A Tale of Two Classifiers: SNoW vs. SVM in Visual Recognition

Ming-Hsuan Yang[1], Dan Roth[2], and Narendra Ahuja[3]

[1] Honda Fundamental Research Labs, Mountain View, CA 94041
myang@hra.com
[2] Beckman Institute and Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, IL 61801
danr@cs.uiuc.edu
[3] Beckman Institute and Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, Urbana, IL 61801
ahuja@vision.ai.uiuc.edu

Abstract. Numerous statistical learning methods have been developed for visual recognition tasks. Few attempts, however, have been made to address theoretical issues, and in particular, study the suitability of different learning algorithms for visual recognition. Large margin classifiers, such as SNoW and SVM, have recently demonstrated their success in object detection and recognition. In this paper, we present a theoretical account of these two learning approaches, and their suitability to visual recognition. Using tools from computational learning theory, we show that the main difference between the generalization bounds of SVM and SNoW depends on the properties of the data. We argue that learning problems in the visual domain have sparseness characteristics and exhibit them by analyzing data taken from face detection experiments. Experimental results exhibit good generalization and robustness properties of the SNoW-based method, and conform to the theoretical analysis.

1 Introduction

Statistical learning methods have become popular as a tool for addressing a variety of computer vision problems, ranging from object recognition [22,23,18], hand-written digit recognition [14], pedestrian detection [20], to face detection [31,28,30]. There are, however, very few attempts to address theoretical issues and, in particular, study the suitability of different learning algorithms to different vision problems.

The goal of this paper is to present a theoretical account of two learning approaches and their suitability to visual recognition. We use tools from computational learning theory [35,36] to study the properties of two successful learning approaches. The algorithms are evaluated on a visual recognition problems, face detection, and the theoretical generalization properties, along with our analysis of the data are used to explain the prediction performance and discuss the suitability of the approaches to visual learning problems. The learning approaches

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2353, pp. 685–699, 2002.

we study are Support Vector Machines (SVMs) [36] and the SNoW learning architecture [26,6]. Both have been studied extensively recently and have shown good empirical performance on several visual learning problems [23,20,38,37]. We study both generalization and computational efficiency issues and derive conclusions that are relevant to further use of these learning methods in visual recognition tasks.

The learning paradigm applied to most visual recognition problems is that of supervised learning. In this case, each image example is represented as a feature vector and a label, typically a binary one, is added to represent whether this example represents an image in the target class (e.g., contains a face) or not. A variety of feature representations have been used in recent works that apply learning methods to visual recognition tasks. Examples include [14] which applied convolutional network with simple local features to hand-written digit recognition, [30] which utilized naive Bayes classifiers over local features in a face detection task, Viola et al. that generated a very large set of selective features use them in texture recognition, object recognition and hand-written digit recognition [8,24,33], and Geman and Amit [2] that used edges and conjunction of edges as local features of images and use those for object recognition. Local feature analysis using Principal Component Analysis and Independent Component Analysis have been applied to face recognition [21] and facial expression recognition with success [9]. Recently, Poggio et al. [20] [18] utilized wavelets and local features (which they call components) with SVMs for object detection. Their experiments on face, car, and pedestrian detection demonstrated good results.

In visual learning situations, the number of features that could potentially affect each decision is very large but, typically, only a small number of them is actually relevant to a decision. Beyond correctness, a realistic learning approach needs therefore to be feature-efficient [15] in that its learning complexity (number of examples required for convergence) depends on the number of relevant features and not the global number of features in the domain. Equivalently, this can be phrased as the dependence of the generalization quality on the number of examples observed. A features efficient algorithm would be able to generalize well already after observing a relatively small number of examples [13].

This paper argues and exhibits, in a limited context, that the SNoW learning architecture, which is based on a feature efficient algorithm, is a suitable architecture for learning in the visual domain. Motivated by the recent success of this approach on several visual recognition tasks we suggest that recent advances in computational learning theory can be used to explain these results; moreover, we show that a careful analysis of the data may help to determine the suitability of a given learning algorithm to a task, and suggest ways for using a learning approach so that its advantages can be exploited.

