On-line Random Naive Bayes for Tracking

Martin Godec, Christian Leistner, Amir Saffari, Horst Bischof
Institute for Computer Vision and Graphics
Graz University of Technology
Graz, Austria
Email: {godec, leistner, saffari, bischof}@icg.tugraz.at

***Abstract*—Randomized learning methods (*i.e.*, Forests or Ferns) have shown excellent capabilities for various computer vision applications. However, it was shown that the tree structure in Forests can be replaced by even simpler structures, *e.g.*, Random Naive Bayes classifiers, yielding similar performance. The goal of this paper is to benefit from these findings to develop an efficient on-line learner. Based on the principals of on-line Random Forests, we adapt the Random Naive Bayes classifier to the on-line domain. For that purpose, we propose to use on-line histograms as weak learners, which yield much better performance than simple decision stumps. Experimentally we show, that the approach is applicable to incremental learning on machine learning datasets. Additionally, we propose to use an *iir filtering-like* forgetting function for the weak learners to enable adaptivity and evaluate our classifier on the task of tracking by detection.**

***Keywords*-On-line Learning; Object Tracking; Naive Bayes;**

I. Introduction

Machine learning methods have been applied to various computer vision problems, *e.g.*, detection, segmentation, or tracking. Recently, randomized learning techniques such as Random Forests [1], [2], [3] and Ferns [4] have been introduced to the vision community. These methods use randomized input selection and random feature selection to create a diverse classifier ensemble. While yielding state-of-the-art results, randomized methods possess several properties that make them particularly interesting for computer vision applications. First, they are very fast in both, training and classification. Second, they can easily be parallelized, which makes them interesting for multi-core and GPU implementations [5]. Third, they are inherently multi-class and do not require to build several binary classifiers to solve a multi-class problem and fourth, the randomization increases the stability and decreases the variance of the trained classifier. Typically, randomized methods are trained off-line, but many applications (*e.g.*, tracking) require on-line training capabilities. This is especially the case if the problem is based on a sequential data source or the learning algorithm has to train on a very large dataset.

Prinzie and van den Poel [6] have shown that the tree structure of Random Forests can be replaced by simpler learning methods such as Naive Bayes without decreasing the performance. Especially the low computational and memory costs of Random Naive Bayes makes it suitable for applications where computational power and memory are limited or if very large datasets have to be processed. Hence, the goal of this work is to adapt this method to the on-line domain.

Inspired by the on-line Random Forest approach of Saffari *et al.* [7], we propose a novel on-line learning method based on Random Naive Bayes classifiers. Compared to on-line Random Forests, our algorithm simplifies the implementation and configuration of on-line randomized learners. In addition, we benefit from the lower memory consumption, high computational efficiency, and fast convergence. We show that our on-line version is able to compete with on-line Random Forests and on-line boosting on different datasets and the task of visual tracking.

In Section II, we first briefly review Random Naive Bayes and derive an on-line formulation. Section III delivers several experiments on both machine learning and tracking tasks. Finally, the paper concludes with Section IV.

II. On-line Random Naive Bayes

In the following, we give an introduction to Naive- and Random Naive Bayes classification. Subsequently, we develop an on-line formulation, which can be used for various applications.

A. Naive Bayes

Given a training dataset $\mathcal{X} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x} \in \mathbb{R}^D$ are the samples in a D-dimensional feature space and $y \in \{1, \cdots, K\}$ are the corresponding labels for a K-class classification problem, then *Bayes'* theorem can be formulated as

$$p(y = i|\mathbf{x}) = \frac{p(i)p(\mathbf{x}|i)}{p(\mathbf{x})}, \tag{1}$$

where $p(i)$ is the class label prior, and $p(y|\mathbf{x})$ is the unknown probability distribution of the joint space of features $\mathbf{x}$ and labels i.

Since these distributions are unknown, we need to estimate them using the given training data. Here, Naive Bayes (NB) is a popular method to simplify this estimation, where we assume independence of features given the class labels

$$p(x_1, x_2, \ldots, x_D|i) \sim p(x_1|i)p(x_2|i)\ldots p(x_D|i), \tag{2}$$

where x_d denotes the d^{th} dimension of the feature vector $\mathbf{x}$. Despite of the naive independence assumption, Naive Bayes delivers good results [8], [9], [6]. In computer vision, features are usually pieces of local information, thus considering them independently is a feasible assumption. Assuming independence and uniform label distribution, a classifier $F(\mathbf{x})$ can be written as

$$F(\mathbf{x}) = \arg\max_i \prod_{d=1}^{D} p(x_d|i). \tag{3}$$

Thereby, the classification result can be calculated by building the product of all class probabilities for the current feature values.

