Published at the 23rd IEEE Conference on Computer Vision and Pattern Recognition, CVPR, June 13-18, San Francisco, CA, USA, 2010

P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints

Zdenek Kalal
University of Surrey
Guildford, UK
z.kalal@surrey.ac.uk

Jiri Matas
Czech Technical University
Prague, Czech Republic
matas@cmp.felk.cvut.cz

Krystian Mikolajczyk
University of Surrey
Guildford, UK
k.mikolajczyk@surrey.ac.uk

Abstract

This paper shows that the performance of a binary classifier can be significantly improved by the processing of structured unlabeled data, i.e. data are structured if knowing the label of one example restricts the labeling of the others. We propose a novel paradigm for training a binary classifier from labeled and unlabeled examples that we call P-N learning. The learning process is guided by positive (P) and negative (N) constraints which restrict the labeling of the unlabeled set. P-N learning evaluates the classifier on the unlabeled data, identifies examples that have been classified in contradiction with structural constraints and augments the training set with the corrected samples in an iterative process. We propose a theory that formulates the conditions under which P-N learning guarantees improvement of the initial classifier and validate it on synthetic and real data. P-N learning is applied to the problem of on-line learning of object detector during tracking. We show that an accurate object detector can be learned from a single example and an unlabeled video sequence where the object may occur. The algorithm is compared with related approaches and state-of-the-art is achieved on a variety of objects (faces, pedestrians, cars, motorbikes and animals).

1. Introduction

Recently, there has been a significant interest in semi-supervised learning, i.e. exploiting both labeled and unlabeled data in classifier training [6, 24]. It has been shown that for certain classes of problems, the unlabeled data can dramatically improve the classifier. Most general learning algorithms [17, 4] assume that the unlabeled examples are independent. Therefore, such algorithms do not enable to exploit dependencies between unlabeled examples which might represent a substantial amount of information.

In computer vision, the data are rarely independent since their labeling is related due to spatio-temporal dependencies. The data with dependent labels will be called *structured*. For instance, in object detection, the task is to label all possible image patches of an input image either as positive (object) or as negative (background). A unique object can occupy at most one location in the input image. In a video, the object location defines a trajectory, which is illustrated in Fig. 2. The trajectory represents a structure of the labeling of the video sequence. All patches close to the trajectory share the same positive label, patches far from the trajectory are negative. Another examples of structured data are in the detection of object parts or in the multi-class recognition of objects in a scene. In these cases, the labeling of the whole image can be constrained by predefined or learned spatial configuration of parts/objects.

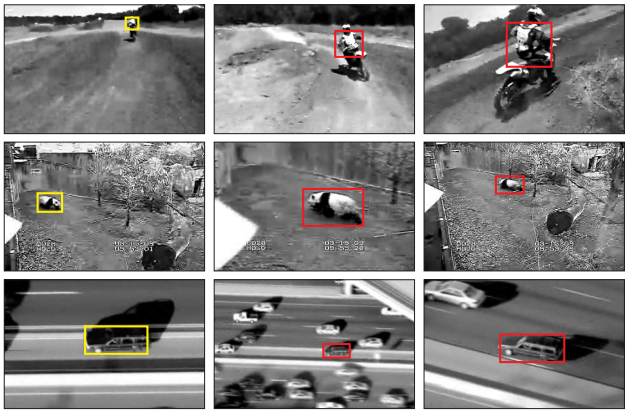

Figure 1. Using a single example (YELLOW), P-N learning builds an object detector from video. The detector localizes the object in significantly different poses (RED).

This paper proposes a new paradigm for learning from structured unlabeled data. The structure in the data is exploited by so called positive and negative structural constraints, which enforce certain labeling of the unlabeled set. Positive constraints specify the acceptable patterns of positive labels, i.e. patches close to the object trajectory are positive. Negative constraints specify acceptable patterns of negative labels, i.e. surrounding of the trajectory is negative. These constrains are used in parallel and we show that their combination enables mutual compensation of their er-

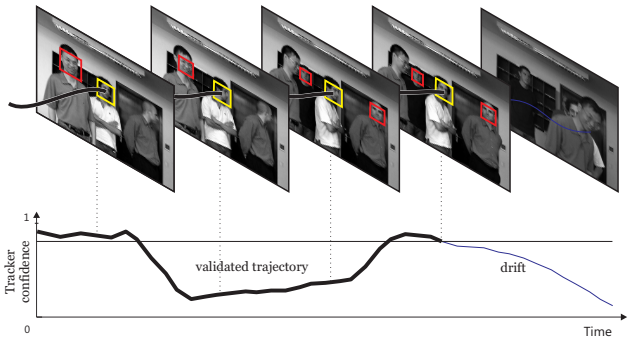

Figure 2. A trajectory of an object represents a structure in video. Patches close to the trajectory are positive (YELLOW), patches far from the trajectory are negative, only the difficult negative examples are shown (RED). In training, the real trajectory is unknown, but parts of it are discovered by a validated adaptive tracker.

rors. These constraints operate on the *whole* unlabeled set and therefore exploit different source of information than the classifier which operates on *single* examples only.