To make things concrete we choose a specific data set for the face detection task, and use it for a detailed study of SNoW, as compared to SVM. SNoW and SVM, presented in Section 2, are representatives of two different classes of linear classifiers; the first is based on Winnow, a multiplicative update rule

algorithm [15], and the second on perceptron, an additive update rule (see [10, 7] for the representation of SVM as a batch version of perceptron).

Several theoretical results have suggested that these two approaches have incomparable generalization performance that depend on well defined properties of the domain and the target concept. We study these properties and conclude, within a limited context, that the face detection data suggests that the SNoW based approach should have advantages in terms of generalization in Section 3.

In addition to generalization, the two learning approaches can also be measured in terms of efficiency for feature representation. This is important especially when one wants to "blow" up the feature space in order to increase the expressivity of the features and allow a linear classifier to discriminate faces from non-faces, or to discriminate between objects. In this case, we argue that the SVM approach is advantageous. We argue that in order to fully exploit the nice generalization properties of SNoW, images should be represented using features that give rise to a fairly small number of *active* features in each image. That is an attempt should be made to use representations that are not pixel based, but rather based on sparser phenomenon in the image, such as edges, conjunctions of those or other types of features. We show some preliminary results that support this and suggest several directions for future work.

The paper is organized as follows. We discuss the task of visual recognition as a classification problem and present the two learning approaches studied here in Section 2. In Section 3 we present generalization bounds for the algorithms and study them in the context of a specific data set. We then move on to comment on efficiency issues, and conclude with future work in Section 5.

2 Visual Learning Framework

Most efficient learning methods known today, including many probabilistic classifiers, make use of a linear decision surface over the feature space. Among these methods we focus here on SVMs and SNoW which have demonstrated good empirical results in vision and natural language processing problems [23,20,26, 27]. SNoW and SVM, are representatives of two different classes of linear classifiers/regressors. SNoW is based on Winnow, a multiplicative update rule algorithm [15,13]; SVMs are based on perceptron [25], an additive update rule. Although SVMs can also be developed independently of the relation to perceptron, for the sake of our theoretical analysis viewing them as a large margin perceptron [10,7] is important. Moreover, recent results [11] have shown that the generalization properties of SVMs are dominated by those of large margin perceptron, and therefore it is sufficient here to study those.

2.1 The SNoW Learning Architecture

The SNoW (Sparse Network of Winnows) learning architecture is a sparse network of linear units over a common pre-defined or incrementally learned feature space [6]. Nodes in the input layer of the network typically represent relations over the input instance and are being used as the input features. Each linear

unit is called a *target node* and represents a concept of interest over the input. In the application described here, target nodes could represent an object in terms features extracted from the 2D image input, a face, or a non-face. In the current presentation we assume that all features are binary (in $\{0, 1\}$), although SNoW can take real numbers as input. An input instance is mapped into a set of features which are active (with feature value 1) in it; this variable size representation is presented to the input layer of SNoW and propagates to the target nodes. Target nodes are linked via weighted edges to (some of) the input features.

Let $\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of features that are active in an example and are linked to the target node t. Then the linear unit corresponding to t is *active* iff

$$\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t,$$

where w_i^t is the weight on the edge connecting the ith feature to the target node t, and θ_t is the threshold for the target node t.

Each SNoW *unit* may include a collection of subnetworks, one for each of the target relations but all using the same feature space. A given example is treated autonomously by each target unit; an example labeled t may be treated as a positive example by the t unit and as a negative example by the rest of the target nodes in its subnetwork. At decision time, a prediction for each subnetwork is derived using a winner-take-all policy. In this way, SNoW may be viewed as a multi-class predictor. In the application described here, we may have one unit with target subnetworks for all the target objects or we may define different units, each with two competing target objects.

SNoW's learning policy is on-line and mistake-driven; several update rules can be used within SNoW, but here we concentrate on the one which is a variant of Littlestone's Winnow update rule [15], a multiplicative update rule that we tailored to the situation in which the set of input features is not known a priori, as in the infinite attribute model [4]. This mechanism is implemented via the sparse architecture of SNoW. That is, (1) input features are allocated in a data driven way – an input node for the feature i is allocated only if the feature i was active in any input sentence and (2) a link (i.e., a non-zero weight) exists between a target node t and a feature i if and only if i was active in an example labeled t.