B. Randomized Learning

Randomized learning methods build on mainly two ideas: using (a) random input selection and (b) random feature selection to train an ensemble consisting of classifiers of similar type. The main advantages of these methods are the increased stability and decreased variance of the resulting classifier. Using random input selection (*e.g.*, *bagging* [10]), several classifiers are trained on different subsets of the training space and additively combined to an ensemble. Bagging has shown to improve the final classifier in terms of stability and classification accuracy and helps to avoid overfitting. In order to perform bagging on-line, Oza [11] proposed to model the sequential arrival of the data by a Poisson distribution.

Using randomized decision trees in a bagged ensemble leads to *Random Forests*. To transform off-line Random Forests [1] to the on-line domain, Saffari *et al.* [7] proposed a tree growing scheme to establish decision trees on-line. They initialize each tree node with a number of random tests and perform splitting of this nodes after a sufficient number of samples have been processed and a splitting criterion is satisfied. The particular node is then frozen by selecting the random test with the highest quality. Since the splitting criterion is a crucial part of the algorithm, it has to be designed and configured very carefully to achieve good performance. As the node count grows exponentially with the tree depth, the memory consumption of this approach cannot be neglected.

Recently, Prinzie and Van den Poel [6] generalized the idea of off-line Random Forests (RF) to MultiNomial Logit and Naive Bayes. They showed that the Random Naive Bayes (RNB) classifier using both, bagging and random feature selection, achieves competitive performance to Random Forests (RF). Hence, we transform their approach to the on-line domain.

C. On-line Learning

When creating the RNB classifiers ensemble, for each classifier we randomly select F features out of the feature pool D. The probability distribution $p(x_f|y)$ of each feature is then modeled for each class y. The probability of a sample observation $\mathbf{x}$ belonging to the class y can then be described as

$$p(\mathbf{x}|y) \sim \sum_{b=1}^{B} \prod_{f=1}^{F} p^b(x_f|y), \tag{4}$$

where we combine B randomly trained Naive Bayes classifiers, each using $F \leq |D|$ features. Following the results of Geurts [12], using randomized threshold selection instead of random input selection does not decrease the performance of the classifier and bagging can be skipped. We perceived similar results for the proposed formulation, even using histograms instead of binary decisions.

Since we want to enable on-line learning for our RNB ensemble, we need to estimate the probability distribution for the given feature x_f on-line. We propose to use equally binned histograms to estimate the probability distributions since they are very fast and easy to implement. Moreover, histograms are applicable to incremental learning and can handle multi-modal distributions easily. Thus, they perfectly fit into our targeted learning scheme.

Since some learning problems require temporal knowledge weighting, we have to consider unlearning of information. This is required, for instance, if we have to cope with temporary noise or outliers and *concept drift*. We propose to use an *iir-like* (*i.e.*, infinite impulse response) filtering for each histogram bin, where the value of each bin can be calculated as

$$w_{t_0}^{norm} = \sum_{t=-\infty}^{t_0} w_t \cdot r^{t_0-t}, \tag{5}$$

where w_t is the learned sample weight at time t. t_0 represents the time of the current update. The speed of forgetting can be defined with the parameter r.

To enhance the significance of the feature pool, we decided to build random hyperplanes within our feature space. These hyperplanes x_h are computed as a weighted linear combination of the features $\mathbf{x}$ in the form $x_H = \mathbf{w}^T\mathbf{x}$, where the weights $\mathbf{w} \in [-1, 1]^D$ for each feature are randomly chosen and $\sum_{d=1}^{D} |w_d| = 1$. The weight vector $\mathbf{w}$ is chosen very sparse in order to only create small local subspaces. A common heuristic would be the selection of $H \approx \log D$ [1]) features with non-zero weights. Experiments show that this gives great improvement of classification performance (see Table II) due to the capturing of dependencies between different object parts.

III. Experiments

In the following, we demonstrate the proposed method for different applications, showing that competitive results can be obtained. To show the generality, we apply the method to different machine learning datasets. We compare our approach to on-line Random Forests (ORF) [7] and on-line AdaBoost (OAB) [13]. Subsequently, we apply our method

to visual tracking and compare to current state-of-the art methods.