The availability of a small number of labeled examples and a large number of structured unlabeled examples suggest the following learning strategy: (i) Using the labeled samples, train an initial classifier and adjust the predefined constraints with the labeled data. (ii) Label the unlabeled data by the classifier and identify examples which have been labeled in contradiction with the structural constraints. (iii) Correct their labels, add to the training set and retrain the classifier. We call this bootstrapping process P-N learning.

The contribution of this paper is a formalization of the P-N learning paradigm for off-line and on-line learning problems. We provide a theoretical analysis of the process and specify conditions under which the learning guarantees improvement of the initial classifier. In the experimental section, we apply the P-N learning to training an object detector during tracking. We propose simple yet powerful constraints that restrict the labeling of patches extracted from video. Our learning method is processing the video sequence in real-time and the resulting detector achieves state-of-the-art performance.

The rest of the paper is organized as follows. Sec. 2 reviews the approaches for exploiting unlabeled data. Sec. 3 then formulates the P-N learning and discusses its convergence. The theory is then applied to learning an object detector from video in Sec. 4 and validated on synthetic data in Sec. 5. The system will be compared with state-of-the-art approaches and analyzed in detail in Sec. 6. The paper finishes with conclusions and future work.

2. Exploiting unlabeled data

In semi-supervised learning, the processing of unlabeled data is guided by some supervisory information [6]. This information often takes a form of *labels* associated with some of the examples. Another setting is to provide *constraints* [1] such as "these unlabeled points have the same label". This form of supervision is more general and enables to express more complex relationships of the data.

Expectation-Maximization (EM) is an algorithm for finding parameters of probabilistic models. In classification, these parameters may correspond to class labels. EM maximizes the data likelihood and therefore works well for classification problems if the distribution of unlabeled examples is low on boundary between classes. This is often called low density separation assumption [6]. EM was successfully applied to document classification [17] and learning of object categories [8]. EM is sometimes interpreted as a "soft" version of self-learning [24].

Self-learning is probably the oldest approach for semi-supervised learning [6]. It starts by training an initial classifier from a labeled training set, the classifier is then evaluated on the unlabeled data. The most confident examples are added along with the estimated labels to the training set and the classifier is retrained, this is an iterative process. The self-learning has been applied to human eye detection [20]. It was observed that the detector improved more if the unlabeled data were selected by an independent measure rather than the classifier confidence. This suggests that the low density separation assumption is not satisfied for object detection and other approaches may work better.

In *co-training* [4], the feature vector describing the examples is split into two parts, also called views. The training is initialized by training a separate classifier on each view. Both classifiers are then evaluated on unlabeled data. The confidently labeled samples from the first classifier are used to augment the training set of the second classifier and vice versa in an iterative process. The underlying assumption of co-training is that the two views are statistically independent. This assumption is satisfied in problems with two modalities, e.g. text classification [4] (text and hyperlinks) or biometric recognition systems [19] (appearance and voice). In visual object detection, co-training has been applied to car detection in surveillance [14] or moving object recognition [10]. We argue that co-training is not a good choice for object detections, since the examples (image patches) are sampled from a single modality. Features extracted from a single modality may be dependent and therefore violate the assumptions of co-training. Another disadvantage of co-training is that it can not exploit the data structure as each example is considered to be independent.

Learning that exploits the structure of the data is related to adaptive object *tracking*, i.e. estimating object location in frame-by-frame fashion and adaptation of the object model. The object to be tracked can be viewed as a single labeled example and the video as unlabeled data. Many authors perform self-learning [9, 2, 7, 15]. This approach predicts the

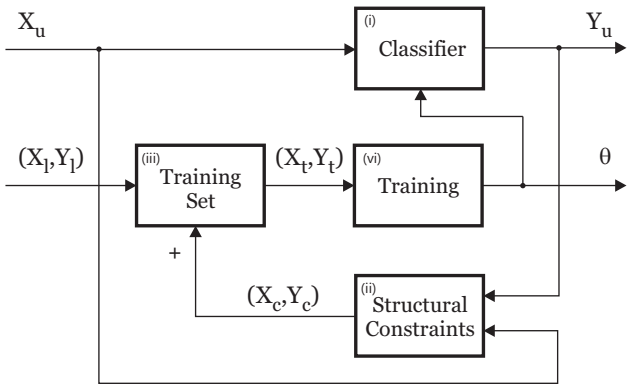

Figure 3. P-N learning first trains a classifier from labeled data and then iterates over: (i) labeling of the unlabeled data by the classifier, (ii) identification and relabeling of examples which labels violate the structural constraints, (iii) extension of the training set, (iv) retraining of the classifier.

position of the object with a tracker and updates the model with positive examples that are close and negative examples that are far from the current position. The strategy is able to adapt the tracker to new appearances and background, but breaks down as soon as the tracker makes a mistake. This problem is addressed in [22] by co-training a generative and discriminative classifiers in the context of tracking. The tracking algorithm demonstrated re-detection capability and scored well in comparison with self-learned trackers on challenging video sequences. Another example is MIL learning [3], where the training examples are delivered by spatially related units, rather than independent training examples. In [12], a tracking algorithm was proposed that combines adaptive tracking with object detections. The learning is based on so called growing and pruning events. The approach demonstrated robust tracking performance in challenging conditions and partially motivated our research.

