

Linear concepts and hidden variables*

Adam J. Grove[†]

44 Murray Place
Princeton NJ 08540
grove@pobox.com

Dan Roth[‡]

Department of Computer Science
University of Illinois at Urbana-Champaign
danr@cs.uiuc.edu

Abstract

We study a learning problem which allows for a “fair” comparison between unsupervised learning methods—probabilistic model construction, and more traditional algorithms that directly learn a classification. The merits of each approach are intuitively clear: inducing a model is more expensive computationally, but may support a wider range of predictions. Its performance, however, will depend on how well the postulated probabilistic model fits that data. To compare the paradigms we consider a model which postulates a single binary-valued *hidden variable* on which all other attributes depend. In this model, finding the most likely value of any one variable (given known values for the others) reduces to testing a linear function of the observed values. We learn the model with two techniques: the standard EM algorithm, and a new algorithm we develop based on covariances. We compare these, in a controlled fashion, against an algorithm (a version of *Winnnow*) that attempts to find a good linear classifier directly. Our conclusions help delimit the fragility of using a model that is even “slightly” simpler than the distribution actually generating the data, vs. the relative robustness of directly searching for a good predictor.

1 Introduction

In the archetypical *unsupervised learning* problem a collection of *unlabeled* learning examples $\{\tilde{x} = (x_0, x_1, \dots, x_n)\}$ is given and a model is learned that fits the data. As an alternative, one may try to learn each of the x_i s as a function of the other bits in the examples, adopting a *supervised learning* model in which the bit x_i is the label. In many cases, the learned model itself is of little interest, and one is more interested in predictions done using this model on future, partially specified, examples. In these cases, a direct comparison between the two learning paradigms can be performed. In this paper we study an instantiation of this problem that allows for an interesting and fair comparison of the paradigms.

We consider the classic task of predicting a *binary* (0/1) target variable x_0 , based on the values of some n other binary variables $x_1 \dots x_n$. We can distinguish between two styles of learning approaches for such tasks. *Parametric* algorithms postulate some form of probabilistic model underlying the data, and try to fit the model’s parameters. To classify an example we can compute the conditional probability distribution for x_0 given the values of the known variables, and then predict the most probable value. *Non-parametric* algorithms do not assume that the training data

*Preliminary version of this paper appeared in NIPS-10.

[†]This work was done while this author was with NECI, Princeton, NJ.

[‡]Supported by NSF grant IIS-9801638. Part of this work was done while visiting Harvard University, supported by ONR grant N00014-96-1-0550.

has a particular form. They instead search directly in the space of possible classification functions, attempting to find one with small error on the training set of examples. While the generalization properties of the parametric methods rely on the ability to approximate the underlying probability distribution, the non-parametric methods make no attempt to estimate this distribution; their generalization relies on approximately fitting the training data and on that the training and test data are sampled from the same distribution [Val84, Vap95].

An important advantage of parametric approaches is that the induced model can be used to support a wide range of inferences, aside from the specified classification task. On the other hand, to postulate a particular form of probabilistic model can be a very strong assumption. So it is important to understand how robust such methods are when the real world deviates from the assumed model.

In this paper, we report on some experiments that test this issue. We consider a standard and rather restricted model involving an unobserved explanatory variable, which still however retains considerable interest. Specifically, we consider the case of $n + 1$ *conditionally independent attributes* x_i together with a single *unobserved* variable z , also assumed to be binary valued, on which the x_i depend (henceforth, the binary CIA model).

Given a fixed value of z , the value of each x_i is *independently* generated: x_i is 1 with some probability p_i^z , otherwise 0. Note that the probabilities depend on z . There are thus $2n + 3$ parameters: the $2(n + 1)$ probabilities p_i^z ($i = 0 \dots n$), and one further parameter α determining the probability that $z = 1$. Although the attributes' values are independent given either value for z , we do not know z and so the attributes will appear to be correlated.

In fact, such models are plausible in many domains. As a motivating example consider the case of disambiguation in natural language interpretation tasks, such as pronunciation or context-sensitive spelling correction [GR99]. In a given *context* the occurrence of a single word or a phoneme may be viewed as a random variable that is, to a good approximation, independent of the occurrence of other words. When the context is not known, however, the words will appear to be correlated.

In our studies, the *unsupervised* methods fit the parameters of the CIA model using the well-known *expectation-maximization* (EM) technique [DLR77], and also with a new algorithm we have developed based on estimating covariances. Both these methods, naturally, make some assumptions on the probabilistic model that generates the data.

In the *unsupervised*, nonparametric case, we simply search for a good *linear separator*. This is because the optimal predictors for the binary CIA model (i.e., for predicting one variable given known values for the rest) are also linear. This means that our comparison is “fair” in the sense that neither strategy can choose from classifiers with more expressive power than the other.

As a representative of the non-parametric class of algorithms, we use the Winnow algorithm of [Lit88], with some modifications. Winnow works directly to find a “good” linear separator. It is guaranteed to find a perfect separator if one exists, and empirically seems to be fairly successful even when there is no perfect separator [GR99, Blu97]. It is also very fast. We have made two significant modifications to the basic Winnow algorithm; one modification serves to reduce the variance of the predictor, and the other improves its generalization ability by searching for a “thick” linear separator, namely, a linear separator that is constrained to have a specified margin [CV95].

Our experimental methodology is to first generate synthetic data from a true CIA model and test performance; we then study various deviations from the model. There are various interesting issues involved in constructing good experiments, including the desirability of controlling the inherent “difficulty” of learning a model.