One of the important properties of the sparse architecture is that the complexity of processing an example depends only on the number of features active in it, n_a, and is independent of the total number of features, n_t, observed over the life time of the system. This is important in domains in which the total number of features is very large, but only a small number of them is active in each example.

The Winnow update rule has, in addition to the threshold θ_t at the target t, two update parameters: a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. These are being used to update the current representation of the target t (the set of weights w_i^t) only when a mistake in prediction is made. Let

$\mathcal{A}_t = \{i_1, \ldots, i_m\}$ be the set of active features that are linked to the target node t. If the algorithm predicts 0 (that is, $\sum_{i \in \mathcal{A}_t} w_i^t \leq \theta_t$) and the received label is 1, the active weights in the current example are *promoted* in a multiplicative fashion:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \alpha \cdot w_i^t.$$

If the algorithm predicts 1 ($\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t$) and the received label is 0, the active weights in the current example are *demoted*:

$$\forall i \in \mathcal{A}_t, w_i^t \leftarrow \beta \cdot w_i^t.$$

All other weights are unchanged.

As will be clear below, the key feature of the Winnow update rule is that the number of examples required to learn a linear function grows linearly with the number n_r of *relevant* features and only logarithmically with the total number of features. This property seems crucial in domains in which the number of potential features is vast, but a relatively small number of them is relevant. Moreover, in the sparse model, the number of examples required before converging to a linear separator that separates the data (provided it exists) scales with $O(n_r \log n_a)$. Winnow is known to learn efficiently any linear function (in general cases efficiency scales with the margin) and to be robust in the presence of various kinds of noise and in cases where no linear function can make perfect classifications, while still maintaining its abovementioned dependence on the number of total and relevant attributes [16,13].

2.2 Large Margin Perceptron and SVMs

In this section we briefly present perceptron and SVM; the presentation concentrates on the linearly separable case, although it can be extended to the more general case [36].

The perceptron also maintains a weight vector w and, given an input vector x^i, predicts that x^i is a positive example iff $w \cdot x^i > \theta$. Like Winnow, the perceptron's update rule is also an on-line and mistake driven, and the only difference between them is that the weight update rule of perceptron is additive. That is, if the linear function misclassified an input training vector x^i with true label y^i (here we assume for notational convenience that $y^i \in \{-1, +1\}$) then we update each component i of the weight vector w by:

$$w_j \leftarrow w_j + \eta x^i y^i,$$

where η is the learning rate parameter.

Like Winnow, the Perceptron is also known to learn every linear function, and in the general case, the number of mistakes required before it converge to a hyperplane that separates the data depends also on the margin in the data, that is, on $\max x^i \cdot y^i$, where $y^i \in \{-1, +1\}$ is the true label of the example x^i.

Linear separability is a rather strict condition. One way to make methods more powerful is to add dimensions of features to the input space. Usually, if we add enough new features, we can make the data linearly separable; if the

separation is sufficiently good, then the expected generalization error will be small, provided that we do not increase the complexity of instances too much by this transformation. In other words, we need to find a nonlinear function to map the training examples from the input space to a higher dimensional feature space, i.e.,

$$\phi : x \in \mathbb{R}^N \rightarrow \phi(x) \in \mathbb{R}^F \tag{1}$$

where $F \gg N$. The decision surface of a Perceptron in the feature space becomes:

$$f(x) = \langle w \cdot \phi(x) \rangle + b = \sum_{i=1}^{M} \eta^i y^i \phi(x^i) \cdot \phi(x) + b \tag{2}$$

However, from a computational point of view this could be prohibitively hard. This problem can sometimes be solved by the kernel trick. Aizerman, Braverman and Rozonoer suggested this method and showed that it can be combined with Perceptron [1]. Boser, Guyon and Vapnik applied the same trick to extend nonlinear SVMs [5].