3. P-N Learning

This section formalizes the off-line version of P-N learning and later generalizes it to on-line learning problem. Let x be an example from a feature-space $\mathcal{X}$ and y be a label from a label-space $\mathcal{Y} = \{-1, 1\}$. A set of examples X and corresponding set of labels Y will be denoted as (X, Y) and called a labeled set. The task of P-N learning is to learn a classifier $f : \mathcal{X} \to \mathcal{Y}$ from a priori labeled set (X_l, Y_l) and bootstrap its performance by unlabeled data X_u. The illustration of the P-N learning approach discussed in this section is given in Fig. 3.

3.1. Classifier bootstrapping

The classifier f is a function from a family $\mathcal{F}$ parametrized by θ. Similarly to supervised setting, P-N learning of the classifier corresponds to estimation θ from training set (X_t, Y_t) with one exception: the training set is iteratively augmented by examples extracted by the constraints from the unlabeled data. The training process is initialized by inserting the a priori labeled examples to the training set, and by estimation of the initial classifier parameters θ^0. The process then proceeds iteratively. In iteration k, the classifier trained in $k-1$ assigns labels to the unlabeled examples, $y_u^k = f(x_u|\theta^{k-1})$ for all $x_u \in X_u$. Notice that the classifier operates on one example at a time only. The constraints are then used to verify if the labels assigned by the classifier are in line with the assumptions made about the data. The example labels that violate the constraints are corrected and added to the training set. The iteration is finished by retraining the classifier with the updated training set. This procedure iterates until convergence or other stopping criterion. Throughout the training, any example can be selected by the constraints multiple times and therefore can be represented in the training set repeatedly even with a different label.

3.2. Constraints

A constraint can be any function that accepts a *set* of examples with labels given by the classifier (X_u, Y_u^k) and outputs a subset of examples with changed labels (X_c^k, Y_c^k). P-N learning enables to use an arbitrary number of such constraints. Two categories of constraints are distinguished that we term P and N. *P-constraints* are used to identify examples that have been labeled negative by the classifier but the constraints require a positive label. In iteration k, P-constraints add $n^+(k)$ examples to the training set with labels changed to positive. These constraints extend the pool of positive training examples and thus improve the generalization properties of the classifier. *N-constraints* are used to identify examples that have been classified as positive but the constraints require negative label. In iteration k, the N-constraints insert $n^-(k)$ negative examples to the training set. These constraints extend the pool of negative training examples and thus improve its discriminative properties of the classifier.

The impact of the constraints on the classifier quality will be now analyzed analytically. Suppose a classifier that assigns random labels to the unlabeled set and then corrects its assignment according to the output of the constraints. The constraints correct the classifier, but in practice also introduce errors by incorrectly relabeling some examples. In iteration k, the error of a classifier is characterized by a number of false positives $\alpha(k)$ and a number of false negatives $\beta(k)$. Let $n_c^+(k)$ be the number of examples for which the label was correctly changed to positive in iteration k by P-constraint. $n_f^+(k)$ is then the number of examples for which the label was incorrectly changed to positive in iteration k. Thus, P-constraints change $n^+(k) = n_c^+(k) + n_f^+(k)$ ex-

amples to positive. In a similar way N-constraints change $n^-(k) = n_c^-(k) + n_f^-(k)$ examples to negative, where $n_c^-(k)$ are correct and $n_f^-(k)$ false assignments. The errors of the classifier thus become:

$$\alpha(k+1) = \alpha(k) - n_c^-(k) + n_f^+(k) \quad (1a)$$

$$\beta(k+1) = \beta(k) - n_c^+(k) + n_f^-(k). \quad (1b)$$

Equation 1a shows that false positives $\alpha(k)$ decrease if $n_c^-(k) > n_f^+(k)$, i.e. number of examples that were correctly relabeled to negative is higher than the number of examples that were incorrectly relabeled to positive. Similarly, the false negatives $\beta(k)$ decrease if $n_c^+(k) > n_f^-(k)$. In order to analyze the convergence, a model needs to be defined that relates the quality of P-N constraints to $n_c^+(k), n_f^+(k), n_c^-(k)$ and $n_f^-(k)$.

The quality of the constraints is characterized by four measures. P-precision is the number of correct positive examples divided by total number of samples output by P-constraints, $P^+ = n_c^+/(n_c^+ + n_f^+)$. P-recall is the number of correct positive examples divided by number of false negatives, $R^+ = n_c^+/\beta$. N-precision is the number of correct negative examples divided by number of all examples output by N-constraints, $P^- = n_c^-/(n_c^- + n_f^-)$. N-recall is the number of correct negative examples divided by total number of false positives, $R^- = n_c^-/\alpha$. We assume here that the constraints are characterized by fixed measures throughout the training, and therefore the time index was dropped from the notation.