While our parametric algorithm assumes the CIA model with a binary valued hidden variable, we are going to evaluate its performance on a class of distributions parameterized by the number of

values the hidden variable takes, and study how performance changes as the target deviates from the assumed model. A similar tradeoff exists for the non-parametric models. It is easy to see that if we want to predict the most likely value of a binary variable z in the CIA model, given observed values for \tilde{x} , the prediction region will be defined by a linear combination of \tilde{x} . The same is true if one wishes to predict the value of any single attribute, such as x_0 , given values for the other attributes. Thus, the learned linear separator is expressive enough to represent the Bayes optimal predictor. This is not true any more when the unobserved variable z can take more than two values. Since we cannot characterize the entire space, we consider here only deviations in which the data is drawn from a CIA model in which the hidden variable can take more than two values.

Our observations are not *qualitatively* surprising. CIA does well when the assumed model is correct, but performance degrades when the world departs from the model. But as we discuss, we found it surprising how fragile this model can sometimes be, when compared against algorithms such as Winnow. This is even though the data is not linearly separable either, and so one might expect the direct learning techniques to degrade in performance as well. But it seems that Winnow and related approaches are far less fragile.

We summarize the conclusions and contributions of this paper as follows. First, this paper presents empirical data concerning the performance of various algorithms for what is one of the most prototypical learning situations. This may be of assistance in selecting a learning technique for a real problem. Second, we have proposed new learning techniques for the parameters of a binary CIA model, as well as a novel version of the Winnow algorithm.

The main contribution of this work is that our results shed light on the specific tradeoff between the unsupervised learning of a probabilistic model versus direct search for a good classifier¹. Specifically, they illustrate the dangers of predicting using a model that is even “slightly” simpler than the distribution actually generating the data, vs. the relative robustness of directly searching for a good predictor. Given that using the parameter fitting approach studied here (e.g., EM) is a fairly common practice even in cases where the generative model is not well understood, this would seem to be an important practical issue as well as a warning sign, and highlights the need for some better theoretical understanding of the notion of “robustness”².

We note that two recent works [FM99, CGG98] study the problem of learning binary CIA and provide algorithms that provably approximate the mixture distribution (in the KL-divergence sense) efficiently (although with large degree polynomials) for the case in which the probabilities p_i^z are bounded away from 1/2 by a constant.

The paper is organized as follows. In Section 2 we present the CIA model. In Section 3 we present our unsupervised learning methods – the EM algorithm and a Covariances-Based approach. A discussion of the CIA model and its relations to linear separators is presented in Section 4. The direct learning method is presented in Section 5, followed by a section on our methodology and experimental design in Section 6 and the experimental results in Section 7.

¹Of course, in practice, there are well-known ways to improve the performance of the classifiers studied here. These may involve model selection techniques, mixture models, etc. However, using these in our setting may obscure the fundamental nature of the problem considered here, which can be only studied in a restricted and controlled context. This is important because these issues are of great concern in many applications and are not understood theoretically.

²In mathematical statistics [HC95] the notion of robust estimators refers to estimators that are good (small variance, say) for a wide variety of distributions (not necessarily the best for any of them). Using this terminology we find here that EM is not robust, at least not under the type of distributional changes we investigated.

2 Conditionally Independent Attributes

Throughout we assume that each example is a binary vector $\tilde{x} \in \{0, 1\}^{n+1}$, and that each example is generated independently at random according to some unknown distribution on $\{0, 1\}^{n+1}$. We use X_i to denote the i 'th attribute, considered as a random variable, and x_i to denote a value for X_i , $i = 0, 1, \dots, n$.

In the conditionally independent attribute (CIA) model, examples are generated as follows. We postulate a ‘‘hidden’’ variable Z with k values, which takes values z for $0 \leq z < k$ with probability $\alpha_z \geq 0$. Since we must have $\sum_{z=0}^{k-1} \alpha_z = 1$ there are $k - 1$ independent parameters.

Having randomly chosen a value z for the hidden variable, we choose the value x_i for each observable X_i : the value is 1 with probability $p_i^{(z)}$, and 0 otherwise. Here $p_i^{(z)} \in [0, 1]$. The attributes’ values are chosen independently of each other, although z remains fixed.

Note that there are thus $(n + 1)k$ probability parameters $p_i^{(z)}$. In the following, let \mathcal{P} denote the set of all $(n + 1)k + k - 1$ parameters in the model. From this point, and until Section 6, we always assume that $k = 2$ and in this case, to simplify notation, we write α_1 as α , $\alpha_0 (= 1 - \alpha)$ as α' , p_i^1 as p_i and p_i^0 as q_i .

3 Unsupervised Learning for CIA

In this section we consider the unsupervised learning problem induced by the CIA. Given a sample of observations $\tilde{x} = (x_0, x_1, \dots, x_n)$ the unsupervised approach attempts to learn the maximum-likelihood parameters of the distribution given the data. That is, we attempt to find the set of parameters that maximizes the probability of the data observed.

If $\tilde{x} = (x_0, x_1, \dots, x_n)$ is one observation, the likelihood is simply:

$$\begin{aligned} P_{\mathcal{P}}(\tilde{x}) &= P(\tilde{x} \wedge (Z = 1)) + P(\tilde{x} \wedge (Z = 0)) = P(\tilde{x}|Z = 1)P(Z = 1) + P(\tilde{x}|Z = 0)P(Z = 0) \\ &= \alpha \prod_{i=0}^n p_i^{x_i} (1 - p_i)^{1-x_i} + \alpha' \prod_{i=0}^n q_i^{x_i} (1 - q_i)^{1-x_i}. \end{aligned} \quad (1)$$

The maximum likelihood estimate of $\mathcal{P} = (\alpha, p_1, \dots, p_n, q_1, \dots, q_n)$ is the value that maximizes $\prod_{j=1}^s P_{\mathcal{P}}(\tilde{x}^j)$ where \tilde{x}^j ranges over all the s examples seen. (Note that we use superscripts to select particular examples and subscripts range over attributes.)

