Multi-Layer Boosting for Pattern Recognition

François Fleuret

IDIAP Research Institute,
Centre du Parc,
P.O. Box 592
1920 Martigny,
Switzerland
fleuret@idiap.ch

Abstract

We extend the standard boosting procedure to train a two-layer classifier dedicated to handwritten character recognition. The scheme we propose relies on a hidden layer which extracts feature vectors on a fixed number of points of interest, and an output layer which combines those feature vectors and the point of interest locations into a final classification decision.

Our main contribution is to show that the classical AdaBoost procedure can be extended to train such a multi-layered structure by propagating the error through the output layer. Such an extension allows for the selection of optimal weak learners by minimizing a weighted error, in both the output layer and the hidden layer. We provide experimental results on the MNIST database and compare to a classical unsupervised EM-based feature extraction.

Key words: boosting, multi-layer perceptron, functional gradient descent, convolutional network.

1 Introduction

Most of the efficient image classification methods combine two levels of modelling. At a lower level, feature extraction recodes the input so that local appearance of the image is captured with invariance to translation. At an upper level the features captured locally are combined with a global configuration model. These two steps are present in a form or another in two-layer convolutional networks [9,3], certain decision trees for digit recognition [1], constellations [8,4] or Bayesian models.

We propose in this paper to extend boosting to train the two levels of such a model jointly. Since boosting can be seen as a functional gradient descent, it can be naturally extended to train a multi-layer classifier, each layer of which being a linear combination of simple functionals taking as input the responses of the functionals from the previous layer. This is similar to the extension of gradient descent from a one-layer to a multi-layer perceptron.

The classifier we describe in this paper to illustrate this idea is composed of three stages. The first one is ad hoc and tags points of interest in the image. The second extracts at each of these points a feature vector which is a function of its neighborhood in the image. The third layer combines both the point of interest locations and the feature vectors to obtain a final response. The functionals doing the feature extraction and the functionals doing the final decision are linear combinations of simple weak learners, selected iteratively in a boosting-like manner to minimize an empirical weighted error rate.

2 Multi-Layer Boosting

2.1 Classical AdaBoost

Boosting [2] was proposed initially as a way to improve the power of a family of classifiers by combining a few of them linearly. The usual way to understand it intuitively is as a technique to increasingly put the emphasis on problematic training examples, so that classifiers built successively are more and more dedicated to challenging samples, thus reducing the global error rate. However, boosting can also be seen as a functional gradient descent where the resulting mapping is obtained by constructing iteratively a linear combination of simple functionals to reduce a loss in a greedy fashion.

Precisely, let $\mathcal{X}$ denote the signal space and

- $\{(x_1, y_1), \dots, (x_N, y_N)\} \in (\mathcal{X} \times \{-1, +1\})^N$ a training set,
- $\tilde{g}_1, \dots, \tilde{g}_B, \forall b, \tilde{g}_b : \mathcal{X} \to \{-1,\, 1\}$ a set of weak learners,
- $\forall t \leq T, g_t = \sum_{s=1}^{t} \beta_s \, \tilde{g}_{b_s}$ the mapping obtained after t steps of boosting.

The role of the AdaBoost procedure is to select the sequence of indexes $b_1, \dots, b_T$ and weights $\beta_1, \dots, \beta_T$ so that the sign of $g_T(x_n)$ is a good predictor of the class y_n. As said above, it can be seen as a functional gradient descent [6]: each added weighted weak learner is as a small step in the vector space of functionals, chosen so that an exponential loss L is minimized.

More precisely at any step, given the functional $g_t : \mathcal{X} \to \mathbb{R}$ built so far and an exponential loss function $L(g) = \sum_n \exp(-y_n\, g(x_n))$, the algorithm chooses the weak learner $\tilde{g}_{b(t+1)}$ maximizing

$$\left| \left. \frac{\partial\, L(g_t + \beta\, \tilde{g})}{\partial\, \beta} \right|_{\beta=0} \right| = \left| \sum_n - y_n\, \tilde{g}(x_n)\, \exp(-y_n\, g_t(x_n)) \right|$$

which corresponds to choosing the direction minimizing the loss the most locally. Given that weak learner, β_{t+1} is uniquely defined as

$$\beta_{t+1} = \arg\min_{\beta} L(g_t\ +\ \beta\, \tilde{g}_{b(t+1)})$$

which corresponds to the line search in gradient descent, and has an analytical form in the case of the exponential loss we consider here.