A. Machine Learning

For evaluation, we use the *DNA*, *Letter*, and *USPS* datasets from the LibSVM Repository[1] and *g50c* from the Semi-Supervised Benchmarks[2] and the same configuration of our ORNB algorithm for all machine learning experiments ($B = 200$, $F = 20$ and $H = 2$). We process all datasets *10* times and report the average classification error. We compare to on-line AdaBoost (OAB) (200 selectors, each 100 features) and on-line Random Forest (ORF) (200 trees with max. depth of 10), both trained for *5* epochs. For multi-class datasets, we employ one-vs-all for OAB. Table I shows that the ORNB classifier outperforms the OAB classifier and reaches comparable results to the ORF.

Dataset	ORNB	OAB	ORF
DNA	**0.098**	0.146	*0.112*
Letter	*0.196*	0.223	**0.169**
USPS	*0.183*	0.184	**0.127**
g50c	*0.125*	0.309	**0.124**

Table I
AVERAGE CLASSIFICATION ERROR (BOLD: BEST RESULT, ITALIC: SECOND BEST).

Due to the small parameter set, we can deemonstrate the influence of each parameter easily (see Table II). It shows that the dimension of the hyperplane features H and the number of features F have large influence on the classification performance, while the bag size B mainly reduces the variance of the classifier.

Parameter	Setting and classification error				
B	50	100	**200**	500	1000
$error$	0.167	0.141	0.125	0.109	0.103
F	1	5	10	**20**	50
$error$	0.182	0.126	0.122	0.125	0.133
H	1	**2**	3	5	10
$error$	0.294	0.125	0.115	0.089	0.094

Table II
PARAMETER INFLUENCE (BOLD: STANDARD SETTINGS USED FOR OTHER EXPERIMENTS).

Finally, we analyzed the runtime complexity and memory consumption of the compared methods. For training, all methods have a linear relation between training data and runtime (ignoring the calculations of boosting loss and tree splits), but ORNB only needs one training epoch and reaches the final classification performance after approximatley 40% of the training data for the used datasets. During testing, OAB is the fastest, evaluating only a subset of weak learners, while ORF depends linearly on the tree depth and ORNB on the number of features. The memory consumption of OAB and ORNB is linearly related to the classifier size, but grows exponentially for ORF.

Sequence	ORNB	OAB [14]	ORF [7]	MIL [15]
Sylvester[3]	**0.75**	*0.65*	0.62	0.60
David[3]	**0.79**	0.26	*0.69*	0.57
Face Occlusion 2[4]	**0.82**	0.68	*0.72*	0.68

Table III
AVERAGE OVERLAP SCORE (BOLD: BEST RESULT, ITALIC: SECOND BEST).

B. Visual Object Tracking

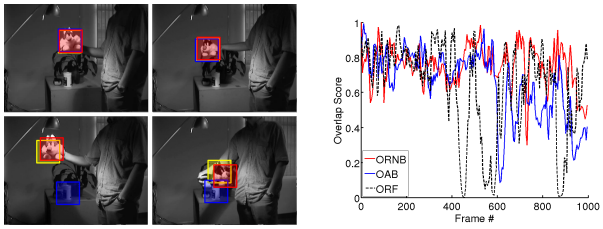
Finally, we compare the performance of ORNB to OAB [14] and ORF [7] for the task of tracking using a forgetting rate r of 0.95, which corresponds to a stable model and slow adaptions. Additionally, we state the results of On-line MILBoost [15], which can be seen as current state-of-the-art method for this task. We use the overlap-criterion of the VOC Challenge [16], which is defined as $A_{overlap} = R_T \cap R_{GT} / R_T \cup R_{GT}$, where R_T is the tracking rectangle and R_{GT} the groundtruth. Due to randomization within the algorithms, we report the average overlap score over 5 runs. To offer a fair comparison, we only use simple Haar-like features and a completely randomized feature pool. The initial training is done only on virtual samples generated out of the first frame (*i.e.*, affine transformations of this frame). We use three benchmark sequences including variations in illumination, pose, scale, rotation and appearance, and partial occlusions.

The experiments show that tracking with ORNB is resistant to various appearance variations (see Figures 1, 2, and 3 for some illustrative examples) and achieves superior results to OAB, ORF and MIL for the tested sequences. Even if our approach results in a rather stable classifier, it is able to adapt to certain changes over time due to the *forgetting rate*. The alignment of the detections is very accurate, which guarantees high quality of the samples used for self-updating. The conservative forgetting rate allows for recovery after occlusions, but preserving a certain amount of adaptivity.