The number of correct and incorrect examples at iteration k are then expressed as follows:

$$n_c^+(k) = R^+\,\beta(k), \quad n_f^+(k) = \frac{(1-P^+)}{P^+} R^+\,\beta(k) \quad (2a)$$

$$n_c^-(k) = R^-\,\alpha(k), \quad n_f^-(k) = \frac{(1-P^-)}{P^-} R^-\,\alpha(k). \quad (2b)$$

By combining the equation 1a, 1b, 2a and 2b we obtain $\alpha(k+1) = (1-R^-)\,\alpha(k) + \frac{(1-P^+)}{P^+} R^+\,\beta(k)$ and $\beta(k+1) = \frac{(1-P^-)}{P^-} R^-\,\alpha(k) + (1-R^+)\,\beta(k)$. After defining the state vector $\vec{x}(k) = \begin{bmatrix}\alpha(k) & \beta(k)\end{bmatrix}^{\mathrm{T}}$ and the transition matrix as

$$\mathbf{M} = \begin{bmatrix} 1-R^- & \frac{(1-P^+)}{P^+} R^+ \\ \frac{(1-P^-)}{P^-} R^- & (1-R^+) \end{bmatrix}, \quad (3)$$

it is possible to rewrite the equations as $\vec{x}(k+1) = \mathbf{M}\vec{x}(k)$. These are recursive equations that correspond to a discrete dynamical system [23].

Based on the well founded theory of dynamical systems, the state vector $\vec{x}$ converges to zero if eigenvalues λ_1, λ_2 of the transition matrix M are smaller than one. Constraints that satisfy this conditions will be called *error-canceling*. Fig. 4 illustrates the evolution of error of the classifier when $\lambda_1 = 0$ and (i) $\lambda_2 < 1$, (ii) $\lambda_2 = 1$, (iii) $\lambda_2 > 1$.

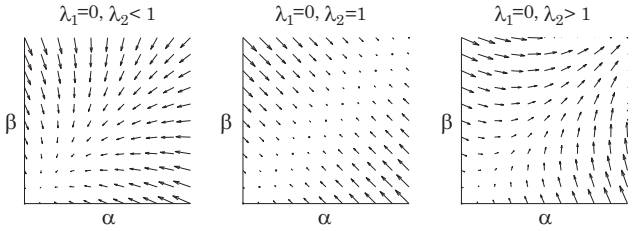

Figure 4. The evolution of errors of the classifier depends on the quality of the structural constraints, which is defined in terms of eigenvalues of matrix $\mathbf{M}$. The errors converge to zero (LEFT), are at the edge of stability (MIDDLE) or are growing (RIGHT).

The matrix $\mathbf{M}$ represents a linear transformation of the 2D space of classifier errors, the eigenvalues can be interpreted as scaling along the dimensions defined by eigenvectors. If the scaling is smaller than one, the errors are reduced in every iteration. In practice, it may be not possible to identify all the errors of the classifier. Therefore, the training does not converge to error-less classifier, but may stabilize at a certain level. Based on the analysis, we further conclude that it is possible to combine imperfect constraints such that their errors are canceling. P-N learning does not put any requirement on the quality of individual constraints, even constraints with very low precision (close to zero) might be used as long as the matrix $\mathbf{M}$ has eigenvalues smaller than one.

On-line P-N Learning. In many problems, the unlabeled data X_u are not known before training, but rather come sequentially. Let $\hat{X}_u^k$ be a set of unlabeled samples that is revealed in time k. All unlabeled data seen so far are then defined as $X_u^k = \{\hat{X}_u^i\}_{i=1:k}$. On-line P-N learning works in a similar way to off-line version with one exception that there is an increasingly larger unlabeled set at disposal. The convergence analysis holds for both off-line and on-line versions.

4. Learning an Object Detector from Video

This section describes an application of P-N learning to the following problem: given a single example of an object, learn an object detector on-line from unlabeled video sequence. This problem will be formalized as on-line P-N learning, where an iteration of the learning process corresponds to discretized time and a frame in time k corresponds to the unlabeled data $\hat{X}_u^k$.

Classifier. Object detectors are algorithms, that decide about presence of an object in an input image and determine its location. We consider type of real-time detectors that are based on a scanning window strategy [21]: the input image is scanned across positions and scales, at each sub-window a binary classifier decides about presence of the object.

Figure 5. A detector based on scanning window and randomized fern forrest classifier. The training and testing is on-line. We use the following setting of the detector: 10,000 windows are scanned, 10 ferns per window, 10 features $\equiv 4^{10}$ posteriors per fern.

The randomized forest classifier [5] was adapted because of its speed, accuracy and possibility of incremental update. Our classifier consists of a number a ferns [18] (simplified trees) that are evaluated in parallel on each patch. Each fern i takes a number of measurements on the input patch resulting in feature vector x_i which points to the leaf-node with posterior probability $\Pr(y = 1|x_i)$. The posteriors from all ferns are averaged and the classifier outputs positive response if the average is bigger than 50%. The measurements taken by each fern are randomly generated a priori and stay unchanged [13] throughout the learning. We adopted 2bit Binary Patterns [12] because of their invariance to illumination and efficient multi-scale implementation using integral images. The posteriors $\Pr(y = 1|x_i)$ represent the internal parameters θ of the classifier and are estimated incrementally throughout the learning process. Each leaf-node records the number of positive p and negative n examples that fell into it during training. The posterior is computed by maximum likelihood estimator, $\Pr(y = 1|x_i) = p/(p + n)$, or is set zero if the leaf is empty.

