Abstract

During training and classification, instances are drawn from the instance space and mapped to the feature space. We focus on the problem of detecting hidden changes in the functions that map instances to feature vectors during classification. We call such changes feature shift and introduce an on-line method for detecting it. Our method is based on a robust similarity measure that uses one-class SVM to monitor distributional changes in the feature space. Unlike previous methods, ours can distinguish between changes in priors and feature shift. The method is empirically evaluated on visual categorization tasks and its advantage verified.

1. Introduction

In concept learning theory, the feature space can be interpreted as a geometrical model of the instance space. The instance space contains the concept to be learned, and the feature space contains the descriptions specified by the user. During training, tagged instances are drawn from the instance space and mapped to the feature space, where the learning algorithm is employed. In traditional setups, the concept remains unchanged during training and classification. There are, however, many real-world setups where its projected image in the feature space changes during classification. We call such changes feature shift.

Feature shift can harm the performance of the classification system. If some of the feature extraction procedures change their behavior during the classification process, the classifier’s performance can deteriorate. For example, in medical diagnosis systems, the concept can be the group of ill patients. A drift in measurement of an MRI scanner or some other laboratory device may cause changes to the representation of the concept in the feature space, while the concept itself — the patient’s illness — obviously remains unchanged. An incorrect diagnosis might result. Another example is in industrial defect classification problems, where inconsistencies in an image acquisition device are considered one of the major obstacles to maintaining the stability of statistical process control (SPC) machines [1].

This research focuses on the problem of detecting hidden changes in the functions that map instances to feature vectors during classification while the concept itself remains unchanged. Once feature shift has been detected, we can stop the classification process, or, alternatively, correct the classifier. There are setups, however, where retraining the classifier is too expensive. In other setups, such as industrial classification systems, retraining is not allowed since changes in the classification performance may destabilize the production process. In such cases, feature shift can be handled by fixing the feature extraction procedure.

To be effective, we require our desired feature shift detector to have the following properties: (1) It should have high precision and recall to avoid false alarms and detection failures; (2) It should be invariant to changes in priors to avoid false alarms that may lead to unnecessary actions. We believe this requirement is very relevant in many domains. For instance, in the medical diagnosis system mentioned above, diseases are usually well defined, though the relative proportion of ill to healthy patients may change over time; (3) It should be able to handle unlabeled data since we are interested in using it during the classification process. While not required, we would like our detector to exploit the labels of the train data to improve precision; (4) It should be efficient to allow applications in online setups.

The task of feature-shift detection in multivariate data can be seen as a statistical test to determine if train and classification data are drawn from a different set of features. For this purpose we could have used dissimilarity methods that try to compare distributions of two samples; such methods can be found in studies on abrupt change detection [2], transfer learning [3], and virtual concept drift [4]. However, these studies do not distinguish between changes in distribution caused by feature shift, and those that occur in prior distributions.
Moreover, change detection algorithms are mostly applied on unlabeled batches and ignore the labeled information, or when the labels are known in one of the batches. On the other hand, studies on transfer learning and concept drift, which try to improve classification performance, are usually focused on change detection when there are labeled data in both batches.

In this study, we introduce an on-line method for detecting feature shift under the requirements specified above: it has high precision and recall, it is invariant to changes in priors, it uses labeled train data to better predict feature shift in unlabeled data, and it is efficient. Instead of basing our measure on estimating probability density functions, we base it on more robust similarity functions that use one-class SVM for monitoring distributional changes in a high-dimensional feature space.

2 The Feature Shift Detector

Let \( I \) be an instance space, and \( f_1(i), \ldots, f_N(i) \) be a set of functions (features) from \( I \) to \( \mathbb{R}^N \). We denote \( x = (f_1, \ldots, f_N) \) to be a feature vector for each \( i \in I \). Feature shift is defined as a change in one of the functions \( f_1(i), \ldots, f_N(i) \). We assume objects are drawn randomly from the instance space, which includes all classes. Each drawn object is transformed to a feature vector \( x \in \mathbb{R}^N \) in the feature space. We denote \( z = (x, c) \) to be an example, where \( x \in \mathbb{R}^N \) is a feature vector and \( c \in \{c_1, \ldots, c_l\} \) is a class label. Let \( B_l, B_{ul} \) be the training and classification batches of examples i.i.d. with respect to \( P_l \) and \( P_{ul} \) distributions. Our goal is thus to develop a method for detecting feature shift between batches, with a special interest in measuring a distance between the labeled batch \( (B_l) \) and the unlabeled one \( (B_{ul}) \). The distance should be sensitive to feature shift and relatively insensitive to changes in priors. This measure will be used to monitor and maintain the overall stability of the feature extraction procedure.

In this paper we assume the concept remains unchanged. We use the decomposition \( P(x) = \sum_c P(c)P(x|c) \) to distinguish between changes in \( P(c) \) and \( P(x|c) \): (1) Changes in the instance space. Changes in \( P(c) \) (changes in priors) occur in the instance space and represent an actual change in the current world state, as reflected in the class prior probabilities; (2) Changes in the feature space. Changes in the conditional distributions \( P(x|c) \) that occur in the feature space and represent a change in one or more of \( f_1(i), \ldots, f_N(i) \) features.