3.1 The Expectation-Maximization algorithm (EM)

Finding the maximum likelihood parameterization analytically appears to be a difficult problem, even in this rather simple setting. However, a practical approach is to use the well-known *Expectation-Maximization* algorithm (EM) [DLR77], which is an iterative approach that always converges to a local maximum of the likelihood function. By performing several EM runs with different (random) initializations one can hope to find a good (ideally, globally optimal) parameterization.

In this paper we omit a general discussion of how EM works but for completeness provide below the derivation of the update rule for the general CIA model.

We begin with a randomly chosen parameterization \mathcal{P} . Using Eq. 1, the probability that a data point \tilde{x}^j comes from each of the k values of Z is given by:

$$P_r^j = P(z = r|\tilde{x}^j) = \frac{P(\tilde{x}^j|P(z = r))}{P(\tilde{x}^j)} = \frac{\alpha_r \prod_{i=0}^n (p_i^r)^{x_i^j} (1 - p_i^r)^{(1-x_i^j)}}{\sum_{z=1}^k [\alpha_z \prod_{i=0}^n (p_i^z)^{x_i^j} (1 - p_i^z)^{(1-x_i^j)}]}. \quad (2)$$

Next we can compute the expected log likelihood of the s examples seen:

$$\begin{aligned}
E\left(\sum_j \log P(\tilde{x}^j | \tilde{\alpha}_z, \tilde{p}_i^z)\right) &\equiv \sum_j E(\log P(\tilde{x}^j | \tilde{\alpha}_z, \tilde{p}_i^z)) \\
&\equiv \sum_j (P_1^j \cdot \log P(1, \tilde{x}^j | \tilde{\alpha}_z, \tilde{p}_i^z) + \dots + P_k^j \cdot \log P(k, \tilde{x}^j | \tilde{\alpha}_z, \tilde{p}_i^z)) \\
&\equiv \sum_j \sum_{z=1}^k P_z^j \cdot \log(\tilde{\alpha}_z \cdot \prod_{i=0}^n (\tilde{p}_i^z)^{x_i^j} (1 - \tilde{p}_i^z)^{1-x_i^j}) \\
&\equiv \sum_j \sum_{z=1}^k P_z^j \cdot (\log \tilde{\alpha}_z + \sum_{i=0}^n x_i^j \log \tilde{p}_i^z + (1 - x_i^j) \log(1 - \tilde{p}_i^z))
\end{aligned}$$

Differentiating with respect to all parameters we can now find new values for $\tilde{\alpha}_i$ and \tilde{p}_i^z for which the expected (log) likelihood receives an extremal value. Remember that $\tilde{\alpha}_k = 1 - \sum_{i=1}^{k-1} \tilde{\alpha}_i$. We have:

$$\frac{dE}{d\tilde{\alpha}_z} = \sum_j \left(\frac{P_z^j}{\tilde{\alpha}_z} - \frac{P_k^j}{\tilde{\alpha}_k} \right) = 0,$$

and thus the new values for the α s are given by:

$$\tilde{\alpha}_z = \begin{cases} \tilde{\alpha}_z \frac{\sum_j P_z^j}{\sum_j P_k^j} & \text{if } z \neq k \\ 1 - \sum_{i=1}^{k-1} \tilde{\alpha}_i & \text{if } z = k \end{cases}$$

Similarly,

$$\frac{dE}{d\tilde{p}_i^z} = \sum_j P_z^j \left(\frac{x_i^j}{\tilde{p}_i^z} - \frac{1 - x_i^j}{1 - \tilde{p}_i^z} \right) = 0$$

and thus

$$\tilde{p}_i^z = \frac{\sum_j P_z^j x_i^j}{\sum_j P_z^j}.$$

In summary (for the binary CIA case), we begin with a randomly chosen parameterization \mathcal{P} , and then we iterate the following two computation steps until (apparent) convergence:

Expectation: For all \tilde{x}^j , compute $u_j = P_{\mathcal{P}}(\tilde{x}^j \wedge z = 1)$ and $v_j = P_{\mathcal{P}}(\tilde{x}^j \wedge z = 0)$.

Maximization: Reestimate \mathcal{P} as follows (writing $U = \sum_j u_j$ and $V = \sum_j v_j$):

$$\alpha \leftarrow \sum_{j=1}^s u_j / (U + V) \quad p_i \leftarrow \sum_{\{j: \tilde{x}_i^j=1\}} u_j / U \quad q_i \leftarrow \sum_{\{j: \tilde{x}_i^j=0\}} v_j / V.$$

The maximization phase works as though we were estimating parameters by taking averages based on weighted *labeled* data (i.e., in which we see z). If \tilde{x}^j is a sample point, these fictional data points are $(\tilde{x}^j, z = 1)$ with weight u_j/U and $(\tilde{x}^j, z = 0)$ with weight v_j/V .

After convergence has been detected (itself a decision that requires some care) all we know is that we are near a *local* minima of the likelihood function. Thus it is prudent to repeat the process with many different restarts. Since our objective is to measure the learning potential intrinsic to CIA, rather than computational efficiency, all our experiments were extremely conservative concerning the stopping criteria at each iteration, and in the number of iterations we tried. But in practice, we are never sure that the true optimum has been located.

