Classifier Adaptation at Prediction Time

Amelie Royer
ENS Rennes, France
amelie.royer@ens-rennes.fr

Christoph H. Lampert
IST Austria
chl@ist.ac.at

Abstract

Classifiers for object categorization are usually evaluated by their accuracy on a set of i.i.d. test examples. This provides us with an estimate of the expected error when applying the classifiers to a single new image. In real application, however, classifiers are rarely only used for a single image and then discarded. Instead, they are applied sequentially to many images, and these are typically not i.i.d. samples from a fixed data distribution, but they carry dependencies and their class distribution varies over time.

In this work, we argue that the phenomenon of correlated data at prediction time is not a nuisance, but a blessing in disguise. We describe a probabilistic method for adapting classifiers at prediction time without having to retrain them. We also introduce a framework for creating realistically distributed image sequences, which offers a way to benchmark classifier adaptation methods, such as the one we propose. Experiments on the ILSVRC2010 and ILSVRC2012 datasets show that adapting object classification systems at prediction time can significantly reduce their error rate, even with no additional human feedback.

1. Introduction

Object recognition systems have become efficient and reliable enough to be useful for practical, even commercial, applications. For example, a system that recognizes all 20,000 ImageNet categories could be sold pre-trained to a wide range of customers. Each customer could use the system afterwards for his or her own application, e.g., supporting customers in a supermarket by recognizing products, educating children by identifying zoo animals, or helping robots to navigate safely in an office environment.

These scenarios have in common that the statistical distribution of the images that need to be classified differs from the distribution of images in the dataset used to train the classifier. One difference is a change in class probabilities: for instance, in the robot example, furniture and office equipment will appear more often than their fraction in ImageNet. Most other classes, such as exotic animals, will appear much less often or not at all. A second difference is that the images at prediction time will not be independent of each other. For example, when a customer in a shop takes a photo of a broccoli, chances are high that the next image will also be a vegetable rather than a dairy product.

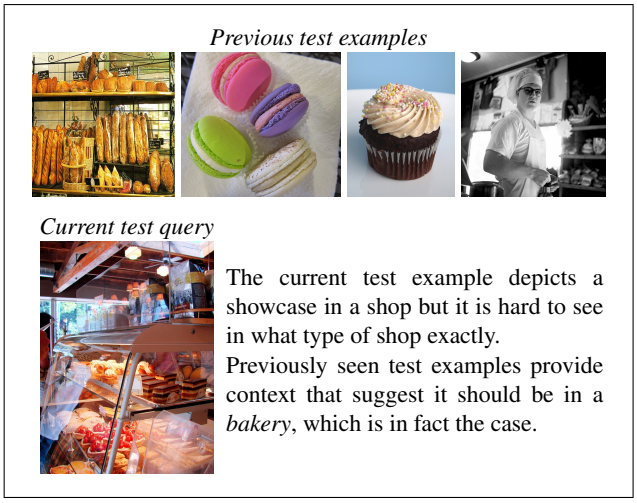

Figure 1. Example of a sequence of test examples with label correlations that help to correctly classify ambiguous images.

Machine learning research generally considers it undesirable if the data distribution changes between training time and prediction time. The phenomenon is called a *domain shift* and is studied in the field of *domain adaptation*. Typically, a domain shift leads to a reduction of classification accuracy, and a significant part of domain adaptation research studies the question of how such a loss in accuracy can be prevented, or at least theoretically quantified.

In this work, we argue in favor of a different view: *at prediction time, non-uniformly sampled and correlated data are actually beneficial*. A non-uniform class distribution has lower entropy than a uniform one, thus we have less uncertainty about which classes to expect and which ones not. A statistical dependence between subsequent images in a sequence means that the past images carry information about the next ones. This can also be used to infer informa-

tive priors about future data. Both effects work in our favor, therefore we should exploit them to *increase the classification accuracy* instead of just trying to prevent a decrease.

Besides the above observation we make two technical contributions. The first contribution is a simple but effective probabilistic method for adapting multi-class classifiers on the fly to *realistic sequences* of test images. The adapted classifiers can achieve a higher overall classification accuracy by avoiding unnecessary mistakes, such as predicting exotic animals classes in an office environment. The second contribution is a framework for benchmarking adaptive classification systems as the one we describe. This is important, because existing benchmarks are designed for the academic setting with identically distributed and independent (i.i.d.) test data. Instead, we propose a set of techniques for generating more realistic sequences of test images, either by sampling from a hidden Markov chain in which the hidden state changes according to a random walk in a semantic space, or in a data-driven way by harvesting the semantic structure of natural language texts.

1.1. Related Work

To our knowledge, no previous work has aimed at generating realistic image sequences for benchmarking adaptive object recognition systems. The problem of classification when the distribution of test examples differs from the distribution of training examples is well known in machine learning research, though. It is studied in the field of *domain adaptation*, see for example [11, 18] for surveys.

Most work in domain adaptation studies the situation when training a classifier, typically under the assumption that test data are available, at least partially, already at training time. This is very different from the setting of classifier adaptation at prediction time that we are interested in. We assume that the classifiers were trained previously, e.g. by a commercial service, and without knowledge of the customer's exact data distribution. Therefore, we are not concerned with questions of changing the learning mechanism or constructing an invariant data representation.