Given the \( B_l \) and \( B_{ul} \) batches, we have two options for estimating the presence of feature shift: (1) Estimating it in terms of changes in the \( P(x) \) distributions between the two populations. Recall, however, that changes in priors may lead to false estimates; (2) Estimating it in terms of changes in the conditional distributions \( P(x|c_1), \ldots, P(x|c_l) \) between the two populations, and combining them all into a single measure.

We find the latter option more attractive, albeit more challenging. Let \( p_1, \ldots, p_l \) be probability density functions (pdfs) of the classes \( c_1, \ldots, c_l \) from which examples in \( B_l \) were drawn. Let \( S^* \) denote a feature-similarity measure we would like to calculate (small values indicate feature shift). In theory, if we could evaluate \( p_1, \ldots, p_l \) and the labels of \( B_{ul} \) were known, we could define \( S^* \) as the average log-likelihood:

\[
S^* (B_l, B_{ul}) = - \frac{1}{|B_{ul}|} \sum_{x_i \in B_{ul}} \log(p_j (x_i)) \quad (1)
\]

where \( j \) is the index of the class type of example \( x_i \).

Estimating the pdf’s in real high-dimensional problems is, however, impractical. Therefore, we use a less precise notion of distances for estimating \( S^* \) (described in the next section). As a result, a set of \( l \) distances (one for each class) are evaluated for each example in \( B_{ul} \), and converted monotonically to \( s_1, \ldots, s_l \) similarity values. We treat these similarity values as pseudo-log-likelihood measures that estimate whether a given feature vector \( x \) is drawn from \( p_1, \ldots, p_l \). Since the labels of examples in \( B_{ul} \) are unknown, the final similarity value of an example in \( B_{ul} \) is defined as the maximal value among all similarity measures:

\[
S (B_l, B_{ul}) = \frac{1}{|B_{ul}|} \sum_{x_i \in B_{ul}} \max_{j=1,\ldots,l} (s_j (x_i)). \quad (2)
\]

We define the change detection measure \( FSD = 1 - S \) (FSD holds for feature shift detector) as our proposed feature-shift detection algorithm.

3 The Similarity Functions

Let \( X_c \subset \mathbb{R}^N \) be a set of \( q_c \) feature vectors of class \( c \). Given a feature vector \( x \in \mathbb{R}^N \), the goal of the similarity function \( s_c \) is to estimate whether \( x \) belongs to the same distribution as the \( X_c \) population. Our approach to constructing the similarity is geometric and learning based. We use a one-class SVM approach for representing the geometrical structure of the concept \( c \) outlined by \( X_c \). Then, the similarity is efficiently quantified in terms of the distance between the feature vector and the constructed geometrical model in the feature space.

The one-class SVM algorithm is a model-free approach for estimating the support vectors (SVs) of a high-dimensional distribution [5]. Suppose we use a mapping function \( \Phi : \mathbb{R}^N \rightarrow \mathcal{F} \) to map the feature vectors in \( X_c \) to some other space \( \mathcal{F} \) such that each mapped
vector in \( X_i \) lies on a hypersphere. The basic concept is to treat the origin of this hypersphere as the only member of the second class, and to find the best spherical cup bounded by the separating \( h \subset \mathcal{F} \) hyperplane that maximizes the margins between classes. This cup is supposed to capture regions in feature space where the distribution of \( X_i \) lives (its support).

Figure 1: In gray: the spherical cup. In bold: the distance.

The \( s_c \) function uses this spherical cup bounded by \( h \) as the geometrical model of the learned concept. See Figure 1. The gray area represents the geometrical model of the learned concept. The value of the similarity function is related to the distance of the transformed feature vector \( \Phi(x) \) from the spherical cup. A kernel \( K(x_i, x_j) \) is used to evaluate the dot products \( \langle \Phi(x_i), \Phi(x_j) \rangle \) in \( \mathcal{F} \). Given a set \( X_c \) of \( q_c \) feature vectors of class \( c \), we define the distance between a feature vector \( x \) and \( X_c \) as:

\[
d_c (x) = - \sum_{i=1}^{q_c} \alpha_i K(x_i, x) + b \tag{3}
\]

where \( \alpha_1, \ldots, \alpha_{q_c} \) are the SVs’ coefficients. Geometrically, this is the distance between \( \Phi(x) \) and \( h \) in \( \mathcal{F} \) (as shown in bold in Figure 1). This distance is used to specify a normalized similarity function in \([0,1]\): Let \( d_{c,1}, \ldots, d_{c,q_c} \) be the corresponding distances of all feature vectors in a training set \( X_c \). Let \( \mu_c \), \( \sigma^2_c \), \( d_{\text{min}}_c \) denote the mean, variance, and minimal value of these distances. Then, given a feature vector \( x \), the normalized similarity measure is defined as:

\[
s_c(x) = \begin{cases} 
1 & \text{if } d_c (x) \leq d_{\text{min}}_c \\
0 & \text{if } d_c (x) \geq \mu_c + k\sigma_c \\
\frac{\mu_c + k\sigma_c - d_c (x)}{\mu_c + k\sigma_c - d_{\text{min}}_c} & \text{otherwise}
\end{cases} \tag{4}
\]

where \( k \) is a constant. Note that the upper bound on the distance is extended to \( \mu_c + k\sigma_c \), so that more extreme values can be evaluated as well.