3.2 Covariances-Based approach

Partly in response to concern just expressed, we also developed another technique for learning \mathcal{P} . The algorithm, which we call COV, is based on measuring the covariance between pairs of attributes.

Since we do not see Z , attributes will appear to be correlated. Let $\mu_i, \mu_j, \mu_{i,j}$ be the expectations of $X_i, X_j, (X_i \text{ and } X_j)$, respectively:

$$\mu_i = \alpha p_i + \alpha' q_i; \mu_j = \alpha p_j + \alpha' q_j; \mu_{i,j} = \alpha p_i p_j + \alpha' q_i q_j, \quad (3)$$

and denote the covariance between X_i and X_j ,

$$y_{i,j} \equiv \mu_{i,j} - \mu_i \mu_j. \quad (4)$$

Then, if the CIA model is the correct model for the data, it is easy to show that

$$y_{i,j} = \alpha \alpha' \delta_i \delta_j$$

where δ_i denotes $p_i - q_i$.

Furthermore, we can get very accurate estimates of μ_i just by observing the proportion of samples in which x_i is 1³. Thus, if we could estimate both α and the δ_i s it would be trivial to set up equations and solve for estimates of p_i and q_i .

To estimate δ_i , suppose we have computed all the pairwise covariances using the data; we use $\hat{y}_{i,j}$ to denote our estimate of $y_{i,j}$. For any distinct $j, k \neq i$ we clearly have

$$\alpha \alpha' \delta_i^2 = \frac{|y_{i,j} y_{i,k}|}{y_{j,k}} \quad (5)$$

so we could estimate δ_i^2 using this equation. A better estimate would be to consider *all* pairs j, k and average the individual estimates. However, not all individual estimates are equally good. The smaller $y_{j,k}$ is, the less reliable we should expect the estimate to be (and in the limit, where X_j and X_k are perfectly uncorrelated, we get no valid estimate at all). This suggests that we use a weighted average, with the weights proportional to $y_{j,k}$. Using these weights leads us to the next equation for determining δ_i , which, after simplification, is:

$$\alpha \alpha' \delta_i^2 = \frac{\sum_{j,k:j \neq k \neq i} |y_{i,j} y_{i,k}|}{\sum_{j,k:j \neq k \neq i} |y_{j,k}|} = \frac{(\sum_{j:j \neq i} |y_{i,j}|)^2 - \sum_{j:j \neq i} y_{i,j}^2}{\sum_{j,k:j \neq k} |y_{j,k}| - 2 \sum_{j:j \neq i} |y_{j,i}|} \quad (6)$$

By substituting the estimates $\hat{y}_{i,j}$ we get an estimate for $\alpha \alpha' \delta_i^2$. This estimate can be computed in linear time except for the determination of $\sum_{j,k:j \neq k} |y_{j,k}|$ which, although quadratic, does not depend on i and so can be computed once and for all.

Thus this phase of the algorithm takes $O(n^2)$ time in total to estimate $\alpha \alpha' \delta_i^2$ for all i . In practice (although not in our experiments), one can construct variants of this procedure that average over fewer terms, gaining efficiency at the sacrifice of some accuracy. The limiting procedure, which is to use just a single pair j, k for each i , gives a linear algorithm. However, this is probably not advisable unless the sample size is quite large.

It remains only to estimate α and the signs of the δ_i 's. Briefly, to determine the signs we first stipulate that δ_0 is positive. (Because we never see z , one sign can be chosen at random.) In principle, then, the sign of δ_j will then be equal to the sign of $y_{0,j}$, which we have an estimate for.

³Since this is just binomial sampling, the error in the estimate decreases as $1/\sqrt{\text{sample size}}$.

In practice, this can be statistically unreliable for small sample sizes and so we use a slightly more involved “voting” procedure.

Finally we estimate α . We have found no better method of doing this than to simply search for the optimal value, using likelihood as the search criterion. However, this is only a 1-dimensional search and it turns out to be quite efficient in practice.

As we have mentioned above, the expectations μ_i can be estimated accurately, with error that decreases as $1/\sqrt{\text{sample size}}$. Furthermore, the deviation from this estimate can be bounded, with high probability, using standard techniques. Similarly, this can be done for the estimates in Eq. 5 and in Eq. 6. Consequently, it is possible to show that with high probability, given sufficient data, the estimates we derive for δ_i^2 are accurate, and when it is bounded away from 0, so is its sign. See, for example, [CGG98] for a similar analysis. We do not give the details of the fairly standard analysis here since this is not the focus of this work.

The entire algorithm works substantially faster than EM in most our experiments. In our experiments the bulk of the work is the $O(n^2)$ non-iterative phase in which we estimate $\alpha\alpha'\delta_i^2$.

4 Linear Separators and CIA

In this section we develop the relations between learning in the CIA model and linear separators, so that we can use it later on in the supervised learning approach to learning in the CIA model.

Given a fully parameterized CIA model, we may be interested in predicting the value of one variable, say X_0 , given known values for the remaining variables. One can show that in fact the optimal prediction region is given by a linear separator in the other variables. The observation that this holds for the case where one wishes to predict z goes back to [MP69]. Observing that it holds when predicting any one of the observed attributes from the rest is a straightforward extension that we present below.