Interestingly, concentrating on the prediction step makes the problem easier for us, not harder. Typically, domain adaptation methods try to prevent a loss in accuracy due to the domain shift. We make use of the change of distribution to actually achieve better classification accuracy. In this aspect, our goal is similar to the objective of *co-classification* [12]. However, our work is different in many other aspects. In particular, co-classification requires all test samples to be available as a batch, while we work in the online classification setting, where samples are classified one-by-one and adaptation happens on the fly.

Little prior work exists on the task of domain adaptation at prediction time itself. One exception in computer vision is [25], where Xu *et al*. use *online transfer learning* [26] to adapt a deformable part model to a new distribution. More related to our work is [10], where Jia and Darrell study a problem similar to ours: they adapt a multi-class image classifier to a specific subset of classes. However, their settings differ in its assumptions: Jia and Darrell assume i.i.d. test example with labels from an unknown subtree of a known taxonomy. In contrast, we do not make any assumptions which classes are present or absent at prediction time, and we also target dependent samples and time-varying data distributions. Adaptation to time-varying data is the topic of [15], in which Levinkov and Fritz describe the adaptation of classifier ensembles at test time and [9], in which Hoffman *et al*. introduce a method for smoothly adapting subspaces to follow a time-varying data distribution. Our work is in fact orthogonal to these: we target distribution changes due to changing class proportions, while the earlier works aim at adapting to changes in appearance of the classes themselves.

2. Classifier Adaptation at Prediction Time

In this section, we formally introduce the problem setting and describe a simple yet effective method to adapt pre-trained classifiers to a new distribution.

2.1. Notation

We work in a multi-class classification setting, where the goal is to assign outputs (class labels), $y \in \mathcal{Y} = \{1, \ldots, K\}$ to inputs (images), $x \in \mathcal{X}$. In contrast to the classical domain adaptation setting, we assume that a pre-trained multi-class classifier is available, $f : \mathcal{X} \to \mathcal{Y}$, which is probabilistic, i.e. $f(x) = \operatorname{argmax}_y f_y(x)$ for functions $f_y : \mathcal{X} \to \mathbb{R}$ that reflect the conditional label probabilities, $P(y|x)$, when $P(x, y)$ is the data distribution at training time.

We are interested in improving the accuracy at prediction time by adapting the predicted class scores, not in changing the training procedure. This setting has the advantage that it is applicable to classifiers of arbitrary parametric form, including e.g., convolutional networks [1], random forests [6] or one-versus-rest support vector machines [14] with Platt scaling [19]. It is sufficient that the functions f_y are available in executable form, and we also do not need access to the original training set. This is important for real world application, where the training set could be too large to keep around for prediction time, or might be commercially valuable and not available to customers. We do, however, assume that we know the class proportions at training time, i.e. a vector $\rho = (\rho_1, \ldots, \rho_K)$, where ρ_y for any $y \in \mathcal{Y}$ denotes which fraction of the training set had label y.

2.2. Prediction task

At prediction time the classifier receives inputs $x_1, x_2, \ldots$ for which it has to predict labels, $y_1, y_2 \ldots$, in an online way, i.e. for any input x_t it outputs a label y_t before

the next input arrives [3]. To reflect many real world scenarios, we do not assume the samples in the sequence to be independent, and we allow their underlying data distribution to be different from the data distribution at training time, or even time-varying. The one assumption we do make is that the distribution does not change arbitrarily, but only due to varying *class priors*. This is a reasonable assumption for object recognition tasks, because modern feature representations are invariant to many nuisance effects, so the overall appearance of a class is rather stable [7]. In contrast, invariant features do not help against the effect of certain classes becoming more or less frequent at prediction time than they had been at training time.

We consider three feedback scenarios: a) *online prediction*, where after the system predicted a label, the correct label for this example, y_t, is revealed to the system, b) *prediction with bandit feedback*, where the only feedback is wether the decision made by the system was correct or not, c) *unsupervised prediction*, where the system is given no feedback about its performance. All three settings occur in real-world situations. Online feedback is common when a computer vision system acts under constant supervision. For example, an intelligent cash register that tries to automatically recognize the products a user wants to buy, but in case of a wrong prediction a human operator stands by to correct the mistakes. Bandit feedback is typical for interactive systems in which a user is asked to provide feedback when a mistake occurs. Users are often willing to do so, because signalling a single bit of information per decision can be done almost effortlessly. Finally, the unsupervised setting is common for automatic systems without an interactive component, e.g. surveillance cameras.

2.3. Classifier adaptation

In this section we introduce our first contribution, a technique for on the fly classifier adaptation. First, we assume that the distribution at prediction time, $Q(x, y)$, is fixed but differs from the distribution at training time, $P(x, y)$, by a change in class proportions from $P(y) = (\rho_1, \dots, \rho_K) = \rho$ to $Q(y) = (\pi_1, \dots, \pi_K) = \pi$. The objects themselves, however, do not change their visual appearance, i.e. $P(x|y) = Q(x|y)$ for all $y \in \mathcal{Y}$. If π were known, we could immediately derive an optimal adaptation rule in closed form, see [23] for a detailed derivation.

Definition 1. Let $f(x) = \operatorname{argmax}_y f_y(x)$, with $f_y : \mathcal{X} \to \mathbb{R}$ for $y \in \mathcal{Y}$, be a probabilistic multiclass classifier that was trained with class proportions ρ. Then we call the classifier $g : \mathcal{X} \to \mathcal{Y}$ given by

$$g(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} g_y(x) \quad \text{for} \quad g_y(x) = \frac{f_y(x)\pi_y}{\rho_y}. \tag{1}$$

the *class-prior adaptation* of f from ρ to π.