A kernel function, $K(x, z)$, is a function of two variables which can be represented as an inner product, i.e., $\phi(x) \cdot \phi(z)$, for some function $\phi : \mathbb{R}^N \rightarrow \mathbb{R}^F$ and $F \gg N$. In other words, we can compute $K(x, z)$ if we can compute $\phi(z)$ and $\phi(z)$. Thus, we can rewrite the Perceptron in a feature space as:

$$f(x) = \sum_{i=1}^{M} \eta^i y^i \langle \phi(x^i) \cdot \phi(x) \rangle + b = \sum_{i=1}^{M} \eta^i y^i K(x^i, x) + b. \tag{3}$$

Consequently with kernel functions, we can find a linear decision surface in a high dimensional space without explicitly projecting the training examples. Furthermore, the constructed linear decision surface in the projected high dimensional space is equivalent to a nonlinear decision surface in the input space, which is particularly useful for the case where the patterns are not linearly separable in the input space. As will be clear later, the kernel trick serves to aid efficiency, in case there is a need to work in a higher dimensional space; however, the generalization properties, in general, depend on the effective, high dimensional, feature space in which the linear classifier is determined.

SVMs, or batch large margin classifiers can be derived directly from a large margin version of perceptron (which we do not describe here; see e.g., [39]) using a standard way to convert the on-line algorithm to a batch algorithm. This is done in order to convert the mistake bounds that are typically derived for on-line algorithms to generalization bounds that are of more interest (e.g., [10]). However, for completeness, we briefly explain the original, direct derivation of SVMs. SVMs can be derived directly from the following inductive inference. Given a labeled set of training samples, an SVM finds the optimal hyperplane that correctly separates the data points while maximizing the distance of either class from the hyperplane (maximizing the margin). Vapnik showed that maximizing the margin is equivalent to minimizing the VC dimension and thus

yields best generalization results [36]. Computing the best hyperplane is posed as a constrained optimization problem and solved using quadratic programming techniques. The optimal hyperplane is defined by

$$\min \frac{1}{2}w^2, \quad \text{subject to} \quad y^i(w^T x^i + b) \geq 1 \quad \forall i = 1, \ldots, M$$

where b is a bias term computed from the margin.

Finally we note that although large margin perceptron and SVMs are very related, it turns out that the generalization bounds of the large margin perceptron are slightly better than those of SVMs [11,3]. Therefore, we will use those in our analysis in Section 3.

Although these are worst case bounds, they have already be shown to be quite representative in some experiments using synthetic data [13], so we can use them to guide our understanding.

3 Generalization and Efficiency

There are two issues that we need to consider when we compare two learning algorithms: generalization and efficiency. Generalization bounds are derived in order to estimate, given the performance on the training data, what will be the performance on previously unseen examples. Beyond correctness, a realistic learning approach needs therefore to be feature-efficient [15] in that its learning complexity (number of examples required for convergence) depends on the number of relevant features and not the global number of features in the domain.

The efficiency issues (for feature representation) have been discussed in Section 2. We compare SNoW and SVM in terms of generalization error bounds in this section.

3.1 Generalization Error Bounds

Learning systems use training data in order to generate a hypothesis, but the key performance measure one cares about is actually how well they will perform on previously unseen examples. Generalization bounds are derived in order to estimate, given the performance on the training data, what will be the performance on previously unseen examples. The assumption underlying the derivation of generalization bounds is the basic assumption of the PAC learning theory [35], that the test data is sampled from the same (unknown) distribution from which the training data was sampled.

In the following we preset two theorems, one describing the generalization error bound of large margin classifiers (e.g., SVMs) and the other for the multiplicative update algorithm (e.g., Winnow). The first one is a variant of Theorem 4.19 in [7,39]:

Theorem 1. *If the data is L_2 norm bounded as $||x||_2 \leq b$, then consider the family Γ of hyperplanes w such that $||w||_2 \leq a$. Denote by $E_a(w)$ the misclassification error of w with the true distribution. Then there is a constant C such*

that for any $\gamma > 0$, with probability $1 - \eta$ over n random samples, any $w \in \Gamma$ satisfies:

$$E_a(w) \leq \frac{k_\gamma}{n} + \sqrt{\frac{C}{\gamma^2 n} a^2 b^2 \ln(\frac{nab}{\gamma} + 2) + \ln \frac{1}{\eta}}$$

where $k_\gamma = |\{i : w^T x^i y^i < \gamma\}|$ is the number of samples with margin less than γ.

Similarly we present a generalization bound for Winnow family of algorithms (e.g., SNoW). See also [13,39] for some more detail.