Thus, this is a *functional generalization* of the usual gradient descent. The only restriction is that not all directions in the space of functions are available, but only the ones corresponding to the weak learners.

2.2 Extension to compositions of mappings

We consider now a slightly more complex situation where the classification rule is a composition of sums of weak-learners. In this settings, a first combination of weak learners maps the input space $\mathcal{X}$ into $\mathbb{R}^Q$, and a second combination of weak learners maps $\mathbb{R}^Q$ into $\{-1,\, 1\}$.

We introduce the following

- $\{(x_1, y_1), \ldots, (x_N, y_N)\} \in (\mathcal{X} \times \{-1, +1\})^N$ a training set,
- $\tilde{f}_1, \ldots, \tilde{f}_A, \forall a,\ \tilde{f}_a : \mathcal{X} \to \{-1,\, 1\}^Q$ a set of weak learners for the intermediate recoding,
- $\forall t \leq T, f_t = \sum_{s=1}^{t} \alpha_s \tilde{f}_{a_s}$ the mapping for the intermediate recoding built after t steps of boosting,
- $\tilde{g}_1, \ldots, \tilde{g}_B, \forall b,\ \tilde{g}_b : \mathcal{X} \times \mathbb{R}^Q \to \{-1,\, 1\}$ a set of weak learners for the output mapping,
- $\forall t \leq T,\ g_t = \sum_{s \leq t} \beta_s \tilde{g}_{b_s}$ the output mapping built after t steps of boosting.

Thus, when the training is over, given a signal x, the predicted class corresponds to the sign of $g_T(x, f_T(x))$. We can transpose directly the AdaBoost training rule presented above, which leads to choose the a_t, α_t, b_t and β_{t+1} to minimize

$$L(g_t, f_t) = \sum_n \exp(-y_n\, g_t(x_n, f_t(x_n)))$$

the choice of b_{t+1} is done with a strict AdaBoost rule, that is by choosing for $\tilde{g}_{t+1}$ the $\tilde{g}$ minimizing

$$\left.\frac{\partial\, L(g_t + \beta\, \tilde{g}, f_t)}{\partial\, \beta}\right|_{\beta=0} = \sum_n -y_n\, \tilde{g}(x_n, f_t(x_n))\ \exp(-y_n\, g_t(x_n, f_t(x_n)))$$

which is a weighted error rate of $\tilde{g}$. The choice of β_t is obtained by line minimization which has a closed form, as usual.

The choice of a_{t+1} is however slightly more complex. We again apply the minimization of the local derivative, that is we want to pick for $\tilde{f}_{a_{t+1}}$ the $\tilde{f}$ minimizing

$$\begin{aligned}
&\left.\frac{\partial\, L(g_t, f_t + \alpha \tilde{f})}{\partial\, \alpha}\right|_{\alpha=0} \\
&\quad= \sum_n \left.\frac{\partial\ \exp(-y_n\, g_t(x_n, f_t(x_n) + \alpha \tilde{f}(x_n)))}{\partial\, \alpha}\right|_{\alpha=0} \\
&\quad= \sum_n -y_n \left.\frac{\partial\, g_t(x_n, f_t(x_n) + \alpha \tilde{f}(x_n))}{\partial\, \alpha}\right|_{\alpha=0} \exp(-y_n\, g_t(x_n, f_t(x_n))) \\
&\quad= \sum_n \sum_q -y_n\, \tilde{f}^q(x_n)\, \nabla^q g_t(x_n, f_t(x_n))\, \exp(-y_n\, g_t(x_n, f_t(x_n)))
\end{aligned}$$

where $\tilde{f}^q$ denotes the q-th component of $\tilde{f}$, and $\nabla^q g_t(x)$ the q-th component of the gradient of $\tilde{g}$ in x. Hence, the choice of the weak learners for the intermediate recoding is also the minimization of a weighted error. The coefficient α_{t+1} is chosen with a line-search, which has to be done numerically with a standard bracketing procedure, since no analytical form can be derived anymore.