Lemma 1. *If $f_y(x) = P(y|x)$, $\rho_y = P(y)$ and $\pi_y = Q(y)$ for all $y \in \mathcal{Y}$, then $g_y(x) \propto Q(y|x)$ and the class-prior adaptation of f from ρ to π is the Bayes-optimal classifier for $Q(x, y)$-distributed data.*

The proof is elementary: inserting the assumptions of Lemma 1 and the assumed relation between P and Q into the definition of $g_y(x)$, we obtain

$$g_y(x) = \frac{P(y|x)Q(y)}{P(y)} = \frac{P(x|y)Q(y)}{P(x)} \tag{2}$$

$$= \frac{Q(x|y)Q(y)}{P(x)} = \frac{Q(y|x)Q(x)}{P(x)} \propto Q(y|x) \tag{3}$$

Note that Lemma 1 also provides us with a formal justification for studying classifier at prediction time: as soon as $\rho \neq \pi$ the original decision rule, f, is not Bayes-optimal anymore, thus it might make more errors than necessary.

2.4. On the fly estimation of the class proportions

In practice, the class proportions, π, are unknown, so we cannot simply compute the class-prior adaptation and predict optimally. Instead, at any time $t = 1, 2, \dots$, we form an estimate of the class proportions, $\pi^{(t)}$, using the available data, and define the correspondingly adapted classifier:

$$g^{(t)}(x) = \operatorname*{argmax}_{y \in \mathcal{Y}} g_y^{(t)}(x) \quad \text{for} \quad g_y^{(t)}(x) = \frac{f_y(x)\pi_y^{(t-1)}}{\rho_y}. \tag{4}$$

where the index $t-1$ of π is due to the fact that when classifying the t-th sample, we only have prior information from the $t-1$ previous examples available.

The problem of estimating a categorical distribution, such as π, from a set of t samples, $y_1, \dots, y_t$ is also a well-studied problem. We adopt a classical Bayesian approach, using a symmetric Dirichlet distribution as prior [16, Chapter 3]. Then the optimal (posterior mean) estimate is

$$\pi_y^{(t)} = \frac{n_t(y) + \alpha}{t + K\alpha}, \quad \text{for } y \in \mathcal{Y}, \tag{5}$$

where $n_t(y)$ is the number of times the label y occurs in the label sequence $y_1, \dots, y_t$, and $\alpha > 0$ is the parameter of the Dirichlet prior. In practice, we use $\alpha = \frac{1}{2}$.

The Bayesian estimate (5) is preferable to the maximum likelihood variant, $\pi_y^{(t)} = n_t(y)/t$, in this context, because it ensures that even unobserved labels receive non-zero probabilities. Probability values of zero should be avoided, since they prevent Equation (4) from predicting the corresponding classes, regardless of the output of the base classifiers f_y.

The above derivation shows that all we need to optimally adapt the classifier scores during prediction is the ability to compute or estimate $n_t(y)$ on the fly. For this we set

$$n_t(y) = \sum_{\tau=1}^{t} \delta_\tau(y), \tag{6}$$

where the definition of the *update vector*, δ_τ, depends on the available feedback. For sequentially arriving data, we compute n_t incrementally,

$$n_t(y) = n_{t-1}(y) + \delta_t(y). \tag{7}$$

which takes time $O(K)$ independent of t.

Online prediction. In the online prediction scenario, when asked to make a prediction for an input x_t, the labels $y_1, \dots y_{t-1}$ are known to the system. Therefore, we can compute the exact n_t by setting

$$\delta_t(y) = [\![y = y_t]\!], \tag{8}$$

where $[\![\cdot]\!]$ are the *Iverson* brackets, i.e. for a logical predicate A one has $[\![A]\!] = 1$, if A is *true*, and $[\![A]\!] = 0$, otherwise.

The law of large numbers guarantees that the resulting estimate $\pi^{(t)}$ converges to the true label proportions, π, and therefore, the estimated classifier $g^{(t)}$ will converge to the optimal g. This holds also for dependent samples under mild conditions on their correlation [8, Chapter VII].

Prediction with bandit feedback. In the bandit setup, the system only receives feedback whether its prediction, $g^{(t)}(x_t)$, was identical to the correct label, y_t, or not. If the answer is *yes*, we know the correct label and we can update n_t by rule (7) with $\delta_t(y) = [\![y = g^{(t)}(x_t)]\!]$.

Otherwise, we only know that the classifier was mistaken, and that one of the remaining $K-1$ labels was correct, but not which one. Thus, we update the parameters of all labels, except the incorrect one, by a uniform fractional amount.

$$\delta_t(y) = \begin{cases} 0, & \text{if } y = g(x_t). \\ \frac{1}{K-1}, & \text{otherwise.} \end{cases} \tag{9}$$

In the situation of binary classifiers, $K = 2$, bandit feedback is as informative as full online feedback, and the above updates are indeed identical to the online rule (8).

Unsupervised prediction. In unsupervised situations there is no external feedback, so we have to trust our own predictions as a proxy for label information. Instead of updating the count of the correct label, we update the count of each label by its expected increment, i.e. the probability of this label being the correct one, according to the current model $Q_t(y|x_t) \propto g_y^{(t)}(x_t)$.

$$\delta_t(y) = \mathbb{E}_{\bar{y} \sim Q_t(\bar{y}|x_t)} \{ [\![y = \bar{y}]\!] \} = \frac{g_y^{(t)}(x_t)}{\sum_{\bar{y}} g_{\bar{y}}^{(t)}(x_t)}. \tag{10}$$

This procedure, in which a system learns from its own predictions, resembles *self-training* in semi-supervised learning [17], or EM-based domain adaptation [23]. The main difference to these techniques lies in the fact that they work with fixed test set over which they iterate multiple times, whereas we work in the streaming setting, where the samples to be classified occur one at a time.