Theorem 2. *If the data is L_∞ norm bounded as $||x||_\infty \leq b$, then consider the family Γ of hyperplanes w such that $||w||_1 \leq a$ and $\sum_j w_j \ln(\frac{w_j ||\mu||_1}{\mu_j ||w||_1}) \leq c$. Denote by $E_m(w)$ the misclassification error of w with the true distribution. Then there is a constant C such that for any $\gamma > 0$, with probability $1 - \eta$ over n random samples, any $w \in \Gamma$ satisfies:*

$$E_m(w) \leq \frac{k_\gamma}{n} + \sqrt{\frac{C}{\gamma^2 n} b^2 (a^2 + ac) \ln(\frac{nab}{\gamma} + 2) + \ln \frac{1}{\eta}}$$

where μ denotes an initial weight vector and $k_\gamma = |\{i : w^T x^i y^i < \gamma\}|$ is the number of samples with margin less than γ.

In order to understand the relative merits of the algorithms, a closer look at the above bounds shows that, modulo some unimportant terms, the error bounds E_a and E_m for the additive algorithms and the multiplicative algorithms scale with:

$$E_a(w) \approx ||w||_2^2 \max_i ||x^i||_2^2,$$

and

$$E_m(w) \approx 2 \ln 2n ||w||_1^2 \max_i ||x^i||_\infty^2$$

where w is the target hyperplane.

From the theorems, the main difference between the error bounds of SVM and SNoW is the properties of data. If the data is L_2 norm bounded and there is a small L_2 norm hyperplane, then SVM is suitable for the problem. On the other hand, Winnow is suitable for a problem where the data is L_∞ norm bounded and there is a small L_1 norm hyperplane. In visual recognition tasks with pixel-based feature representation, the hyperplane function is usually sparse since the image dimensionality is usually high and most pixels do not contribute to the construction of the hyperplane (i.e., irrelevant to the learning task and has zero weight term in the hyperplane). Consequently, the hyperplane usually has small L_1 and L_2 norms. Theoretical analysis indicates that the advantage of the Winnow family of algorithms (e.g., SNoW) over Perceptron family of algorithms (e.g., SVM) requires the data to have small L_∞ norm but large L_2 norm. Numerical experiments in [13] have confirmed the above claims and demonstrated the generalization bounds are quite tight.

4 Empirical Study

For concreteness, we choose a specific data set for the face detection task, and use it for a detailed study of SNoW, as compared to SVM. The empirical results are discussed in this section.

4.1 Experiment I: Generalization

Our experiments on object recognition and face detection demonstrate that SNoW performs well or outperforms SVM in object recognition and face detection [38,37]. To better understand why and when SNoW achieves such performance, and compare the empirical results with the theorems, we perform more experiments on face detection. The training set consists of 6,977 images (2,429 faces and 4,548 non-faces), and the test set consists of 24,045 images (472 faces and 23,573 non-faces). Our training and test sets are similar to the one used in [12] which also shows that SVMs with the feature representation of normalized intensity values perform better than the ones with Harr wavelet and gradient representations. In our experiment, each image is normalized to 20×20 pixels and processed with histogram equalization and quantization (50 rather than 256 scales). Figure 1 shows some face images in the training and test sets.

Fig. 1. Sample face images: each image is normalized to 20×20 pixels with histogram equalization.

We use the the quantized intensity values as image features to train SVMs with linear kernel. For SNoW, we also use quantized intensity values as features of images, which we call linear features. Let the pixel at (x, y) of an image with width w and height h have intensity value $I(x, y)$ $(0 \leq I(x, y) \leq 49)$. This information is encoded as a feature whose index is $50 \times (y \times w + x) + I(x, y)$. This representation ensures that different points in the {`position` $\times$ `intensity`} space are mapped to different features. (That is, the feature indexed $50 \times (y \times w + x) + I(x, y)$ is *active* if and only if the intensity in position (x, y) is $I(x, y)$.) In [32], a similar binary representation scheme is adopted to train restricted Boltzmann machines for face recognition which achieves good results. Note that although the number of potential features in our representation is $20,000$ (400×50), only 400 of those are active (present) in each example, and it is plausible that many features will never be active. Since the algorithm's complexity depends on the number of active features in an example, rather than the total number of features, the sparseness also ensures efficiency. Figure 2

shows the linear features learned by the SNoW where the features associated with larger weights are represented with brighter intensity values. Note that the most salient features in faces are around the eye, nose, and the face contour areas, which correspond well to several psychological studies on face recognition.