3 Application to character recognition

3.1 Recognition

Our predictor is very close in spirit to a convolutional multi-layer perceptron [9,3] and consists of a hidden layer extracting the local appearance of the pattern to classify at a few points selected by a difference-of-Gaussians operator [5], and of a second layer combining these local prediction with global geometrical properties to obtain a classification.

More precisely, as described on Figure 1, given an image of size 32×32, the algorithm is:

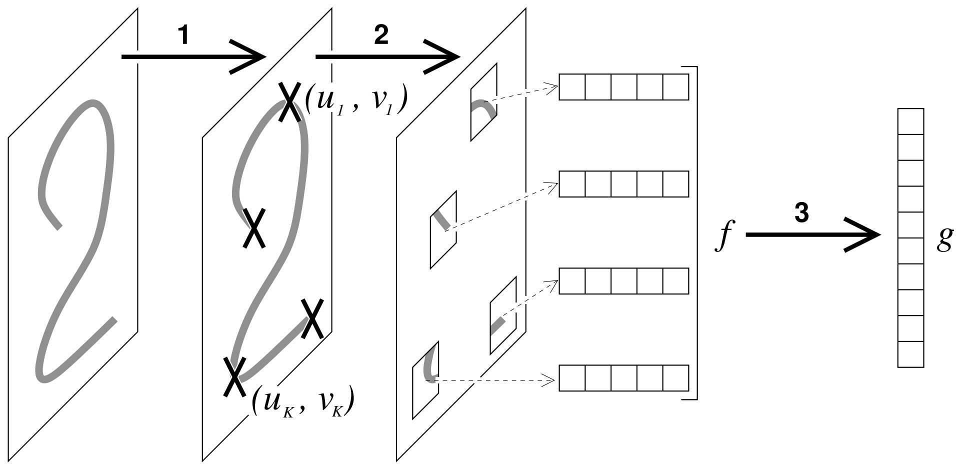


Fig. 1. The classification is composed of three steps: (1) on the input image I a fixed number K of points of interest $(u_1, v_1), \ldots, (u_K, v_K)$ are tagged on the maxima of a difference of Gaussians, (2) a feature vector of size D is computed for each of these points of interest, as a function of the $\Delta \times \Delta$ neighborhood of each points, resulting in a vector $f(I)$ of dimension $D\,K$, (3) one response g_c is computed for each of the C classes as a function of both the locations of the points of interest and the feature vectors.

(1) Select in the 32×32 images a small number K $(= 10)$ of point using the difference of Gaussians maximization criterion [5]. Let

$$(u_1, v_1), \ldots, (u_K, v_K) \in \{0, \ldots, 31\}^{2K}$$

denote their coordinates and x the pair composed of the image itself and the coordinates of these points.

(2) For every selected point k, extract a subimage $I(u_k, v_k)$ of resolution $\Delta \times \Delta$ (with $\Delta = 11$) centered on (u_k, v_k) and compute a feature vector $(\psi_1(I(u_k, v_k)), \ldots, \psi_D(I(u_k, v_k)))$ with $D = 32$, and where the ψ_d are linear combinations of thresholded Haar wavelets.

At the end of that step, the image has been recoded into a list of K points, each one being characterized by its two coordinates (u_k, v_k) in the image plane and a vector of D features. Let $f(x)$ denote the resulting vector of dimension $D\,K$.

(3) Compute as many scores as there are classes

$$(g_1(x, f(x)), \ldots, g_C(x, f(x)))$$

and apply a winner-take-all rule to make a hard decision.