In the unsupervised setting the system receives no feedback by which it could notice and correct its own mistakes. Therefore, there is no guarantee that the estimated $g^{(t)}$ will converge to the optimal classifier. Nevertheless, we observed the procedure to work remarkably well in practice for object categorization tasks, where the underlying state-of-the-art classifiers are rather reliable, see Section 4.

Prediction with a time-varying distribution. When the data distribution at prediction time is not constant but varies over time, one cannot hope for convergence in any of the above settings, since there is no "correct" π to converge to. However, assuming that the distribution changes not too rapidly, we can still form an on the fly estimate of the current class proportions using a sliding-window process: Let L be the window size, then we follow Equation (5) to estimate $\pi^{(t)}(y)$ for the first L steps. Afterwards, we estimate the label counts from the L most recent steps only, i.e.

$$\pi_y^{(t)} = \frac{n_t(y) + \alpha}{L + K\alpha}, \text{ with } n_t(y) = \sum_{\tau=t-L+1}^{t} \delta_t(y), \tag{11}$$

with $\delta_\tau(y)$ given by Equation (8), (9), or (10), depending on the available feedback. We can also again compute n_t incrementally, using the update rule

$$n_t(y) = n_{t-1} + \delta_t(y) - \delta_{t-L}(y), \tag{12}$$

which shows an alternative way to think of this procedure: at every time step, we *learn* from the most recent data, and we *forget* data that occurred too long ago. Consequently, the estimate reflects the recent label distribution, but it remains flexible enough to adapt to changes in the distribution. In practice we use a fixed value of $L = 100$, which is a trade-off between having a window large enough to compute accurate estimates, and yet small enough to adapt to fast distribution variations over time.

3. A Benchmark for Classifier Adaptation

In existing image categorization benchmarks the training and test sets have independent samples with identical data distributions. Since this i.i.d. condition is typically not fulfilled when one just collects images, e.g. from the Internet, it is enforced artificially. For example, for the *ImageNet* challenges, training and test sets are created by randomly selecting a prescribed number of images for each class from a larger annotated corpus [22].

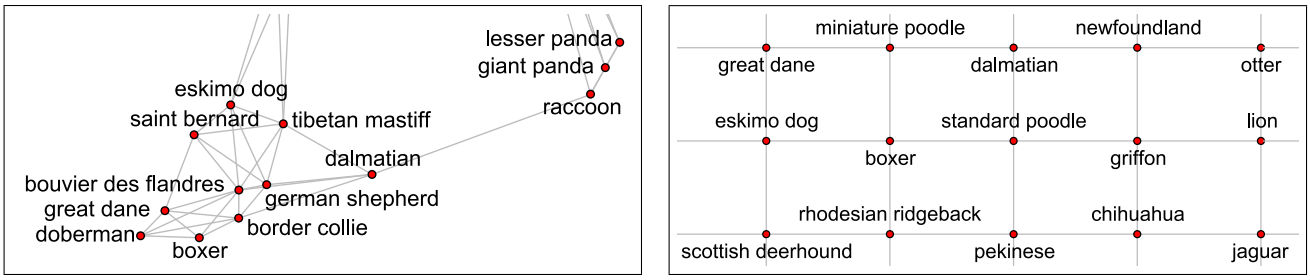

Figure 2. Excerpts of the MDS (left) and KS (right) graphs for the ILSVRC2010 classes with WordNet distance (not all edges are drawn).

> ... when the ***rabbit*** actually took a ***watch*** out of its *waistcoat-pocket* and looked at it and then hurried on, *Alice* started to her *feet*, for it flashed across her *mind* that she had never before seen a ***rabbit*** with either a *waistcoat-pocket*, or a ***watch*** to take out of it, ...

Figure 3. Excerpt from *Alice in Wonderland* by Lewis Carroll with nouns italicized and ILSVRC2010 (super-)classes marked in bold.

Unfortunately, the fact that the natural dependencies have been removed implies that existing datasets cannot directly be used to benchmark the effect of classifier adaptation, as we introduced it in the previous section. The way we overcome this problem forms a second contribution of this work. We describe three methods for creating realistic, non-i.i.d. test sets from an existing object classification dataset, using ImageNet as a running example.

3.1. Generating Realistic Image Sequences

The main idea for re-introducing dependencies into the test data is to change the sampling process by which the test sets are created. Instead of sampling a fixed number of images independently for each class, we create sequences of dependent samples using a two-stage latent variable construction: 1) we create *sequences of class labels*, using one of three techniques described below, 2) for each class label in the sequence, we sample one example image of the respective class out of the image corpus.

Sequences from semantic random walks. For the first two generation methods, we assume that a matrix of pairwise similarities between classes is available. For the ImageNet dataset, we use the *least common ancestor*-distance as induced by the WordNet hierarchy, $d(y, \bar{y}) = \mathrm{height}(\mathrm{lca}(y, \bar{y}))$, where lca denotes the least common ancestor and $\mathrm{height}(\mathrm{z})$ denotes the length of the longest path from the node z to a leaf in the hierarchy [22]. Other semantic distances could also be used, e.g. based on Wikipedia or web searches [21]. We use the distance to create an embedding of all classes into $\mathbb{R}^2$. Afterwards, starting at a random location, we generate a label sequence by a random walk in a k nearest neighbors graph with uniform transition probabilities for staying in the current node or visiting any of its neighbors. The difference between the two methods lies in the way the lower-dimensional embedding is constructed. As we will see, this has a strong impact on the random walk characteristics.