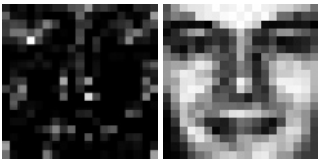

Fig. 2. (Left): Linear features learned by the SNoW: Features associated with larger weights are represented by brighter intensities. (Right): A sample face in the training set.

For the baseline study where SNoW and SVM have the same feature representation, i.e., quantized intensity values, SNoW outperforms linear SVM (which is often used in real time applications) as shown by the ROC curves (the lower two curves) in Figure 3 (we will discuss the upper two curves in Section 4.2).

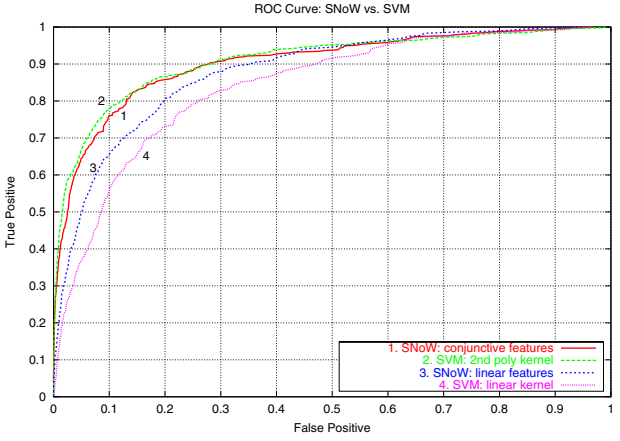

Fig. 3. ROC curves: SNoW vs. SVM.

At first glance of the learning curves, the performance of SNoW and SVM may seem to be inferior to other reported results in the literature [31,28,19,29, 38]. However, note that Figure 3 shows only the raw output from a classifier while most face detection methods use various heuristics (such as thresholding, overlap elimination, and arbitration among multiple classifiers, as discussed in [28]) to increase the final detection rates. Furthermore, the training set size used in the reported literature is much larger than the one in this paper. Here we use

a specific image set to evaluate the performance of the raw outputs of SNoW and SVM in details, and the results are incomparable to the ones reported in the literature.

Table 1. L_2 and L_∞ norms of data in the experiments

	Linear Features	Conjunctive Features
SNoW	563	2.55×10^9
SVM	55	1.16×10^9

For visual pattern recognition, most data dimensions are not useful as demonstrated in the Eigenface [34] approach and others. Many studies have also shown that the target hyperplane function in visual pattern recognition is usually sparse. Consequently, the target hyperplane has a relatively small L_2 norm and relatively small L_1 norm. Under such situations, the Perceptron does not have any theoretical advantage over Winnow. Thus it is not surprising to see that the Winnow family and the Perceptron family of algorithms perform equally well in several applications [38,37].

For the experiments with linear features (i.e., quantized intensity values), the L_2 norm is on the average 10.2 times larger than the L_∞ norm as shown in Table 1. The number of active features in the final hyperplane of SNoW is very sparse, i.e., only 1.6% of all possible features. The number of support vectors is also sparse, i.e., only 5% of all the training examples. The empirical results show that SNoW outperforms SVM (shown in ROC curves in Figure 3) and match the predictions of the theorems well.

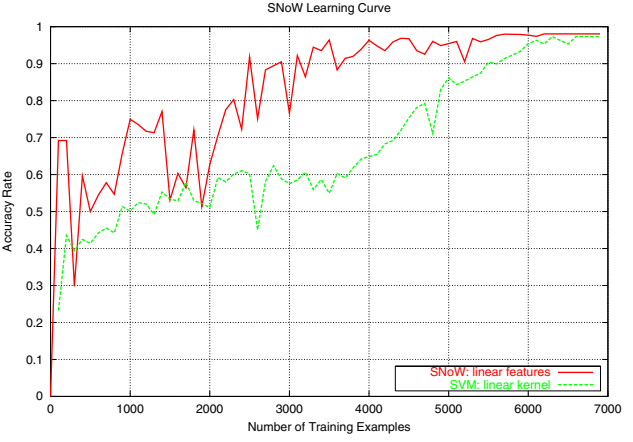

Fig. 4. Learning curves: SNoW vs. SVM.