Training of the initial classifier is performed in the first frame. The posteriors are initialized to zero and are updated by 300 positive examples generated by affine warping of the selected patch [13, 18]. The classifier is then evaluated on all patches. The detections far from the target represent the negative examples and update the posteriors. This approach to extraction of negative training examples is closely related to bootstrapping [11] and stems from the fact that the class priors are highly asymmetric, $\Pr(y = -1) \gg \Pr(y = 1)$.

Constraints. To explain P-N constraints for on-line detector learning, the application scenario is illustrated in Fig. 2. Every patch extracted from the video represents an unlabeled example. Patches within one image are related spatially, i.e. have spatial structure. The patches are also related from one frame to another, i.e. have temporal structure. Therefore knowing a label of a single patch, for example the one selected in the first frame, allows to draw a hypothesis about the labels of other patches. The constraints that will be introduced are based on fact that a single object appears in one location only and therefore its trajectory defines a curve in the video volume. This curve is not continuous due to frame-cuts or low-frame-rate, or even not defined if the object is not present. Parts of the curve are given by an adaptive Lucas-Kanade [16] tracker which follows the selected object from frame to frame. Since the tracker may drift away it is essential to estimate, when the tracker was following the object correctly. For this purpose a confidence of the tracker is defined as NCC between the tracked patch and the patch selected in the first frame. The continuous trajectory is considered correct if the last frame of the trajectory has higher confidence than 80%. If the trajectory is validated it triggers the application of the P-N constraints that exploit the structure of the data. *P-constraints* require that all patches that are close to validated trajectory have positive label. *N-constraints* require all patches in surrounding of a validated trajectory have negative label. Notice, that the appearance of the positive patches is not restricted but they are added to the training set as a consequence of discovered trajectory in the unlabeled data. Negative examples found in the surrounding naturally discriminate against difficult clutter or objects from the same class.

Processing of a video sequence. The P-N learning is initialized in the first frame by learning the *Initial Detector* and setting the initial position of the LK tracker. For each frame, the detector and the tracker find the location(s) of the object. The patches close to the trajectory given by the tracker and detections far away from this trajectory are used as positive and negative examples, respectively. If the trajectory is validated these examples are used to update the detector. However, if there is a strong detection far away from the track, the tracker is re-initialized and the collected examples discarded. The trajectory of the tracker is denoted as *P-N Tracker*, since it is re-initialized by a detector trained by online P-N learning. The detector that is obtained after processing all frames of the sequence is called *Final Detector*.

Evaluation. The performance of the detectors and the tracker is evaluated using standard precision P, recall R and f-measure F statistics. P is the number of correct detections divided by number of all detections, R is the number of correct detections divided by the number of object occurrences that should have been detected. F combines these two measures as $F = 2PR/(P + R)$. A detection was considered to be correct if its overlap with ground truth bounding box was larger than 50%.

Setting the classifier parameters. The classifier has two parameters that determine its accuracy and speed: number of ferns in the forest and number of features measured in each fern (depth). Performance of randomized forest increases with more trees [5], but the speed drops linearly. In

our experiments we use 10 ferns, which is found as a satisfying compromise for most of the objects tested in this paper. The number of features in each fern determines the classifier discriminability. A single 2bit Binary Pattern outputs 4 possible values, if d features are used, the number of leaf-nodes in each fern is 4^d. We were using depths in the range from $d = 6 : 10$ in our experiments, and observed very similar behavior for all of them. For the sake of consistency, the depth is fixed $d = 10$ throughout the paper, but this choice is not critical.

5. Analysis of P-N Learning on Synthetic Data

In this experiment, an object detector will be trained on a real sequence with simulated P-N constraints. The simulation allows to analyze the learning performance for an arbitrary error of the constraints. The purpose of the experiment is to demonstrate that the initial classifier is improved if the P-N constraints are error-canceling.

As discussed in section 3.2, the constraints are error-canceling if the eigenvalues of matrix $\mathbf{M}$ are smaller than one. Matrix $\mathbf{M}$ depends on four parameters, P^+, R^+, P^-, R^-. To reduce this 4D space of parameters, we analyze the system performance at equal error rate. The parameters are set to $P^+ = R^+ = P^- = R^- = 1 - \epsilon$, where ϵ represents error of the constraints. The matrix then becomes $\mathbf{M} = \epsilon\mathbf{I}$, where $\mathbf{I}$ is a 2x2 matrix with all elements equal to 1. The eigenvalues of this matrix are $\lambda_1 = 0, \lambda_2 = 2\epsilon$. Therefore the P-N learning will be improving the performance if $\epsilon < 0.5$. In this experiment, the error is varied in the range $\epsilon = 0 : 0.9$.