The training algorithm consists of building iteratively and jointly the ψ_d as sums of thresholded Haar wavelets [7] and the $g_1, \dots, g_C$ as sums of smooth disjunctions of the ψ_d. To fit this algorithm in the framework described in § 2.2, we consider the signal x to be both the input image in gray levels and the coordinates of the points of interest $(u_1, v_1), \dots, (u_K, v_K)$, thus $\mathcal{X} = [0,1]^{32\times 32} \times \{0, \dots, 31\}^{2K}$

The number of intermediate features is $Q = D\,K$, that is as many as the number of features per point of interest, time the number of points of interest. Thus, the resulting mapping f_T we wish to obtain at the end of the training has the following form

$$f_T(x) = \left(\underbrace{\psi_1(I(u_1,v_1)), \dots, \psi_D(I(u_1,v_1))}_{D \text{ features on } (u_1,v_1)}, \underbrace{\psi_1(I(u_2,v_2)), \dots, \psi_D(I(u_2,v_2))}_{D \text{ features on } (u_2,v_2)}, \dots \right)$$

and a weak learner $\tilde{f}$ is defined by an index d (corresponding to the ψ_d we are implicitely modifying), a Haar wavelet h and a threshold τ, and is equal to

$$\tilde{f}(x) = \left(\underbrace{0, \dots, 0, \sigma(h(I(u_1,v_1)) - \tau), 0, \dots, 0}_{D \text{ features on } (u_1,v_1)}, \underbrace{0, \dots, 0, \sigma(h(I(u_2,v_2)) - \tau), 0, \dots, 0}_{D \text{ features on } (u_2,v_2)}, \dots \right)$$

where σ is the heavyside function equal to -1 on $\mathbb{R}^-$ and $+1$ on $\mathbb{R}^+$.

Every weak learner $\tilde{g}$ is defined by an index d and a rectangular area $\mathcal{R}$ and is a smooth maximum of the responses of ψ_d on the points (u_k, v_k) localized in $\mathcal{R}$:

$$\tilde{g}(I) = 1 \;-\; 2 \left\{ 1 \;+ \sum_{(u_k,v_k)\in\mathcal{R}} \exp\left(\psi_d(I(u_k,v_k))\right) \right\}^{-1}.$$

Such an expression varies between -1 and 1 and roughly increases with the maximum over the value of ψ_d on points contained in the rectangle $\mathcal{R}$.

We initially break the symmetry between features by populating all the ψ_d with one wavelet picked at random. The training is done by applying the generalized AdaBoost procedure described in §2.2, with the optimization performed at each iteration by sampling at random $1,000$ weak learners and keeping the optimal one according to the current weightings.

In the following, we introduce $\xi_t^c = g_c(x_t)$ and $\zeta_{t,k}^d = \psi_d\left(I_t(u_k^t, v_k^t)\right)$. Also, for the sake of clarity L denotes either the loss as a function of the responses in the hidden layer or in the output layer.

One learning step can be summarized as follow:

```
$\forall t, c,\ \dot{\xi}_t^c \leftarrow \frac{\partial L}{\partial \xi_t^c}$
$\forall t, k, d,\ \dot{\zeta}_{t,k}^d \leftarrow \frac{\partial L}{\partial \zeta_{t,k}^d} = \sum_c \frac{\partial L}{\partial \xi_t^c} \frac{\partial \xi_t^c}{\partial \zeta_{t,k}^d}$
for $d = 1 \dots D$ do
  $(h^*, \tau^*) \leftarrow \arg\max_{h,\tau} \left| \sum_{t,k} \dot{\zeta}_{t,k}^d \, \sigma(h(I_t(u_k^t, v_k^t)) - \tau) \right|$
  $\alpha^* \leftarrow \arg\min_\alpha L(\psi_1, \dots, \psi_d + \alpha\, \sigma(h^* - \tau), \dots, \psi_D)$
  $\psi_d \leftarrow \psi_d + \alpha^* \sigma(h^* - \tau)$
end for
for $c = 1 \dots C$ do
  $\tilde{g}^* \leftarrow \arg\max_{\tilde{g}} \left| \sum_{t,k} \dot{\xi}_t^c \, \tilde{g}(x_t) \right|$
  $\beta^* \leftarrow \arg\min_\beta L(g_1, \dots, g_c + \beta\, \tilde{g}^*, \dots, g_C)$
  $g_c \leftarrow g_c + \beta^* \tilde{g}^*$
end for
```

4 Results

We compare the performance of such a multi-layer boosting approach to a more classical unsupervised clustering to select the features. Also, we study the influence of the number of wavelets in the feature coding on the final performance of the classifier both in term of exponential loss reduction during training and in term of test error rates.