It is implicit in the following that we are considering some fixed set of parameters \mathcal{P} . When X_0 is unobserved, the conditional odds that X_0 is 1 can be written:

$$\begin{aligned}
O(X_0 = 1|X_1 = x_1, \dots, X_n = x_n) & \\
&\equiv \frac{P(X_0 = 1, X_1 = x_1, \dots, X_n = x_n)}{P(X_0 = 0, X_1 = x_1, \dots, X_n = x_n)} \\
&\equiv \frac{\alpha P(1, x_1, \dots, x_n|Z = 1) + \alpha' P(1, x_1, \dots, x_n|Z = 0)}{\alpha P(0, x_1, \dots, x_n|Z = 1) + \alpha' P(0, x_1, \dots, x_n|Z = 0)} \\
&\equiv \frac{q_0 \left(\frac{\alpha p_0 P(X_1=x_1, \dots, X_n=x_n|Z=1)}{\alpha' q_0 P(X_1=x_1, \dots, X_n=x_n|Z=0)} + 1 \right)}{(1 - q_0) \left(\frac{\alpha(1-p_0)P(X_1=x_1, \dots, X_n=x_n|Z=1)}{\alpha'(1-q_0)P(X_1=x_1, \dots, X_n=x_n|Z=0)} + 1 \right)} \\
&\equiv \frac{p_0 Q + q_0}{(1 - p_0)Q + (1 - q_0)},
\end{aligned}$$

where

$$Q = \frac{\alpha P(X_1 = x_1, \dots, X_n = x_n|Z = 1)}{\alpha' P(X_1 = x_1, \dots, X_n = x_n|Z = 0)}.$$

The region in which $X_1 = 1$ is more likely than $X_1 = 0$ is the region in which $O(x_0 = 1|x_1, \dots, x_n) > 1$. From the above, this occurs when

$$p_0 Q + q_0 > (1 - p_0)Q + (1 - q_0),$$

or

$$Q \cdot (2p_0 - 1) > 1 - 2q_0.$$

Assuming, for the moment, that $p_0 > 1/2$, we have that there is some constant $C = \frac{1-2q_0}{2p_0-1}$ such that if $Q > C$, $X_0 = 1$ is more likely than $X_0 = 0$. Equivalently, $\log Q > \log C$. Finally, we note that

$$\begin{aligned} L &= \log Q \\ &= \log \frac{\alpha}{\alpha'} + \sum_{i=2}^n (\log p_i - \log q_i) \cdot x_i + (\log(1-p_i) - \log(1-q_i)) \cdot (1-x_i) \\ &= c_* + c_1 x_1 + \dots + c_n x_n, \end{aligned}$$

is a linear function of the x_i 's. Thus the Bayes optimal prediction rule for x_0 , given x_1, \dots, x_n is given by the linear condition (under which we predict 1): $\log Q - \log C > 0$. Note that if in fact $p_0 < 0.5$, the rule is reversed: we predict 1 if $\log Q - \log C < 0$. Finally note that $C < 0$ if both $p_0, q_0 > 0.5$ (resp. $p_i, q_i < 0.5$). In this case the rule is degenerate: always predict 1 (resp., 0).

Note that if we had been trying to predict some other variable, x_1 say, rather than x_0 , the coefficients for all x_i ($i \neq 0, 1$) would have been the same, up to sign. (In fact, $|c_i| = \left| \log \frac{p_i(1-q_i)}{(1-p_i)q_i} \right|$ always.) Only the constant term, in this case $c_0 = c_* - \log \frac{1-2q_1}{2p_1-1}$, depends on which variable we are trying to predict. The same observation is true if we wish instead to predict the value of Z given values for all of the x_i . This observation means that, in principle, if we learn a linear predictor for one variable we can use this to help form a predictor for any other variable, or for Z . (There are various ways of dealing with the varying constant coefficient. Perhaps the simplest is to note that we can perform a binary search to find the “best” constant; i.e., that which leads to best performance on a training set.) One application of this is when we really are interested in Z . Since we do not see Z , this is an unsupervised learning problem. However, we can convert it to a supervised learning problem by trying to learn a good linear separator for any variable x_i . On the other hand suppose, more commonly, that we really are interested in learning how to predict x_0 . We might try to learn a linear predictor for x_0 , perhaps using *Winnnow*. However, we can also simultaneously try to learn predictors for *each* variable x_i given the others. To the extent the CIA model is accurate, these predictors can be used to improve the predictor we find for x_0 .

In the experimental study that follows, we concentrate on predicting x_0 and point out, given the discussion above, that the same holds for any other predictions, including uncovering the hidden variable.

We make a simple observation concerning learnability of the Bayes optimal predictor for a binary CIA model. This predictor clearly minimizes the expected *loss* among the class of linear predictors, which for such a predictor L is defined to be 1 for points in the region $L < 0$ such that $x_0 = 1$ and for points in the region $L > 0$ for which $x_0 = 0$, and is 0 otherwise. This suggests an obvious learning strategy: simply try to find the line which minimizes this loss on the training set. How many samples would then be required? We can appeal to the result of [KS94] (Thm 5.1), which shows that, even though there is no perfect classifier, the sample size required nevertheless depends the VC-dimension of the class (see, e.g., [Vap95]), which for the class of linear classifiers is just $n + 1$. It follows that a polynomial size sample is sufficient to ensure the “correctness” (in the precise sense of [KS94]) of this learning procedure for learning the Bayes-optimal separator under the binary CIA assumption.

This observation only tells us about sample complexity. Unfortunately, in general the task of finding a linear separator that minimizes disagreements on a collection of examples is known to be NP-hard [HS92]. So instead we use an algorithm called *Winnnow* that is known to produce good

results when a linear separator exists, as well as under certain more relaxed assumptions [Lit91], and appears to be quite effective in practice [GR99, Blu97].

