

# Patch Learning for Incremental Classifier Design

Poster: Rudy Sicard<sup>1</sup> and Thierry Artières<sup>2</sup> and Eric Petit<sup>1</sup>

**Abstract.** We present a learning algorithm for nominal data. It builds a classifier by adding iteratively a simple patch function that modifies the current classifier. Its main advantage lies in the possibility to learn every patch function parameters optimally from the Bayesian point of view hence avoiding overtraining.

## 1 INTRODUCTION

We present a learning algorithm for nominal data for designing complex classifiers while avoiding overtraining limitation, which is useful for small training sets and high dimensional data. Bayesian Model Averaging (BMA) provides a theoretical solution for avoiding overtraining [1] through computing expected probabilities by summing over all model parameters values. Unfortunately BMA is usually intractable so that approximations were proposed like Point Estimate Approximation (PEA) [2]. PEA consists in replacing the expectation over parameter value by a single term computed for one particular parameter value, Maximum A Posteriori (MAP) estimation is a typical example. However, MAP requires an ad-hoc (since it depends on the data and the task) parameter prior distribution that favours simple classifiers. Other approximation schemes have been proposed in the past such as Kullback Leibler (KL) projection [3] which has been used up to now for simple classifiers only. Building on KL projection we propose to learn incrementally a complex classifier in a way that is close to Boosting [4], by adding iteratively simple patch functions, where each patch function slightly modifies the current classifier. The major idea of our approach is to use simple enough patch functions so that above approximation scheme (and then optimal learning) is tractable. We focus on patch functions defined with a single parameter. Depending of the nature of the patch functions the built classifier resembles standard classifiers like Naïve Bayes (NB), logistic classification or other standard techniques. We first present what a patch function is and its optimal learning. Then we discuss incremental learning of complex classifiers. Lastly, we report experimental results that show the generalization ability and the accuracy of our technique on various benchmark datasets.

## 2 PATCH DESIGN AND LEARNING

For the sake of clarity we focus here on a simple problem with two classes but the extension to a more general multi-class problem is straightforward. In our mind a patch is an elementary function that modifies the behaviour of a current probabilistic classifier,  $F$ , that takes as input a vector  $x$  (of  $d$  nominal features) and outputs class posterior probabilities  $\{P_F(C_j/x), j = 1, 2\}$ . In the following a patch

is defined with one parameter  $\alpha$  and a Boolean test  $T$ . It is used to modify  $F$  into  $F'_{T,\alpha}$  by changing the behaviour of  $F$  for inputs that verify  $T$  only. New classifier  $F'_{T,\alpha}$  is defined as follows:

$$\text{if } T(x) \text{ is true } \begin{cases} P_{F'_{T,\alpha}}(C_1/x) = 1/z(x) \times \alpha \times P_F(C_1/x) \\ P_{F'_{T,\alpha}}(C_2/x) = 1/z(x) \times (1-\alpha) \times P_F(C_2/x) \end{cases} \quad (1)$$

$$\text{else } P_{F'_{T,\alpha}}(C_j/x) = P_F(C_j/x), j = 1, 2$$

where  $z(x) = (\alpha \times P_F(C_1/x) + (1-\alpha) \times P_F(C_2/x))$  is a normalization factor. Examples of test functions would be:  $T(x)$  is true if the  $i^{\text{th}}$  feature of  $x$  has a particular value  $v$  (single feature patch) or  $T(x)$  is true if the  $i^{\text{th}}$  feature has a value  $u$  and the  $j^{\text{th}}$  feature has value  $v$  (double feature patch). Learning a patch is done through Bayesian learning. Using posterior class probability estimated with  $F'_{T,\alpha}$ , the posterior probability for parameter  $\alpha$  (thus indexed by  $F'_T$ ) is:

$$P_{F'_T}(\alpha / D) \propto Z(D) \times P(\alpha) \prod_{i/T(x_i)=\text{true}} P_{F'_{T,\alpha}}(y_i / x_i) \quad (2)$$

where  $Z(D)$  is a normalization factor,  $P(\alpha)$  is a prior distribution and the product ranges over training samples  $x_i$  (whose class is  $y_i$ ) that verify the test  $T$ . The optimal Bayesian classifier may be defined through Bayesian Model Averaging [1]:

$$P_{F'_T}(C_i / D, x) = \int_{\alpha=0}^1 p_{F'_{T,\alpha}}(C_i / x) p_{F'_T}(\alpha / D) d\alpha \quad (3)$$