For evaluation we used sequence 6 from Fig. 8. The constraints were generated as follows. Suppose in frame k, the classifier generates $\beta(k)$ false negatives. P-constraints relabel $n_c^+(k) = (1 - \epsilon)\,\beta(k)$ of them to positive which guarantees $R^+ = 1 - \epsilon$. In order to satisfy the requirement precision $P^+ = 1 - \epsilon$, the P-constraints relabels additional $n_f^+(k) = \epsilon\,\beta(k)$ background samples to positive. Therefore the total number of examples relabeled to positive in iteration k is $n^+ = n_c^+(k) + n_f^+(k) = \beta(k)$. The N-constraints were generated analogically.

The performance of the detector as a function of number of processed frames is depicted in Fig. 6. Notice that if $\epsilon \leq 0.5$ the performance of the detector increases with more training data. In general, $\epsilon = 0.5$ will give unstable results although in this sequence it leads to improvements. Increasing the noise-level further leads to sudden degradation of the classifier. This simulation results are in line with the theory.

The error-less P-N learning ($\epsilon = 0$) is analyzed in more detail. In this case *all* classifier errors are identified and *no* miss-labeled examples are added to the training set. Three different classifiers were trained using: (i) P-constraints, (ii) N-constraints, (iii) P-N constraints. The classifier performance was measured using precision, recall and f-measure and the results are shown in Fig. 7. Precision (LEFT) is decreased by P-constraints since only positive examples are added to the training set, these cause the classifier to be too generative. Recall (MIDDLE) is decreased by N-constraints since these add only negative examples and cause the classifier to be too discriminative. F-Measure (RIGHT) shows that using P-N filters together works the best. Notice, that even error-less constraints cause classification errors if used individually, which leads to low precision or low recall of the classifier. Both precision and recall are high if the P-N constraints are used together since the errors are mutually compensating.

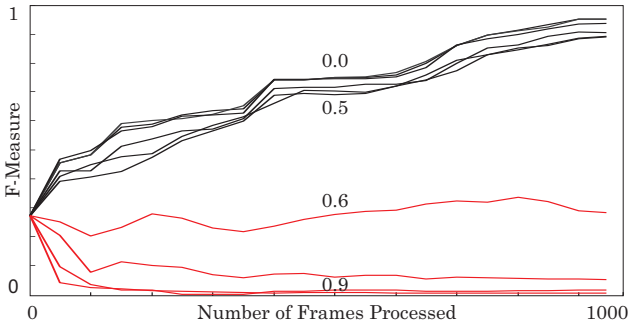

Figure 6. Performance of a detector as a function of the number of processed frames. The detectors were trained by synthetic P-N constraints with certain level of error. The classifier is improved up to error 50% (BLACK), higher error degrades it (RED).

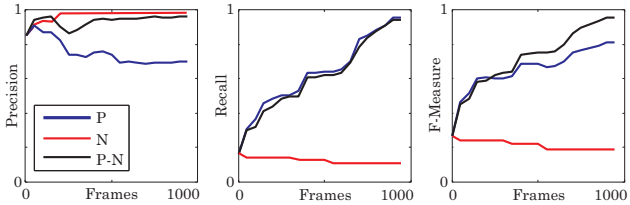

Figure 7. Performance of detectors trained by error-less P-constraints, N-constraints and P-N constraints measured by precision (LEFT), recall (MIDDLE) and f-measure (RIGHT). Even perfect P or N constraints, on their own, generate classifier errors.

6. Experiments on Real Data

Learning of an object detector is tested on 10 video sequences illustrated in Fig. 8. The sequences contain various objects in challenging conditions that include abrupt camera motion, motion blur, appearance change and partial or full occlusions. All sequences have been processed with the same parameter setting as discussed in Sec. 4. The output of each experiment is the initial and final detector and the trajectory given by the P-N tracker.

Evaluation of P-N Tracker. Sequences 1-6 were used in [22] for comparison of recent tracking systems [15, 7,

Sequence	Frames	[15]	[7]	[2]	[3]	[22]	**P-N Tracker**
1. David	761	17	n/a	94	135	759	**761**
2. Jumping	313	75	313	44	313	313	**313**
3. Pedestrian 1	140	11	6	22	101	**140**	27
4. Pedestrian 2	338	33	8	118	37	240	**338**
5. Pedestrian 3	184	50	5	53	49	154	**184**
6. Car	945	163	n/a	10	45	802	**945**

Table 1. Comparison with recent tracking methods in terms of the frame number after which the tracker doesn't recover from failure.

2, 22], this experiment adds [3] and our P-N tracker. The performance measure is the frame number after which the system does not recover from failure. Table 1 shows the resulting performance. In 5 out of 6 videos the P-N tracker is able to track the object up to the end of the sequence. In sequence 3, the P-N tracker fails after frame 27, while [3] is able to track up to frame 101 and [22] up to 140. This video shows abrupt camera motions and the Lucas-Kanade tracker fails quickly, therefore the P-constraints did not identify sufficient number of training examples to improve the initial detector and it was not able to recover the tracker from its failure.