5 Learning using a Winnow-based algorithm

The basic version of the Winnow algorithm [Lit88] keeps an n -dimensional vector $w = (w_1, \dots, w_n)$ of positive weights (i.e., w_i is the weight associated with the i th feature), which it updates whenever a mistake is made. Initially, the weight vector is typically set to assign an equal positive weight to all features. The algorithm has 3 parameters, a promotion parameter $\alpha > 1$, a demotion parameter $0 < \beta < 1$ and a threshold θ . For a given instance (x_1, \dots, x_n) the algorithm predicts that $x_0 = 1$ iff

$$\sum_{i=1}^m w_i x_i > \theta.$$

The algorithm updates its hypothesis only when a mistake is made. If the algorithm predicts 0 and the label (i.e., x_0) is 1 (positive example) then the weights which correspond to active attributes ($x_i = 1$) are promoted—the weight w_i is replaced by a larger weight $\alpha \cdot w_i$. Conversely, if algorithm predicts 1 and the received label is 0, then the weights which correspond to active features are demoted by factor β . In both cases, all other weights maintain the same value. We allow for negative weights as follows. Given an example (x_1, \dots, x_n) , we rewrite it as an example over $2n$ variables $(y_1, y_2, \dots, y_{2n})$ where $y_i = x_i$ and $y_{n+i} = 1 - x_i$, $i = 1, \dots, n$. We then apply Winnow just as above to learn $2n$ (positive) weights. If w_i^+ is the weight associated with x_i and w_i^- is the weight associated with x_{n+i} (i.e., $1 - x_i$), then the prediction rule is simply to compare

$$\sum_{i=1}^n (w_i^+ x_i + w_i^- (1 - x_i))$$

with the threshold.

In the experiments described here we have made two significant modifications to the basic algorithm. To reduce variance, our final classifier is a weighted average of several classifiers; each is trained using a subsample from the training set, and its weight is based based on how well it was doing on that sample (this is a form of Bagging [Bre96]). Second, we biased the algorithm so as to look for “thick” classifiers. To understand this, consider the case in which the data is perfectly linearly separable (this is *not* the case for the probabilistic concepts generating our experimental data or for most real world data). Then there will generally be many linear concepts that separate the training data we actually see. Among these, it seems plausible that we have a better chance of doing well on the unseen test data if we choose a linear concept that separates the positive and negative training examples as “widely” as possible. This intuition is made formal within the support vector machine approach [CV95]. The idea of having a wide separation is less clear when there is no perfect separator, but we can still appeal to the basic intuition. To bias the search towards “thick” separators, we change Winnow’s training rule somewhat. We now have a new margin parameter τ . As before, we always update when our current hypothesis makes a mistake, but now we *also* update if $|\sum_{i=1}^n w_i x_i - \theta|$ is less than τ , even if the prediction is correct. In our experiments, we found that performance when using this version of Winnow is better than that of the basic algorithm, so in this paper we present results for the former.

6 Experimental Methodology

Aside from the choice of algorithm used, the number of attributes n , and the sample size s , our experiments also differed in two other dimensions. These are the type of process generating the data (we will be interested in various deviations from CIA), and the “difficulty” of the problem. These features are determined by the *data model* we use (i.e., the distribution over $\{0, 1\}^{n+1}$ used to generate data sets).

Our first experiments consider the case where the data really is drawn from a binary CIA distribution. We associated with any such distribution a “difficulty” parameter B , which is the accuracy with which one could predict the value of Z if one actually knew the correct model. (Of course, even with knowledge of the correct model we should not expect 100% accuracy. Specifically, B is computed by averaging $\max\{P_1^j, 1 - P_1^j\}$ from Eq. 2 across data points and sampled modeled.) The ability to control B allows us to select and study models with different qualitative characteristics. In particular, this has allowed us to concentrate our experiments on fairly “hard” instances⁴, and to more meaningfully compare trials with differing numbers of attributes.

We denote by $\text{CIA}(n, 2, b)$ the class of all data models which are binary CIA distributions over n variables with difficulty b . It is nontrivial to efficiently select random models from this class. Briefly, our scheme is to choose each parameter in a CIA model independently from a symmetric beta distribution. Thus, the model parameters will have expected value 0.5. We choose the parameter of the beta distribution (which determines concentration about 0.5) so that the average B value of the models thus generated, equals b . Finally, we use rejection sampling to find CIA models with B values that are exactly $b \pm 1\%$. The next family of data models we used are also CIA models, but now using more than two values for the hidden variable. We denote the family using k values as $\text{CIA}(n, k, b)$ where n and b are as before. When $k > 2$ there are more complex correlation patterns between the X_i than when $k = 2$. Furthermore, the optimal predictor is not necessarily linear. The specific results we discuss in the next section have concentrated on this case.

Given any set of parameters, including a particular class of data models, our experiments are designed with the goal of good statistical accuracy. We repeatedly (typically 100 to 300 times) choose a data model at random from the chosen class, choose a sample of the appropriate size from this model, and then run all our algorithms. Each algorithm produces a (linear) hypothesis. We measure the success rate S_{alg} (i.e., the proportion of times a hypothesis makes the correct prediction of x_0) by drawing yet more random samples from the data model being used. In the test phase we always draw enough new samples so that the confidence interval for S_{alg} , for the results on a single model, has width at most $\pm 1\%$.