The first method, MDS, relies on *(metric) multidimensional scaling* [5]. It assigns a 2D coordinate tuple to each class such that all pairwise distances are preserved as well as possible. The result is a highly heterogeneous class graph in which groups of similar classes form dense clusters, and large gaps occur between group of classes that are pairwise dissimilar, see Figure 2 (left). Consequently, a random walk on the resulting graph tends to stay for long periods within a cluster of semantically similar classes, even though occasional transitions between clusters occur. A real world analogy to this would be the daily visual experience of a person, which is also partitioned into rather long semantically homogeneous contexts, e.g. first at home, then at work, later at a restaurant or a club, and finally back home.

The KS method is based on *kernelized sorting* [20]. It arranges the classes on a 2D grid based on their pairwise distance to each other. Neighboring classes in the grid are usually semantically similar, but all neighbors have the same distances, so there are no clusters or gaps, see Figure 2 (right). A random walk in the KS setup produces label sequences that can change rapidly between semantically different groups. The effect is comparable, for example, to watching a variety show, in which short segments appear in a loosely related order.

Sequences with context switches. In the sequences described above, the underlying label process changes in a semantically *smooth* way, since the random walk happens amongst neighbors in the projected similarity graph. This is appropriate for static situations but it does not reflect the full breadth of realistic events, which might also exhibit discontinuous changes.

We therefore add the possibility of abrupt topic changes to our models by augmenting the Markov chains with a possibility to jump to a uniformly random label instead of making a step in the k-NN graph. The consequence is that the overall label sequence consists of multiple segments of variable lengths. Each segment has a distribution as produced by the MDS or KS method with random starting point, and the segment lengths are exponentially distributed with pa-

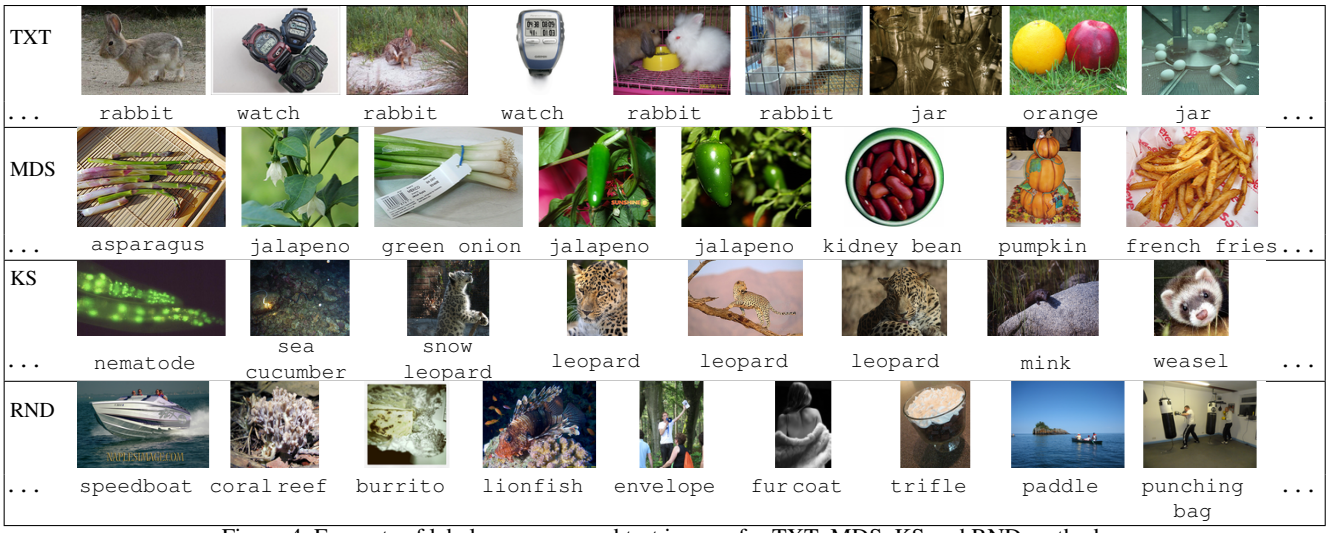

Figure 4. Excerpts of label sequences and test images for TXT, MDS, KS and RND method.

rameter $\frac{1}{\lambda}$, where λ is the probability of starting a new segment at any step. We denote the mechanism of generating sequences in this way by MDS(λ) and KS(λ).

Sequences from natural language. All generative processes described above are useful for benchmarking adaptive classifiers, since they can produce sequences of arbitrary length and with controllable variability. However, the sequences they produce are still artificial in the sense that their underlying random walk is only a weak reflection of a truly natural scenario, such as a person interacting with his or her environment. Therefore we propose a third technique, TXT, based on actual human experience. Its underlying idea is to use natural texts as written representations of realistic sequences of concepts in a natural order.

We assume that a corpus of well-formed texts is available. From each text we first discard all words that are not nouns. This suppresses unwanted homographs (such as the verb *saw* mistaken for the object *a saw*) without losing any relevant words, since all object categories in our classification tasks grammatically are nouns. We then scan the noun sequence and keep those words that correspond to one of the object categories in our task. Because for fine-grained classification tasks the exact category names might never occur in natural text, we also search for super-categories of the target classes according to the WordNet hierarchy. Figure 3 shows an example of the process. When we encounter such a super-class (e.g. *dog*), we randomly sample one of its leaf descendant, which is a target class (e.g. *tibetan mastiff*).