4.1 Baseline method

To measure the improvement of performance due to the joint learning, we use for baseline a very similar two-layer classifier. As for our approach, the hidden layer is composed of D feature extractors, the responses of which are combined into the output layer through disjunctions over rectangular areas. The training algorithm for that output layer is the same as for our approach, and boils down to a classical AdaBoost procedure (see the choice of the b_t and β_t coefficients in §2.2, page 3).

However, the hidden layer for this baseline is trained in a non-supervised manner. This is done by fitting a mixture of Gaussian densities to the training data with a standard EM procedure, hence learning a family of D centroids. By taking into account the variance of classes, such a procedure is slightly more accurate than k-mean, used with success for building such code-books of patches [8].

Hence, we first build a training set containing all the $\Delta \times \Delta$ images centered on the points of interest in the training images. We initialize the EM procedure by picking D samples in this set as the initial class sample averages, and associating each sample of the complete set to the cluster of closest average. From this initialization we apply the classical EM updating rule for mixtures of Gaussians: The expectation and variance of each cluster and the probability of each sample to belong to each class are estimated alternatively. To deal with the high dimension of the space we force the covariance matrix to be proportional to the identity. Figure 5 (a) shows for 10 classes the 10 samples responding the most.

Given an image I of resolution $\Delta \times \Delta$, we can compute a feature vector of dimension D by first computing the distances to the D centroids chosen with the learning procedure described above, and then by thresholding each of these D distance. The choice of thresholds that gave the best results on the test set was the median of each distance on the training examples. Hence, during test there are always roughly half of the features below threshold and half of the feature above threshold.

This baseline has all the strengths of the new method we propose except that it lacks the joint training of the hidden layer: The features encoded in the hidden layer are here trained separately in a non-supervised way.

4.2 Influence of the number of wavelets on the training error

Our first series of experiments tests the performance of the classifier trained with $N = 10,000$ samples, with $T = 100$ weak learners for each class in the output layer and various numbers of wavelets in the hidden layer.

Figure 2 shows the evolution of the exponential loss as a function of the number of weak learners in the output layer, and Figure 3 as a function of the number of wavelets in the hidden layer, for a fixed number of weak learners in the output layer.

Since we equalize the number of clusters between the baseline and the multi-layer boosting, there are no equivalent "capacity parameter" for the former similar to the number of wavelets for the latter. Hence results are always computed after convergence of EM for the baseline.

As expected, the more wavelets in the hidden layer, the more the loss is reduced. Interestingly, 10 wavelets already outperforms the unsupervised EM

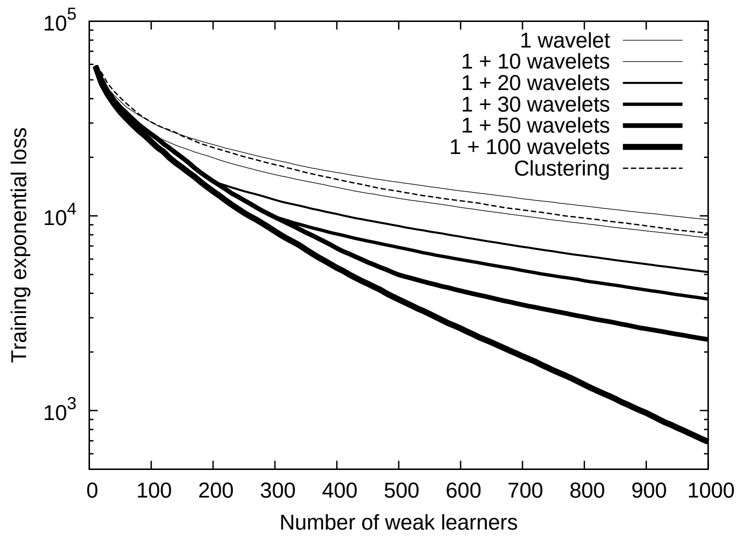


Fig. 2. Exponential loss as a function of the total number of weak learners in the output layer when the wavelets in the hidden layer are added from the first step of learning.

clustering and as shown on Figure 4 the test error rate does not improve when adding more than 50 wavelets.