We use the S_{alg} values to construct a *normalized* measure of performance (denoted T) as follows. Let S_{best} be the best possible accuracy attainable for predicting x_0 (i.e., the accuracy achieved by the actual model generating the data). Let S_{const} denote the performance of the best possible constant prediction rule (i.e., the rule that predicts the most likely *a priori* value for x_0). Note that S_{const} and S_{best} can vary from model to model. For each model we compute

$$\frac{S_{alg} - S_{const}}{S_{best} - S_{const}},$$

and our normalized statistic T is the average of these values. It can be thought of as measuring the

⁴Note that if one simply chooses parameters of a CIA model independently at random, without examining the difficulty of the model or adjusting for n , one will get many truly trivial problems (in which X_0 is best predicted with a constant) and, as n increases, one would also get a growing, and disproportionate, number of “practically” trivial problems (in which any small subset of the attributes is enough to predict Z with nearly 100% accuracy, and thus predict optimally for X_0).

percentage of the *possible* predictive power, over a plausible baseline, that an algorithm achieves. Note that for a single model, the constant predictor and the optimal predictor score 0 and 1, respectively, in this measure. However, we still occasionally see negative normalized scores, especially when algorithms are trained on very small samples.

Two other performance measures that we considered were the averaged *raw* accuracy figures, and the average accuracies where we normalize each raw figure simply by dividing by the corresponding S_{best} ; we call this latter measure U .⁵ The *raw* measure has the flaw that it is hard to compare between different data models; we wouldn't know whether 70%, say, is superb performance relative to what is possible or whether it is unacceptable. The last two measures have, in principle, the statistical disadvantage that they weight “easy” problems excessively relative to hard problems.

7 Results

We only report on a small, but representative, selection of our experiments in any detail. For instance, although we have considered many values of n ranging from 10 to 500, here we show six graphs giving the learning curves for $CIA(n, k, 0.90)$ for $n = 10, 75$, and for $k = 2, 3, 5$; as noted, we display the normalized statistic T . The error bars show the standard error⁶, providing an indication of accuracy.

Not surprisingly, when the data model is binary CIA, the EM algorithm does extremely well, learning significantly (if not overwhelmingly) faster than Winnow. But as we depart from the binary CIA assumption, the performance of EM quickly degrades.

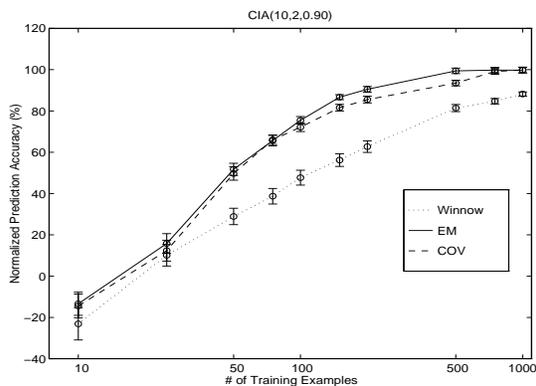


Figure 1: CIA(10,2,0.9)

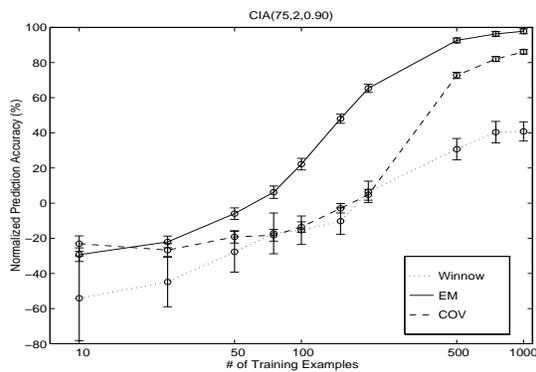


Figure 2: CIA(75,2,0.9)

When $k = 3$ performances is, on the whole, very similar for Winnow and EM. But when $k = 5$ Winnow is already superior to EM; significantly and uniformly so for $n = 10$. For fixed k the difference seems to become somewhat less dramatic as n increases; in Figure 6 (for $n = 75$) Winnow is less obviously dominant, and in fact is not better than EM until the sample size has reached 100. (But when $s \leq n$, meaning that we have fewer samples than attributes, the performance is uniformly dismal anyway.)

⁵The three measures are *not* interderivable because S_{const} and S_{best} vary from model to model. Thus, in a certain sense, they measure different things. In a certain sense, the averaged T penalizes “easy” problems (in which it is easy to do fairly well) relative to hard problems more than does U , and even more when compared with the raw average.

⁶Computed as the observed standard deviation, divided by the square root of the number of trials.

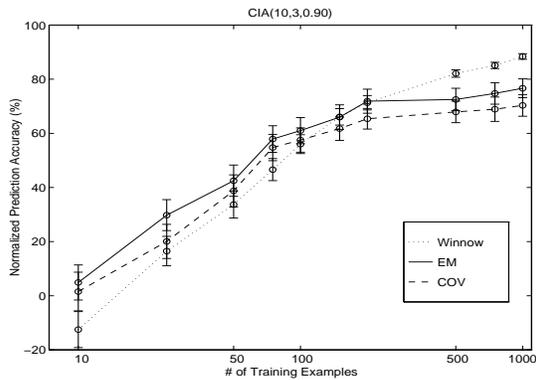


Figure 3: CIA(10,3,0.9)

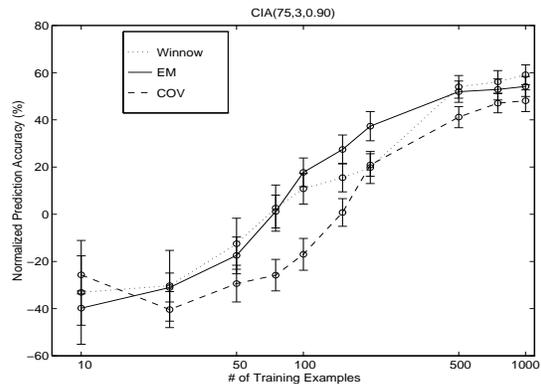


Figure 4: CIA(75,3,0.9)