Benchmark dataset. We used the above procedures to compile two new datasets. They differ in the underlying class structure and image sources, which we take from either the ILSVRC2010[1] or the ILSVRC2012[2] database.[3]

For each dataset we create a total of 1000 test sequences with different characteristics. We create 100 TXT-sequences based on the 100 most popular English books from *Project Gutenberg*[4] at that time (a list is given in the supplemental material) using the NLTK toolkit [2]. The sequences vary in length between 400 and 20000 elements. The average length is 3135 for ILSVRC2010 and 3475 for ILSVRC2012. For MDS and KS each, we create 100 sequences of length 3000 using an 8-nearest neighbor graph. We also generate 100 sequences each of MDS(λ) and KS(λ) for each value $\lambda \in \{0.001, 0.01, 0.1\}$. Finally, we also create 100 sequences in which the labels are sampled uniformly at random. We refer to these sequences as RND. They serve as a test set to analyze what happens if the assumption of "realistic" sequences is violated.

Figure 4 illustrates exemplary segments of the resulting label sequences. While they might not look truly natural to a human reader, TXT, MDS and KS do indeed show many typical characteristics that are not present in RND, such as short-term repetitions of labels and the fact that labels are often semantically related if they occur within a short time of each other. For the TXT sequences, the alternation between *watch* and *rabbit* reflects that in the underlying English text, like in real life, two semantically unrelated objects might nevertheless co-occur and interact.

[1] http://www.image-net.org/challenges/LSVRC/2010/
[2] http://www.image-net.org/challenges/LSVRC/2012/
[3] We use the *validation* part of the dataset for this purpose, since annotation for the ILSVRC 2012 test data is not publicly available.
[4] http://www.gutenberg.org

4. Experiments

We demonstrate the potential of adaptive classification by reporting on experiments in a variety of situations. For different base classifiers and adaptation strategies we study if, and how much, adapting the classifier improves the classification error rate over the standard, non-adaptive, setting.

Base Classifiers. Clearly, whether adapting a classifier will be successful or not depends on the underlying base classifier. To avoid a bias due to this choice, we report results for two different base classifiers: CNN is a *convolutional network classifier* based on the *CCV* library.[5] We downloaded pre-trained models that achieve close to state-of-the-art performance for ILSVRC2010 and ILSVRC2012 from the libCCV website. SVM is a *support vector machine* classifier with subsequent Platt scaling [19] to produce probabilistic outputs. It was trained on 4096 dimensional Fisher vectors [24] using the *JSGD* toolkit.[6]

These classifiers have different characteristics, but they both reflect the recent state-of-the-art in object categorization. CNNs are based on the *deep learning* paradigm that learns a feature representation and classifier jointly [1]. They achieve currently the best results on the ImageNet datasets [13]. SVMs are based on statistical learning theory and the *maximum margin* principle [14]. In combination with Fisher vector representations they achieve best results amongst those methods that follow the classical computer vision pipeline in which feature extraction and classifier learning are separate steps [4].

Feedback and Adaptation. We study three feedback scenarios (*online*, *bandit* and *unsupervised*) and compare the classifiers without adaptation against the proposed classifier adaptation strategy (*adaptive*) and its windowed variant (*dynamic*). For all scenarios we measure the classification quality by the top-5 error rate, as it is the standard in the ImageNet challenges.

4.1. Results

Table 1 reports average classification error and standard deviations for both datasets and both choices of base classifiers in the situation with online feedback. It shows that for all situations with realistic sequences, it is better to rely on an adaptive classifier than on the static base classifiers. This is particularly visible for the MDS sequence, for which the adaptation is able to cut the top-5 error by more than half (e.g. on ILSVRC2012, the CNN error rate is reduced from 16.1% to 5.2%, and the SVM error rate from 52.8% to 21.0%). Note that the test images come from the original ImageNet challenge datasets and the task is still to predict one of the 1000 possible classes per image. Therefore, the absolute error rates we report are –to some extent– comparable to the challenge results.

The table shows that for TXT and MDS without context switches, the static adaptation has an advantage over the dynamic strategy. However, the difference between the two methods is relatively small compared to the difference not using adaptation at all. For the rapidly changing KS and KS(λ) sequences, as well as for MDS(λ) when the probability of context switches is high, the dynamic strategy achieves clearly better results than the static one. In fact, for the most challenging sequences, plain multinomial adaptation can actually reduce the accuracy compared to not adapting, whereas the dynamic variant always improves over the non-adaptive baseline. Therefore, we recommend to use dynamic adaptation unless it is known a priori that the samples at prediction will have a benign distribution.

For the RND sequences there is no structure in the test examples to adapt to. In these cases, the non-adaptive classifier is in fact optimal and achieves the lowest error rates. However, the disadvantage of the adaptive methods is not very large, either, especially for the dynamic variant.

Table 2 reports the analogue results for the situations with bandit feedback and without feedback (unsupervised) for the CCV classifier and ILSVRC2012 dataset. Further tables can be found in the supplemental material. The results show a similar trend as above: adaptation always leads to reduced error rates, except for the fully random RND sequences. It is remarkable that this holds even for the situation when no feedback is available at prediction time, i.e. the classifier adapts itself only based on its own confidences. We attribute this to the fact that the base classifiers we rely on are indeed more often right than wrong, so their output –even if not perfect– can be relied on for adaptation.