4.3 Testing error rates

Finally, to compare this approach to the state of the art, we have computed the test error rate with the classifiers used in the experiments presented above, which is built with only $10,000$ samples and 100 weak learners in each of the C functional of the output layers, and also with a classifier trained with the full MNIST database ($60,000$ samples) which combines $1,000$ weak learners in the output layer and 50 wavelets per feature in the hidden layer.

In both case, we improve the classification by trying 3 different rotations in the image plane ($-\frac{\pi}{20}$, 0 and $+\frac{\pi}{20}$) to account for the tilt of the character and averaging the responses of the output layer over the three rounds. The error rates of the classifiers combining only 100 weak learners are given vs. the number of features on Figure 4. The error rate with the classifier trained on the full MNIST database is 1.48% with optimization of the tilt and 1.87% without.

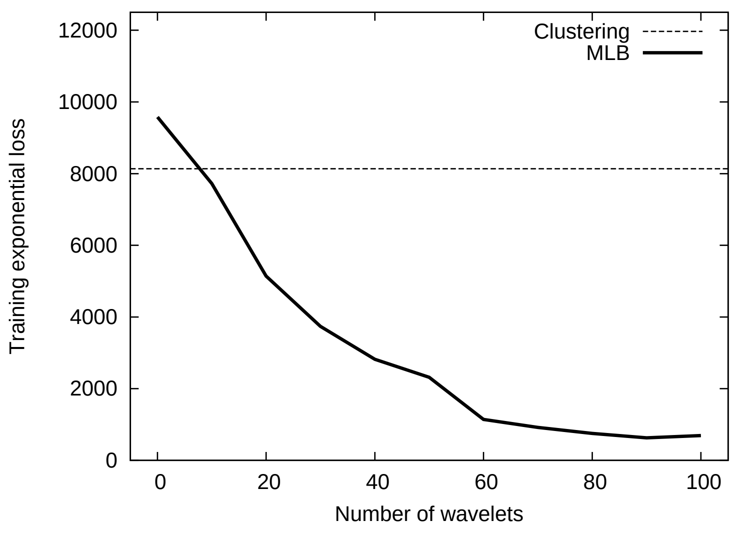


Fig. 3. Exponential loss during training as a function of the number of Haar wavelets used in the hidden layer for the Multi-Layer Boosting (MLB) with 100 weak learners per class in the output layer.

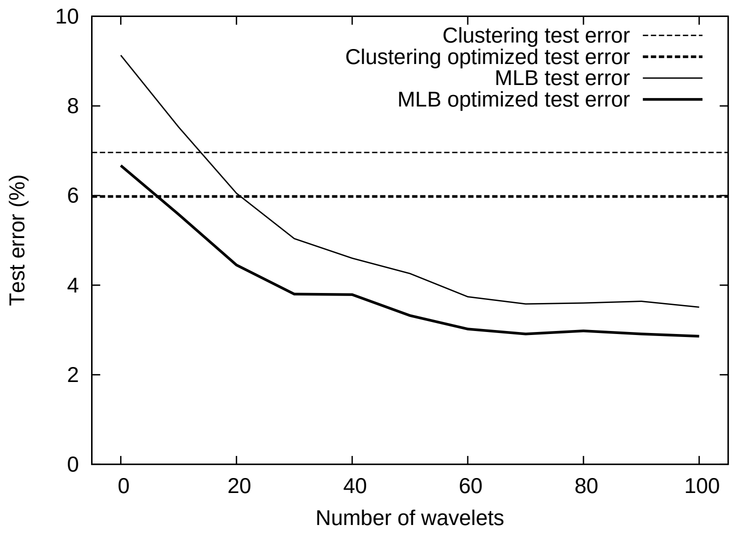
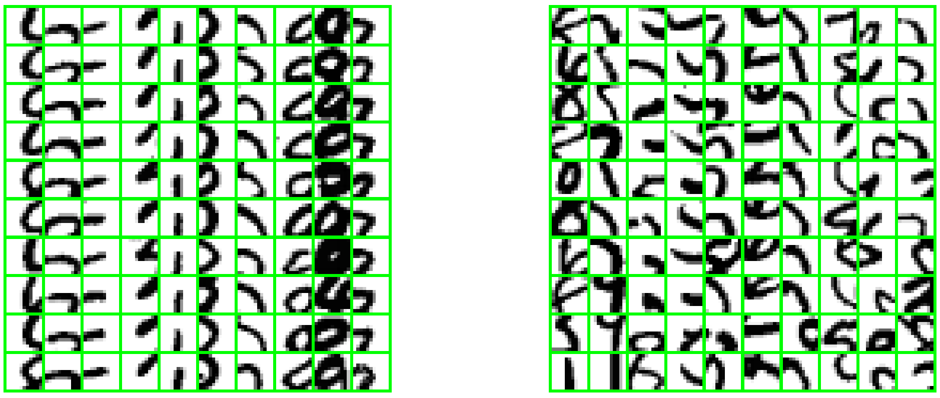