IV. CONCLUSION

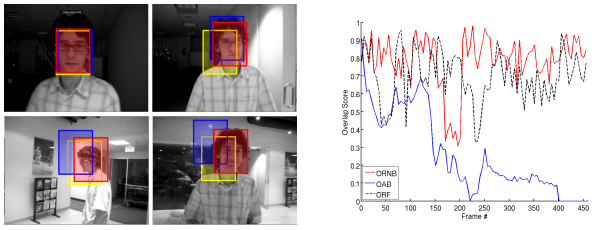
Motivated by recent work on on-line Random Forests and the efficiency of Naive Bayes classifiers, we derived an on-line multi-class Random Naive Bayes (ORNB) algorithm. In particular, we combined an ensemble of Naive Bayes classifiers using random feature selection and histograms to establish a robust and fast classifier. To show the efficiency

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[2] http://www.kyb.tuebingen.mpg.de/ssl-book/
[3] http://www.cs.toronto.edu/~dross/ivt/ [17]
[4] http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml



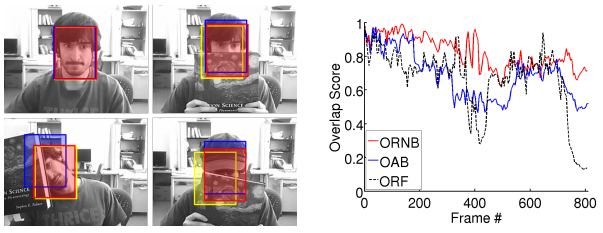
(a) Sylvester (b) Tracking Score

Figure 1. Illustrative examples for *Sylvester* sequence (red: ORNB; blue: OAB; yellow/black: ORF).



(a) David (b) Tracking Score

Figure 2. Illustrative examples for *David* sequence (red: ORNB; blue: OAB; yellow/black: ORF).



(a) Face Occlusion 2 (b) Tracking Score

Figure 3. Illustrative examples for *Face Occlusion 2* sequence (red: ORNB; blue: OAB; yellow/black: ORF).

and the generality of the approach, we applied it to multi-class classification and visual tracking. For both cases, we showed that competitive results can be obtained. The main advantage of our method is the simplicity (*e.g.*, low memory consumption, reduced computational complexity, small parameter set, high parallelization potential), which makes the method highly applicable.

Future work will focus on several issues within the learning algorithm: using the *out-of-bag-error* enables (a) classifier selection depending on the error and (b) alternative forgetting methods. Further a heterogeneous classifier ensembles (*e.g.*, mixing of random naive bayes and random trees) or heterogeneous feature pools (*e.g.*, color, local binary patterns, histogram of oriented gradients) could improve the performance. Additionally, ORNB classifiers could be used as weak learners for other high-level on-line learning algorithms, such as on-line boosting.

Acknowledgment

This work was supported by the FFG projects EVis (813399) and OUTLIER (820923) under the FIT-IT programme.

References

[1] L. Breiman, "Random forests," *Machine Learning*, 2001.

[2] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Proc. IEEE CVPR*, 2008.

[3] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2006.

[4] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

[5] T. Sharp, "Implementing decision trees and forests on a gpu," in *Proc. ECCV*, 2008.

[6] A. Prinzie and D. Van den Poel, "Random multiclass classification: Generalizing random forests to random mnl and random nb," in *Database and Expert Systems Applications*, 2007.

[7] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *Proc. IEEE OLCV Workshop*, 2009.

[8] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," 1997.

[9] D. J. Hand and K. Yu, "Idiot's bayes: Not so stupid after all?" *International Statistical Review*, 2001.

[10] L. Breiman, "Bagging predictors," in *Machine Learning*, 1996.

[11] N. C. Oza, "Online ensemble learning," Ph.D. dissertation, University of California, Berkeley, 2001.

[12] P. G. June, "Extremely randomized trees," in *Machine Learning*, 2003.

[13] C. Leistner, A. Saffari, P. M. Roth, and H. Bischof, "On robustness of on-line boosting - a competitive study," in *Proc. IEEE OLCV Workshop*, 2009.

[14] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE CVPR*, vol. I, 2006, pp. 260–267.

[15] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online mulitple instance learning," in *Proc. IEEE CVPR*, 2009.

[16] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object class challenge 2007," in *VOC*, 2007.

[17] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *Intern. Journal of Computer Vision*, 2008.