Overall, our experiments make a strong case in favor of classifier adaptation at prediction time. In particular, the dynamic strategy appears as a promising compromise. It offers a significant reduction of error rate at almost no computational cost, if the label distribution differs between training and prediction time, or even varies over time. If, however, at some time the data at prediction time is in fact completely random, the loss of accuracy is small enough to be tolerable in most practical situations.

5. Conclusion

In this work we called attention to the possibility of improving the accuracy of real world object classifiers by adapting them on the fly to the –potentially time-varying– data distribution at prediction time. We introduced an adaptation strategy for the situation when the label distributions at test time differs from the one at training time, as it is common for real world categorization tasks.

Our experiments showed that for non-i.i.d. test data adaptation can significantly reduce the error rates even for

[5] http://libccv.org/doc/doc-convnet
[6] http://lear.inrialpes.fr/src/jsgd/

Table 1. *Top-5* classification error rates without classifier adaptation (CNN or SVM), with regular adaptation (CNN+adapt, SVM+adapt) or dynamic adaptation (CNN+dyn, SVM+dyn) in the situation with online feedback. Bold entries mark the lowest entries in each setting if the difference is 10^{-3}-significant according to a Wilcoxon signed rank test. See Section 4.1 for a discussion of the results.

ILSVRC2010	CNN	CNN+adapt	CNN+dyn
TXT	14.3 ± 1.5	**9.4 ± 1.6**	10.5 ± 1.5
MDS	13.9 ± 5.0	**6.6 ± 5.0**	7.9 ± 4.9
MDS(0.001)	14.6 ± 3.8	**8.0 ± 4.1**	8.3 ± 3.8
MDS(0.01)	14.2 ± 1.5	11.0 ± 1.5	**9.3 ± 0.7**
MDS(0.1)	14.4 ± 0.8	14.4 ± 0.7	**12.6 ± 0.7**
KS	14.6 ± 1.5	13.7 ± 1.4	**10.9 ± 1.1**
KS(0.001)	14.5 ± 1.4	13.7 ± 1.4	**10.9 ± 1.1**
KS(0.01)	14.4 ± 1.1	13.8 ± 1.1	**11.0 ± 0.9**
KS(0.1)	14.4 ± 0.7	15.0 ± 0.8	**12.2 ± 0.6**
RND	**14.3 ± 0.7**	16.1 ± 0.7	14.9 ± 0.7

ILSVRC2012	CNN	CNN+adapt	CNN+dyn
TXT	19.8 ± 1.9	**12.8 ± 1.9**	14.5 ± 1.7
MDS	16.1 ± 6.5	**5.2 ± 3.0**	6.6 ± 2.6
MDS(0.001)	15.6 ± 4.5	**6.8 ± 2.6**	7.0 ± 1.9
MDS(0.01)	15.7 ± 1.8	11.6 ± 1.5	**9.1 ± 1.1**
MDS(0.1)	16.2 ± 0.8	16.4 ± 0.8	**14.0 ± 0.7**
KS	16.4 ± 1.8	15.2 ± 1.7	**11.8 ± 1.3**
KS(0.001)	16.5 ± 1.8	15.4 ± 1.9	**11.8 ± 1.2**
KS(0.01)	16.4 ± 1.4	15.8 ± 1.4	**12.0 ± 1.0**
KS(0.1)	16.5 ± 0.8	17.1 ± 0.9	**13.8 ± 0.8**
RND	**16.5 ± 0.6**	18.7 ± 0.7	17.2 ± 0.6

ILSVRC2010	SVM	SVM+adapt	SVM+dyn
TXT	39.6 ± 2.4	**29.2 ± 3.7**	31.0 ± 3.5
MDS	43.2 ± 9.4	**18.4 ± 9.1**	22.8 ± 7.5
MDS(0.001)	43.5 ± 7.4	25.0 ± 7.3	**24.5 ± 6.1**
MDS(0.01)	44.3 ± 2.5	36.8 ± 3.0	**29.4 ± 2.5**
MDS(0.1)	44.5 ± 1.1	44.5 ± 1.2	**40.2 ± 1.1**
KS	44.3 ± 1.9	42.3 ± 1.8	**34.9 ± 1.8**
KS(0.001)	44.2 ± 2.0	42.5 ± 2.1	**35.0 ± 1.7**
KS(0.01)	44.0 ± 1.4	43.0 ± 1.5	**35.6 ± 1.3**
KS(0.1)	44.2 ± 1.2	45.1 ± 1.1	**39.2 ± 1.1**
RND	**44.1 ± 0.8**	47.3 ± 0.9	45.5 ± 0.9

ILSVRC2012	SVM	SVM+adapt	SVM+dyn
TXT	50.6 ± 2.9	**38.0 ± 4.4**	40.9 ± 4.2
MDS	52.8 ± 9.6	**21.0 ± 13.4**	26.3 ± 10.8
MDS(0.001)	52.4 ± 6.0	30.5 ± 10.1	**28.7 ± 8.0**
MDS(0.01)	52.8 ± 2.6	44.3 ± 3.2	**36.1 ± 2.9**
MDS(0.1)	53.0 ± 1.1	53.3 ± 1.2	**47.8 ± 1.1**
KS	53.1 ± 1.9	50.9 ± 1.9	**41.1 ± 1.6**
KS(0.001)	53.1 ± 1.6	50.8 ± 1.8	**41.1 ± 1.5**
KS(0.01)	52.9 ± 1.6	52.1 ± 1.7	**42.1 ± 1.5**
KS(0.1)	52.9 ± 1.0	54.4 ± 1.1	**46.7 ± 1.0**
RND	**52.9 ± 0.8**	57.1 ± 0.8	54.5 ± 0.8

Table 2. *Top-5* classification error rates without classifier adaptation (CNN), with regular adaptation (CNN+adapt) or dynamic adaptation (CNN+dyn) in the situation with bandit feedback (left) or no feedback (right).