\(^1\)We use a Gaussian kernel \( K(x_i, x_j) = e^{-\gamma(x_i-x_j)^2} \).

4 Empirical Evaluation

All experiments use image datasets, for which it is easy to simulate and visualize feature shift problems. Experiments were conducted on 8 image datasets taken from the TILDA databases \(^2\). The TILDA database consists of images of textile defects from four types of textile surfaces, composed of two types of fabric. In total, 8 datasets were used, corresponding to 8 different fabrics. In all datasets, the goal of the defect classification process is to classify each of the test images into one of four defect categories: thread flow (E3), wrinkle (E4), contaminant (E5), and shade (E6). 50 images for each defect type made for a total of 200 images per dataset, randomly split to train and test data of equal size. For each dataset, the following actions were taken: (1) Simulate changes in priors in test images by ordering the images according to the \( C1 \rightarrow C2 \rightarrow C3 \rightarrow C4 \) order. This way, batches of consecutive images are strongly biased to one or two classes. Within each category the examples are shuffled randomly; (2) Create two series of test images, and distort one of them with feature shift (all test images are included in both series at the same order). Feature shift is introduced to the distorted images by de-focusing, simulated by convolving the image with an averaging filter of size 5; (3) Create batches of 25 consecutive images. This process is repeated for all possible batches in both series; (4) Apply feature extraction on the train data and test batches. First, a set of SIFT descriptors is calculated for the SIFT feature points. These descriptors are clustered into a vocabulary of visual keypoints. Then, a spatial histogram is applied to characterize these clustered keypoints in terms of their spatial locations in the image. Finally, this histogram is concatenated into a long feature vector of 1200 features. These methods are efficiently implemented using the VLFEAT library [6].

We use the LibSVM package of the one-class SVM algorithm to implement our method. We use \( k = 6 \) in Eq. 4 for all experiments. For computational reasons, the set of 100 most informative features is used for evaluation (the same 100 features are also used for the reference methods) \(^3\).

The following 5 reference methods were all taken from the family of change-detection methods that compare multivariate distributions without an intermediate density estimation step: (1) Metavariable Wald-Wolfowitz test (WW) [7]; (2) Kernel change detection (KCD) [8]; (3) Maximum mean discrepancy test (MMD) [9]; (4) PAC-Bayesian margin test (PBM) [10]; (5) Error-rate test (ERT) [10].

\(^2\)http://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en

\(^3\)Information-gain feature selection is applied.
4.1 The Precision-Recall Tradeoff

We tested the precision-recall rates of the six methods in detecting feature shift. For each dataset, we apply the six methods on all batches in both series. Recall that in half of the batches there are changes in priors and feature shift, while in the other half only changes in priors. Then, for each method, precision-recall plots are measured over all possible thresholds.

Figure 2: Precision-recall curves for the six detection methods

Figure 2 shows the precision-recall plots averaged over experiments. It can be seen that the FSD method provides the best compromise between precision and recall. For example, in terms of their break even point (BEP) measures – the points at which precision equals recall – FSD outperforms the other 5 reference methods with a BEP of 0.86 while its second best competitor does so with BEP of 0.72.

We hypothesize that the poor performance of the competing methods is due to their inability to differentiate between feature shift and changes in priors. We test this hypothesis by repeating this experiment with batches created from randomly permuted series, thus avoiding changes in priors. Indeed, the performance of the reference methods improved to a level close to that of FSD.

Note that, as opposed to KCD PBM and ERT, our method does not need to be retrained on each test batch. Hence, in the context of an online detection process, FSD is relatively cheap, and estimated in $O(n_{sv} m)$ time where $n_{sv}$ is the number of support vectors in the training data and $m$ is the size of the test batch. In comparison to other methods, FSD is still the least computationally demanding.\footnote{MMD and WW complexities are estimated in $O\left(\left(n+m\right)^2\right)$ time where $n$ is the size of the train data.}

Additional experiments were made to measure the correlation of feature shift measures to changes in results. Excellent correlation were measured between FSD and changes in accuracy, and poor correlations for the other methods. Due to space limitations, we could not dwell on this much.\footnote{The prepared data and detailed results are available in http://www.cs.technion.ac.il/~assafgr/articles/fsd.html.}

5 Conclusions

The efficiency and robustness of our measure over other methods is demonstrated when both feature shift and changes in priors occur. As far as we known, the work described in the paper is the first to formalize the feature shift problem. Our method is especially useful for the quality control community. In practice, defect classification systems must perform very reliable and exhibit anytime stability. Our method for measuring the presence and extent of feature shift can be used to detect instabilities in industrial monitoring processes and increase management system efficiency.

References