Figure 4 shows the on-line performance of SNoW (with linear features) and SVM (with linear kernel) in which we evaluate the system with the test set after

we train a classifier with p examples ($100 \leq p \leq 6,900$). The results, averaged over 20 runs with random sampling of examples, demonstrate that SNoW is able to learn the decision hyperplane rather efficiently. On the other hand, a SVM usually needs to go though all the training examples in order to extract a minimum set of support vectors which maximizes the margins of an optimal decision hyperplane.

Although SNoW is an on-line mistake-bound algorithm, it is possible that one can improve SNoW's performance by presenting the same example several times. Nevertheless, SNoW usually converges fast. Figure 5 show the performance of SNoW (with linear features) after only one iteration is close to the best results achieved by only two iterations.

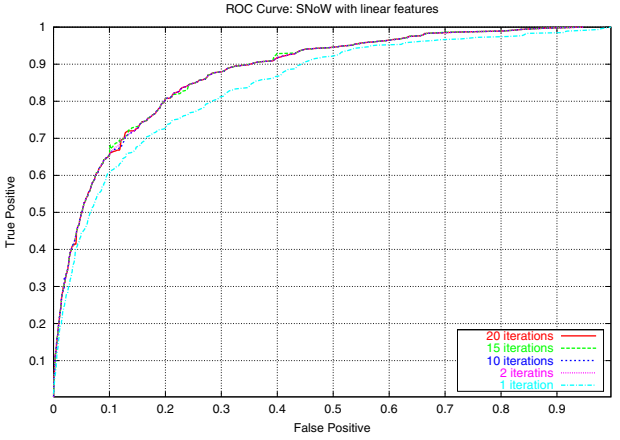

Fig. 5. SNoW convergence rate.

4.2 Experiment II: Efficiency

Since the features in the SVM with polynomial (or Gaussian) kernels are more expressive than the linear features, we choose to use conjunctive features to capture local information of image patterns. The idea is, in spirit, similar to n-gram used in natural language understanding. For each pixel, we represent the conjunction of intensity values of m pixels within a window of $w \times w$ pixels as a new feature value and use them as feature vectors. Let $I(p, q)$ and $I(r, s)$ denote the intensity values of pixels at (p, q) and (r, s) ($p \leq r$ and $q \leq s$), we use a feature value to encode the occurrence of the $I(p, q)$, $I(r, s)$ at distance of $(r - p) \times w + (s - q)$. Each feature value is then mapped to a binary feature using the method discussed in Section 4.1. See also [27] for more detail. A recent study also shows that low order conjunctive features support unambiguous perception in object representation (i.e., conjunctive feature representation is expressive enough to encode the objects) and lead to excellent empirical results for recognition tasks [17].

To make sure that the combined computational requirement of SNoW (computational loads of features in training) does not outweigh the one of SVM, we choose to use a window of 4×4 pixels and conjunctions of 2 pixels. Figure 3 shows the ROC curves of SVM with second order polynomial kernel and SNoW with conjunctive features. Although SVM performs slightly better than SNoW, we think that SNoW can perform as well as SVM if the feature representation is as expressive as the one in SVM with polynomial kernel. We will discuss these issues in Section 5.

The L_2 norm of the local conjunctive features (generated by 4×4 window) is only 2.2 times larger than the L_∞ norm as shown in Table 1. In this case, SVM performs slightly better than SNoW. The results conform to predictions of the theoretical analysis which indicates that the advantage of SNoW over SVM requires the data to have large L_2 norm but small L_∞ norm.

5 Concluding Remarks

This paper proposes theoretical arguments that suggests that the SNoW-based learning framework has important advantages for visual recognition tasks. Given good experimental results with SNoW on several visual recognition tasks such as face detection and object recognition, the main contribution of this work is in providing an explanation for this phenomena - by giving a theoretical analysis and validating it with real world data - and providing ways for constructing good feature representations for visual learning tasks.