Given  $x$ , this BMA distribution minimizes the distance (measured as the KL divergence [3]) with the true distribution by averaging over  $\alpha$  [1]. Computing this distribution may be done analytically for well chosen priors but this is expensive. Here, we seek to approximate this term with a PEA strategy which aims at replacing summation like in Eq. (4) with one term  $p_{F'_{T,\alpha}}(C_i / x)$ . Following

[3] we chose to seek a PEA approximation that is as close as possible to this optimal BMA classifier, w.r.t. KL divergence. Since the optimal PEA may depend on  $x$ , i.e. may be obtained with possibly different  $\alpha$ , the solution is to minimize the expected KL divergence, over  $x$ , of PEA with respect to BMA [2]. We do not detail here methods to perform this minimization for space reasons but this is not a critical point, it is performed numerically (note that the criterion is convex and hence does not exhibit local minima).

## 3 COMPLEX CLASSIFIER CONSTRUCTION

Using the patch learning routine discussed above, one can build incrementally complex classifiers by adding sequentially a number of patches, starting from e.g. an initial neutral classifier ( $\forall x, \forall j = 1..2, P(C_j/x) = 0.5$ ). Let  $\{(T_i, \alpha_i)_{i=1..N}\}$  be the successive

<sup>1</sup> France Télécom – R&D/TECH/IDEA/TIPS – France.

<sup>2</sup> LIP6 - Université Pierre et Marie Curie, France.

patches that are learned for building a classifier  $F$ , each patch is defined by a Boolean test  $T_i$  and a parameter value  $\alpha_i$ . Then, the final classifier may be written in a standard form:

$$P(C_1/x) = Z(x) \times \exp(W^t U(x)) \quad (4)$$

where  $W = (\log(\alpha_i))_{i=1..N}$  is the parameter vector for classifier  $F$  and  $U(x) = (T_i(x))_{i=1..N}$  is the feature vector corresponding to input  $x$  (with  $T_i(x) = 0/1$ ). Depending on the nature of the patches, the final classifier may correspond to various classical models. For instance, when considering single feature patches, decision boundaries are linear and have similar form as for NB or logistic classification models. Training consists in applying iteratively a number of patches (eventually all possible patches). In the case of single feature patches, learning algorithm resumes to:

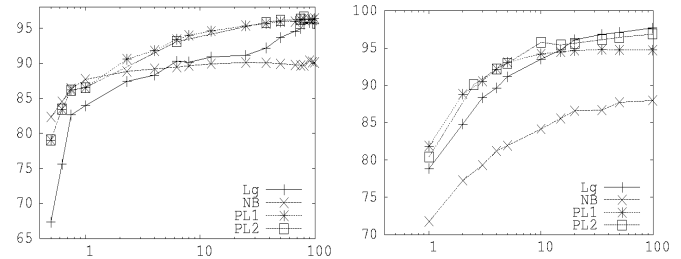
For each feature  $a$  and for each feature value  $v_a$   
 Add the patch with test  $T(x)=[x.a==v_a]$  and  
 optimal  $\alpha$  parameter as discussed in section 2.

As usual, one may design more complex classifier through the use of complex features and the addition of many features but this may raise overtraining problem. At the opposite, using our optimal approach described in section 2 should overcome this problem. The counterpart of this generalization ability is that our approach cannot be shown to be the globally optimal. It is locally optimal only in that each patch is learned optimally. For instance, the final classifier will depend on the order the patches are added. However, we believe this global non optimality must be considered together with its generalization power.