Fig. 4. Test error rates with and without pose optimization as a function of the number of Haar wavelets per feature in the hidden layer. Not shown here is the error rate of 1.48% with $1,000$ weak learners in the output layer, 50 wavelets in the hidden layer and tilt optimization (1.87% without).

5 Conclusion

We have demonstrated how boosting can be extended to a multi-layer setting. The output functionals are trained with a classical AdaBoost while the inner layer is trained by first propagating the derivative of the loss function through the output layer and then using an AdaBoost-like procedure.

Such an architecture is very close to a multi-layer perceptron. Still, it retains all



(a) (b)

Fig. 5. These figure show for each of the $D = 32$ feature component the eight $\Delta \times \Delta$ subimages with the highest score. Figure (a) corresponds to the EM clustering and (b) to the multi-layer boosting with 100 wavelets. As observed for multi-layer perceptrons, the features obtained by supervised learning are less consistent geometrically than those learnt by unsupervised learning.

the *good* properties of boosting, mainly the ability to combine heterogeneous weak learners into a unified scheme. Compared to a MLP, the price to pay is computational: since the constructed classifiers do not live anymore in $\mathbb{R}^n$ as MLPs do, the computational cost increases with the number of combined weak learners.

The computational cost for test is not too high and remains linear with the number of weak learners, as usual for boosted classifiers. That is $O(N_1 + N_2)$ where N_1 is the number of weak learners in the hidden layer and N_2 the number of weak learners in the output layer. On the contrary, the asymptotic training cost increases: since the responses of the output layer weak learners change when the hidden layer is modified, they have to be recomputed at each learning step. Finally, if all the weak learners in the hidden layer are added from the first step, the learning cost turns to be $O(N_1^2 + N_2)$. However experiments showed that a few tens of features in the ψ_ds are sufficient to reach the maximum performance (see Figure 4), which means that this asymptotic cost is not a real issue.

This study demonstrates that multi-layer boosting is doable and provides the expected improvement due to learning jointly the local and the global model of the patterns to classify. Our future work will be focused on refining the weak learners combined in the output layer to imbed invariance to translation or local deformations at reasonable cost.

6 Acknowledgments

This work was supported by the Swiss National Science Foundation under the National Centre of Competence in Research (NCCR) on "Interactive Multimodal Information Management (IM2).

References

[1] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1300–1305, november 1997.

[2] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[3] Y. LeCun and Y. Bengio. Word-level training of a handwritten word recognizer based on convolutional neural networks. In IAPR, editor, *Proceedings of the International Conference on Pattern Recognition*, volume II, pages 88–92, Jerusalem, October 1994. IEEE.

[4] F. Li, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the International Conference on Computer Vision*, volume 2, page 1134, 2003.

[5] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

[6] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *Neural Information Processing Systems*, pages 512–518. MIT Press, 2000.

[7] C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 555–562, 1998.

[8] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proceedings of the European Conference on Computer Vision*, pages 18–32, 2000.

[9] Q. Z. Wu, Y. LeCun, L. D. Jackel, and B. S. Jeng. On-line recognition of limited vocabulary chinese character using multiple convolutional neural networks. In *Proceedings of the 1993 IEEE International Symposium on circuits and systems*, volume 4, pages 2435–2438. IEEE, 1993.