Detailed performance analysis of the P-N learning is performed on all 10 sequences from Fig. 8. The performance of the initial detector, final detector and the P-N tracker is measured using precision, recall and f-measure. Next, the P-N constraints are measured by P^+, R^+, P^- and R^- averaged over time.

Table 2 (3rd column) shows the resulting scores of the initial detector. This detector has high precision for most of the sequences with exception of sequence 9 and 10. Sequence 9 is very long (9928 frames) there is a significant background clutter and objects similar to the target (cars). Recall of the initial detector is low for the majority of sequences except for sequence 5 where the recall is 73%. This indicates that the appearance of the object does not vary significantly. The scores of the final detector are displayed in the 4th column of the Table 2. The recall of the detector was significantly increased with little drop of precision. In sequence 9, even the precision was increased from 36% to 90%, which shows that the false positives of the initial classifier were identified by N-constraints and corrected. Most significant increase of the performance is for sequences 7-10 which are the most challenging of the whole set. The initial detector fails here but for the final detector the f-measure in the range of 25-83%! This demonstrates the benefit of P-N learning. The 5th column evaluates the P-N tracker. Its precision is typically lower than precision of the final detector, since the entire trajectory of the tracker was considered for evaluation, including drifts. In sequences 1 and 4 the tracker significantly outperforms the final detector. This shows that the tracker was following the object correctly but its trajectory was not validated by the constraints. Interesting observation is from sequences 2,3,7 and 8 where the tracker gives lower scores than the final detector. This demonstrates that even less reliable tracker is able to train an accurate detector in P-N learning.

Sequences 7 and 8 have been used in [12] and the following results were reported. Performance of tracker in sequence 7: 0.88/0.82/0.85, sequence 8: 0.96/0.54/0.69. The results are comparable to ours (Table 2, row 7-8, column 5).

The last three columns of Table 2 report the performance of P-N constraints. Both constraints have precision higher than 60% except for sequence 10 which has P-precision just 31%. Recall of the constraints is in the range of 2-78%. The last column shows the corresponding eigenvalues of matrix $\mathbf{M}$. Notice that all eigenvalues are smaller than one. This demonstrates that the proposed constraints work across different scenarios and lead to improvement of the initial detector. The larger these eigenvalues are, the less the P-N learning improves the performance. For example in sequence 10 one eigenvalue is 0.99 which reflects poor performance of the P-N constraints. The target of this sequence is a panda, which changes its pose throughout the sequence. Lucas-Kanade and NCC are not very reliable in this scenario, but still P-N learning exploits the information provided by the tracker and improves the detector.

Conclusions

P-N learning, a novel approach for processing of labeled and unlabeled examples, has been proposed. The underlying assumption of the learning process is that the unlabeled data are structured. The structure of the data is exploited by positive and negative constraints that restrict the labeling of the unlabeled data. These constraints provide a feedback about the performance of the classifier which is iteratively improved in a bootstrapping fashion. We have formulated conditions under which the P-N learning guarantees improvement of the classifier. The conditions have been validated on synthetic and real data.

The P-N learning has been applied to the problem of on-line learning of an object detector from a single example and an unlabeled video sequence. We have proposed novel constraints that exploit the spatio-temporal properties of a video and demonstrated that they lead to significant improvement of the detector for variety of objects and conditions. Since the learning runs on-line (at 20 fps), the system has been compared with relevant tracking algorithms and state-of-the-art results have been achieved.

The formalized P-N learning theory enables to guide the design of structural constraints that satisfy the requirements on the learning stability. We believe that designing of more sophisticated structural constraints is a promising direction for future research.

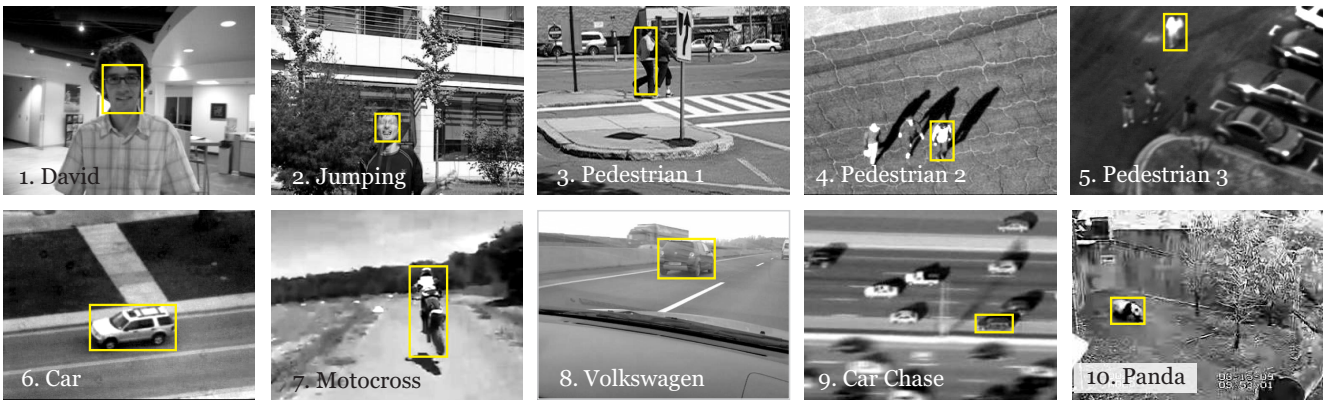

Figure 8. Sample images from evaluation sequences with objects marked. Full videos with ground truth are available online.