## 4 EXPERIMENTS

We performed experiments on a number of UCI benchmark datasets<sup>4</sup> and compared our approach with standard techniques using the Weka package [5]. We report here two series of experiments. First we investigated overtraining robustness and compared our patch learning algorithm with single and double features patches to NB and Logistic classification (Lg). Figure 1 shows their performances averaged over 100 runs as a function of the training set size with a logarithmic scale to focus on small sizes on two datasets whose settings are detailed in Table 1. It may be seen first that NB is significantly less accurate than other approaches, second that our approach (with single or double patches) significantly outperforms others for a wide range of training set sizes, third that as more training data becomes PL2 and Lg tend to behave similarly while slightly outperforming PL1 on the *kr-kp* dataset. This was clearly expected. With small training set size our approach naturally outperforms standard methods such as Lg or NB. As more training data becomes available all models (except NB) perform similarly on *vote* dataset since it is almost linearly separable while PL1 perform slightly lower than richer (PL2) and globally optimally learned (Lg) models on the more difficult *kr-kp* dataset. Note that Lg may slightly outperform patch learning in some cases as we will see, which may come from the fact that Lg is globally optimal. The second series of experiments compare patch learning to standard methods, NB, Lg and AODE [6] on a number of UCI benchmark datasets (Table 1) with a standard training set size and a 10-fold cross validation procedure. It may be seen that, except for the *zoo* dataset, double feature

patches outperform single feature patches, that single patches is comparable or better than standard techniques and that double features patches outperforms all other methods in average. Hence patch learning is not only able to learn complex classifier while avoiding overtraining (as in Fig. 1), it also allows learning efficient classifiers that compare well to more standard techniques.



**Figure 1.** Comparison of generalization ability for Naïve Bayes (NB), Logistic Classification (Lg), Patch learning with single (PL1) and double (PL2) feature patches. Performances are plotted as a function of the size of the training set (percentage of whole dataset size) on the UCI *vote* dataset (left) and *kr-kp* dataset (right) whose settings are detailed in Table 1.

**Table 1.** Comparison of error-rates for Naïve Bayes (NB), AODE, Logistic classification (Lg) and patch learning (PL1 and PL2) on various datasets from UCI<sup>4</sup>. The last row indicates mean error-rate relatively to NB performance. Note that the number of classes  $\#C$ , the dimension of data  $d$  and the training set size  $TSS$  (#samples) are indicated for each dataset.

Data set	#C	$d$	TSS	NB	AODE	PL1	PL2	Lg
Audiology	23	70	226	26.6	26.6	19.1	16.4	17.3
Hepatis	2	18	155	17.5	16.8	18.1	16.8	22.7
Kr-kp	2	37	3196	12.1	8.8	5.4	2.9	2.5
Mushroom	2	22	8124	4.2	0.05	0.05	0.0	0.0
P-tumor	23	18	329	49.9	50.4	53.4	54.3	55.1
Splice	3	60	3190	4.7	3.9	3.6	3.6	9.0
Vote	2	16	435	9.9	5.7	3.9	3.2	3.9
Zoo	7	17	101	6.8	4.9	2.8	3.8	5.8
Average performance				1.00	0.73	0.61	0.57	0.64

## 5 CONCLUSION

We presented an incremental learning algorithm for nominal data that allows designing iteratively complex classifiers. It relies on the optimal learning of simple patch functions. Although the resulting classifier is not globally optimal it exhibits good generalization behaviour with small training sets while comparing well to standard techniques with reasonable training set size. Note that up to now we used patches involving single and double features only since the use of higher is computationally more expensive. This is still under investigation.

## 6 REFERENCES

- [1] Hoeting J., Madigan D., Raftery A. and Volinsky C., *Bayesian Model Averaging*, Statistical Science 14, 382-401, 1999.
- [2] Herbrich R., Graepel T., Campbell C., *Bayes Point Machines*, Journal of Machine Learning Research, 1:245-279, 2001.
- [3] Snelson, E. and Ghahramani, Z., *Compact approximations to Bayesian predictive distributions*, ICML 2005.
- [4] Tresp W., Committee machines, in Handbook for Neural Network Signal Processing, Y.H. Hu and J-N. Hwang eds. CRC Press, 2001.
- [5] Witten I.H. and Eibe F., *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufman, 2005.
- [6] Webb G., Boughton J., Wang Z., *Not So Naive Bayes: Aggregating One-Dependence Estimators*, Machine Learning 58(1): 5-24, 2005.

<sup>4</sup> Newman D.J. and al., UCI repository of machine learning databases, <http://www.ics.uci.edu/~mllearn/MLRepository.html>, University of California.