJS. Both theoretical analysis and empirical study validates the effectiveness of SEES. Our work contributes to making ML more reliable when data can change. A natural next step is how to improve estimation performance under SJS when a small number of target labels can be queried. This paper focuses on model performance estimation and explanation. Developing ML models robust to different SJS is also an open question.

References

[1] The BANKCHURN dataset. https://www.kaggle.com/code/kmalit/bank-customer-churn-prediction/notebook. [Accessed 2022].

[2] The COVID-19 government dataset. https://data.gov.il/dataset/covid-19. [Accessed 2022].

[3] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *International Conference on Machine Learning*, pages 222–232. PMLR, 2020.

[4] Steffen Bickel and Tobias Scheffer. Dirichlet-enhanced spam filtering based on biased samples. *Advances in neural information processing systems*, 19, 2006.

[5] Alfred A Buck and John J Gart. Comparison of a screening test and a reference test in epidemiologic studies: I. indices of agreement and their relation to prevalence. *American Journal of Epidemiology*, 83(3):586–592, 1966.

[6] Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. *Advances in Neural Information Processing Systems*, 34, 2021.

[7] Mayee Chen, Karan Goel, Nimit S Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Ré. Mandoline: Model evaluation under distribution shift. In *International Conference on Machine Learning*, pages 1617–1629. PMLR, 2021.

[8] Ching-Yao Chuang, Antonio Torralba, and Stefanie Jegelka. Estimating generalization under distribution shifts via domain-invariant representations. *arXiv preprint arXiv:2007.03511*, 2020.

[9] Weijian Deng, Stephen Gould, and Liang Zheng. What does rotation prediction tell us about classifier accuracy under varying testing environments? In *International Conference on Machine Learning*, pages 2579–2589. PMLR, 2021.

[10] Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15069–15078, 2021.

[11] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[12] Pinar Donmez, Guy Lebanon, and Krishnakumar Balasubramanian. Unsupervised supervised learning i: Estimating classification and regression errors without labels. *Journal of Machine Learning Research*, 11(4), 2010.

[13] George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008.

[14] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[15] Patrick Grother, Mei Ngan, and Kayee Hanaoka. Face recognition vendor test part 3: Demographic effects, 2019-12-19 2019.

[16] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1134–1144, 2021.

[17] Hirotaka Hachiya, Masashi Sugiyama, and Naonori Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012.

[18] Suwicha Jirayucharoensak, Setha Pan-Ngum, and Pasin Israsena. Eeg-based emotion recognition using deep learning network with principal component based covariate shift adaptation. *The Scientific World Journal*, 2014, 2014.

[19] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Comput. Surv.*, 52(4):79:1–79:36, 2019.

[20] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[21] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[22] Charles F Manski and Steven R Lerman. The estimation of choice probabilities from choice based samples. *Econometrica: Journal of the Econometric Society*, pages 1977–1988, 1977.

[23] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[24] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[26] Ashkan Rezaei, Anqi Liu, Omid Memarrast, and Brian D Ziebart. Robust fairness under covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9419–9427, 2021.

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020.

[28] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[29] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[30] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in neural information processing systems*, 20, 2007.

[31] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

[32] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[33] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32, 2019.

[34] Peter Welinder, Max Welling, and Pietro Perona. A lazy man's approach to benchmarking: Semisupervised classifier evaluation and recalibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3262–3269, 2013.

[35] Ruihan Wu, Chuan Guo, Yi Su, and Kilian Q Weinberger. Online adaptation to label distribution shift. *Advances in Neural Information Processing Systems*, 34, 2021.

[36] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Advances in neural information processing systems*, 24, 2011.

[37] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480, 2009.

[38] Eric Zhao, Anqi Liu, Animashree Anandkumar, and Yisong Yue. Active learning under label shift. In *International Conference on Artificial Intelligence and Statistics*, pages 3412–3420. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] . See Appendix A.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] . See Appendix A.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] .
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] .
   (b) Did you include complete proofs of all theoretical results? [Yes] .See Appendix B
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . Please see Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . Please see Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] .
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] .
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] .
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] .