Sequence	Frames	Initial Detector Precision / Recall / F-measure	Final Detector Precision / Recall / F-measure	P-N Tracker Precision / Recall / F-measure	P-constraints P^+, R^+	N-constraints P^-, R^-	Eigenvalues λ_1, λ_2
1. David	761	1.00 / 0.01 / **0.02**	1.00 / 0.32 / **0.49**	0.94 / 0.94 / **0.94**	1.00 / 0.08	0.99 / 0.17	0.92 / 0.83
2. Jumping	313	1.00 / 0.01 / **0.02**	0.99 / 0.88 / **0.93**	0.86 / 0.77 / **0.81**	0.86 / 0.24	0.98 / 0.30	0.70 / 0.77
3. Pedestrian 1	140	1.00 / 0.06 / **0.12**	1.00 / 0.12 / **0.22**	0.22 / 0.16 / **0.18**	0.81 / 0.04	1.00 / 0.04	0.96 / 0.96
4. Pedestrian 2	338	1.00 / 0.02 / **0.03**	1.00 / 0.34 / **0.51**	1.00 / 0.95 / **0.97**	1.00 / 0.25	1.00 / 0.24	0.76 / 0.75
5. Pedestrian 3	184	1.00 / 0.73 / **0.84**	0.97 / 0.93 / **0.95**	1.00 / 0.94 / **0.97**	0.98 / 0.78	0.98 / 0.68	0.32 / 0.22
6. Car	945	1.00 / 0.04 / **0.08**	0.99 / 0.82 / **0.90**	0.93 / 0.83 / **0.88**	1.00 / 0.52	1.00 / 0.46	0.48 / 0.54
7. Motocross	2665	1.00 / 0.00 / **0.00**	0.92 / 0.32 / **0.47**	0.86 / 0.50 / **0.63**	0.96 / 0.19	0.84 / 0.08	0.92 / 0.81
8. Volkswagen	8576	1.00 / 0.00 / **0.00**	0.92 / 0.75 / **0.83**	0.67 / 0.79 / **0.72**	0.70 / 0.23	0.99 / 0.09	0.91 / 0.77
9. Car Chase	9928	0.36 / 0.00 / **0.00**	0.90 / 0.42 / **0.57**	0.81 / 0.43 / **0.56**	0.64 / 0.19	0.95 / 0.22	0.76 / 0.83
10. Panda	3000	0.79 / 0.01 / **0.01**	0.51 / 0.16 / **0.25**	0.25 / 0.24 / **0.25**	0.31 / 0.02	0.96 / 0.19	0.81 / 0.99

Table 2. Performance analysis of P-N learning. The Initial Detector is trained on the first frame only. The Final Detector is obtained by P-N learning after one pass through the sequence. The P-N Tracker is the adaptive Lucas-Kanade tracker re-initialized by the on-line trained detector. The last three columns display internal statistics of the training process, for explanation of the variables see Section 3.

Acknowledgment. This research was supported by UK EPSRC EP/F0034 20/1 and the BBC R&D grants (ZK, KM) and by Czech Science Foundation project 102/07/1317 (JM).

References

[1] Y. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272(4):64–71, 1995. 2

[2] S. Avidan. Ensemble tracking. *PAMI*, 29(2):261–271, 2007. 2, 7

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. *CVPR*, 2009. 3, 7

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *COLT*, 1998. 1, 2

[5] L. Breiman. Random forests. *ML*, 45(1):5–32, 2001. 5

[6] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. 1, 2

[7] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005. 2, 7

[8] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *CVPR*, 2, 2003. 2

[9] H. Grabner and H. Bischof. On-line boosting and vision. *CVPR*, 2006. 2

[10] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. *CVPR*, 2005. 2

[11] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. *BMVC*, 2008. 5

[12] Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. *OLCV*, 2009. 3, 5, 7

[13] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. *CVPR*, 2005. 5

[14] A. Levin, P. Viola, and Y. Freund. Unsupervised improvement of visual detectors using co-training. *ICCV*, 2003. 2

[15] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. *NIPS*, 2005. 2, 7

[16] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, 81:674–679, 1981. 5

[17] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2):103–134, 2000. 1, 2

[18] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *CVPR*, 2007. 5

[19] N. Poh, R. Wong, J. Kittler, and F. Roli. Challenges and research directions for adaptive biometric recognition systems. *ICAB*, 2009. 2

[20] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. *WACV*, 2005. 2

[21] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001. 4

[22] Q. Yu, T. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. *ECCV*, 2008. 3, 6, 7

[23] K. Zhou, J. Doyle, and K. Glover. *Robust and optimal control*. Prentice Hall Englewood Cliffs, NJ, 1996. 4

[24] X. Zhu and A. Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool Publishers, 2009. 1, 2