We have shown that SNoW, being based on a multiplicative update algorithm, has nice generalization properties compared to other learning algorithms. On the other hand, algorithms that are based on additive update algorithms, like perceptrons and SVM, have nice computational properties, stemming from the ability to use the kernel trick to avoid computation with data of very high dimensionality. We then argue that SNoW, with its ability to handle variable size examples, does not suffer from the dimensionality of the data but only depends on the presence of few active features in each example. Moving to a sparse representation of images, (e.g., edges, conjunctions of more than two linear features or others local features) would allow one to enjoy the best of both worlds - a good generalization performance along with computational efficiency. We believe this is an important direction for future research.

Acknowledgements. Ming-Hsuan Yang was supported by Ray Ozzie Fewllowship and Office of Naval Research grant N00014-00-1-009. Dan Roth was supported in part by National Science Foundation grants IIS-9984168 and IIS-0085836. Narendra Ahuja was supported in part by Office of Office of Naval Research grant N00014-00-1-009.

References

1. M. Aizerman, E. Braverman, and L. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

2. Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, 1999.
3. S. Ben-David and H. U. Simon. Efficient learning of linear perceptron. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 189–195. MIT Press, 2001.
4. A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, 1992.
5. B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.
6. A. Carleson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, 1999.
7. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based learning methods*. Cambridge University Press, 2000.
8. J. De Bonet and P. Viola. Texture recognition using a non-parametric multi-scale statistical model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 641–647, 1998.
9. G. Donato, M. S. Bartlett, J. C. Hager, P. Ekman, and T. J. Sejnowski. Classifying facial actions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):974–989, 2000.
10. Y. Freund and R. Schapire. Large margin classification using the perceptron. *Machine Learning*, 37(3):277–296, 1999.
11. T. Graepel, R. Herbrich, and R. C. Williamson. From margin to sparsity. In *Advances in Neural Information Processing Systems 13*, pages 210–216. MIT Press, 2001.
12. B. Heisele, T. Poggio, and M. Pontil. Face detection in still gray images. Technical Report AI Memo 1687, MIT AI Lab, 2000.
13. J. Kivinen, M. K. Warmuth, and P. Auer. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bound when few input variables are relevant. *Artificial Intelligence*, 1-2:325–343, 1997.
14. Y. Le Cun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pages 53–60, 1995.
15. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
16. N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using winnow. In *Proceedings of the fourth Annual Workshop on Computational Learning Theory*, pages 147–156, 1991.
17. B. W. Mel and J. Fiser. Minimizing binding errors using learned conjunctive features. *Neural Computation*, 12:247–278, 2000.
18. A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.
19. E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.
20. C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

21. P. Penev and J. Atick. Local feature analysis: A general statistical theory for object representation. *Network: Computation in Neural Systems*, 7(3):477–500, 1996.
22. T. Poggio and S. Edelman. A network that learns to recognize 3D objects. *Nature*, 343:263–266, 1990.
23. M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.
24. T. Rikert, M. Jones, and P. Viola. A cluster-based statistical model for object detection. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1046–1053, 1999.
25. F. Rosenblatt. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.
26. D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 806–813, 1998.
27. D. Roth, M.-H. Yang, and N. Ahuja. Learning to recognize objects. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 724–731, 2000.
28. H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.
29. H. Schneiderman. *A Statistical Approach to 3D Object Detection Applied to Faces and Cars*. PhD thesis, Carnegie Mellon University, 2000.
30. H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 45–51, 1998.
31. K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.
32. Y. W. Teh and G. E. Hinton. Rate-coded restricted Boltzmann machines for face recognition. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 908–914. MIT Press, 2001.
33. K. Tieu and P. Viola. Boosting image retrieval. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 228–235, 2000.
34. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
35. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, Nov. 1984.
36. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
37. M.-H. Yang, D. Roth, and N. Ahuja. Learning to recognize 3D objects with SNoW. In *Proceedings of European Conference on Computer Vision*, volume 1, pages 439–454, 2000.
38. M.-H. Yang, D. Roth, and N. Ahuja. A SNoW-based face detector. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances of Neural Information Processing Systems*, pages 855–861. MIT Press, 2000.
39. T. Zhang. Some theoretical results concerning the convergence of compositions of regularized linear functions. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 370–376. MIT Press, 2000.