(a) Bandit Feedback

ILSVRC2012	CNN	CNN+adapt	CNN+dyn
TXT	19.8 ± 1.9	**14.1 ± 1.9**	16.4 ± 1.7
MDS	16.1 ± 6.5	**6.1 ± 3.4**	8.3 ± 3.3
MDS(0.001)	15.6 ± 4.5	**8.2 ± 3.2**	9.0 ± 2.6
MDS(0.01)	15.7 ± 1.8	13.0 ± 1.7	**11.1 ± 1.4**
MDS(0.1)	16.2 ± 0.8	16.9 ± 0.8	**15.1 ± 0.8**
KS	16.4 ± 1.8	16.3 ± 1.8	**13.8 ± 1.6**
KS(0.001)	16.5 ± 1.8	16.5 ± 1.9	**13.9 ± 1.6**
KS(0.01)	16.4 ± 1.4	16.6 ± 1.5	**14.1 ± 1.3**
KS(0.1)	16.5 ± 0.8	17.5 ± 0.9	**15.2 ± 0.8**
RND	**16.5 ± 0.6**	18.5 ± 0.7	16.8 ± 0.6

(b) No Feedback (Unsupervised)

ILSVRC2012	CNN	CNN+adapt	CNN+dyn
TXT	19.8 ± 1.9	**14.8 ± 1.8**	16.4 ± 1.7
MDS	16.1 ± 6.5	**6.8 ± 3.6**	8.1 ± 2.9
MDS(0.001)	15.6 ± 4.5	**9.0 ± 3.4**	9.4 ± 2.7
MDS(0.01)	15.7 ± 1.8	13.6 ± 1.7	**11.3 ± 1.5**
MDS(0.1)	16.2 ± 0.8	16.8 ± 0.8	**14.9 ± 0.8**
KS	16.4 ± 1.8	16.5 ± 1.7	**14.2 ± 1.5**
KS(0.001)	16.5 ± 1.8	16.6 ± 1.9	**14.1 ± 1.5**
KS(0.01)	16.4 ± 1.4	16.7 ± 1.5	**14.1 ± 1.1**
KS(0.1)	16.5 ± 0.8	17.4 ± 0.8	**15.2 ± 0.8**
RND	**16.5 ± 0.6**	18.0 ± 0.7	16.9 ± 0.6

state-of-the-art classifiers, such as convolutional networks. As a second contribution, we introduced three techniques for creating test sets that better reflect real world classification problems by including label correlations and context switches. The source code as well as the test sets we created are available at the first author's homepage[7] to provide a reproducible setting for evaluating adaptation strategies.

In future work, we plan to study adaptation methods beyond probabilistic classification as well as more powerful adaptation strategies, e.g., taking into account label correlations. We also plan to generalize the proposed setting to the situation where a change in class priors and a domain shift in the image domain are handled simultaneously.

[7] http://www.ist.ac.at/~aroyer

Acknowledgements

This work was funded by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 308036.

References

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. 2, 7

[2] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, 2009. 6

[3] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge, 2006. 3

[4] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference (BMVC)*, 2011. 7

[5] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994. 5

[6] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227, 2012. 2

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learing (ICML)*, 2014. 3

[8] W. Feller. *An introduction to probability theory and its applications. Volume 2*. Wiley, New York, 1966. 4

[9] J. Hoffman, T. Darrell, and K. Saenko. Continuous manifold based adaptation for evolving visual domains. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[10] Y. Jia and T. Darrell. Latent task adaptation with large-scale hierarchies. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[11] J. Jiang. A literature survey on domain adaptation of statistical classifiers. http://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey/, 2008. 2

[12] S. Khamis and C. H. Lampert. CoConut: Co-classification with output space regularization. In *British Machine Vision Conference (BMVC)*, 2014. 2

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012. 7

[14] C. H. Lampert. Kernel methods in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 4(3):193–285, 2009. 2, 7

[15] E. Levinkov and M. Fritz. Sequential Bayesian model update under structured scene prior for semantic road scenes labeling. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 2

[16] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. Cambridge, 2012. 3

[17] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000. 4

[18] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. 2

[19] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Cambridge, 1999. 2, 7

[20] N. Quadrianto, S. Le, and A. J. Smola. Kernelized sorting. In *Conference on Neural Information Processing Systems (NIPS)*, 2009. 5

[21] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele. What helps where–and why? Semantic relatedness for knowledge transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 5

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575 [cs.CV], 2014. 4, 5

[23] M. Saerens, P. Latinne, and C. Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural Computation*, 14(1):21–41, 2002. 3, 4

[24] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *International Journal of Computer Vision (IJCV)*, 105(3):222–245, 2013. 7

[25] J. Xu, S. Ramos, D. Vázquez, and A. M. López. Incremental domain adaptation of deformable part-based models. In *British Machine Vision Conference (BMVC)*, 2014. 2

[26] P. Zhao and S. C. Hoi. OTL: A framework of online transfer learning. In *International Conference on Machine Learing (ICML)*, 2010. 2