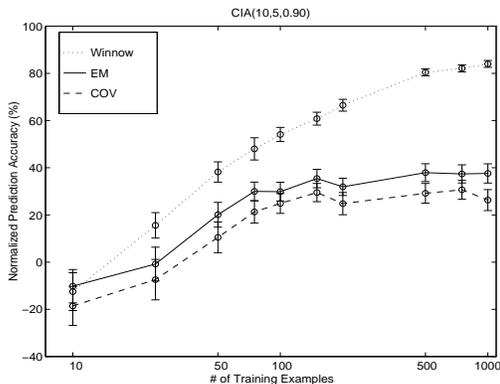


Figure 5: CIA(10,5,0.9)

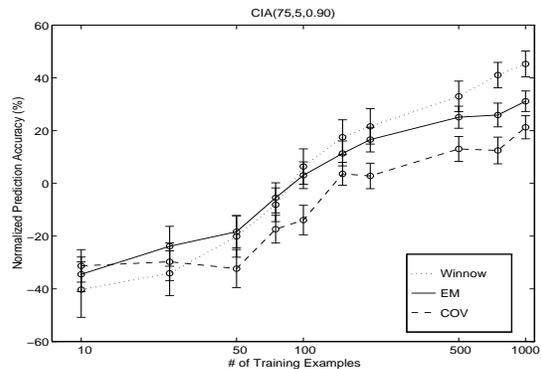


Figure 6: CIA(75,5,0.9)

Should we attribute this degradation to the binary CIA assumption, or to the EM itself? (I.e., it is possible, in principle, that the true maximum likelihood binary-CIA model would perform much better than we are seeing, and that the problem is simply that EM doesn't find it.) This question is our reason for also considering the covariance algorithm. We see that the results for COV are generally similar to EM's, supporting our belief that the phenomena we see are properties inherent to the model rather than to the specific algorithm being used. Similarly (the results are omitted) we have tried several other algorithms that try to find good linear separators directly, including the classic *Perceptron* algorithm [MP69]; our version of Winnow was the best on the experiments we tried and thus we conjecture that its performance is (somewhat) indicative of what is possible for any such approach.

As the comparison between $n = 10$ and $n = 75$ illustrates, there is little qualitative difference between the phenomena observed as the number of attributes increases. Nevertheless, as n grows it does seem that Winnow needs more examples before its performance surpasses that of the other algorithms (for any fixed k). As already noted, this may be due simply to the very "noisy" nature of the region $s \leq n$. It is also partially an artifact of way we select models by fixing the difficulty level: for fixed b , as n increases all the model probabilities must become closer to 0.5 on average.

As previously noted, we also experimented with varying "difficulty" (B) levels. Illustrative of these are Figures 7 and 8, which show runs under CIA(10, 2, 0.98) and CIA(10, 5, 0.98). As can be

seen the main difference is that Winnow is a little faster in surpassing EM when the data deviates from the assumed model, but when the data model really is binary CIA, and EM converge even faster to an optimal performance.

Our final graphs, Figures 9 and 10, show results for CIA(10, 3, 0.98) using the two other performance measures discussed in Section 6. These are extremely similar and useful mostly for indicating the scale of the *unnormalized* performance (or *partially* normalized in the case of U). We found all measures to highlight essentially the same qualitative phenomena, but that T was more informative about the details.

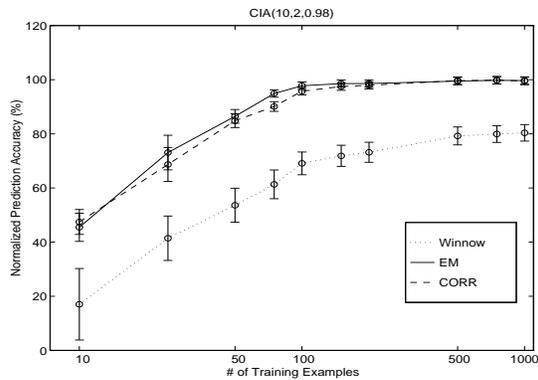


Figure 7: CIA(10,2,0.98)

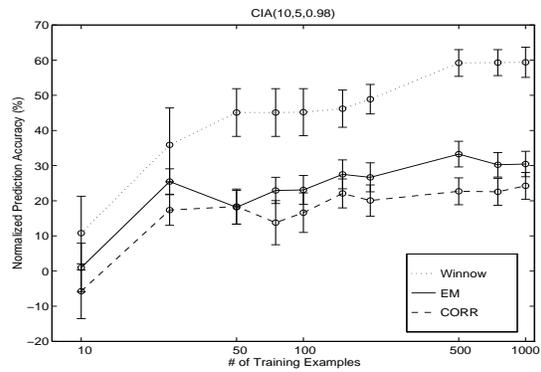


Figure 8: CIA(10,5,0.98)

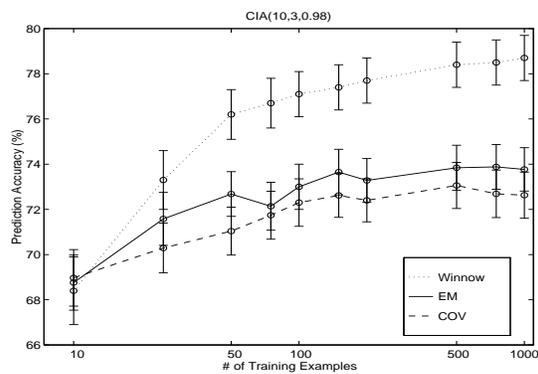


Fig. 9: CIA(10,3,0.98) (Raw Accuracy)

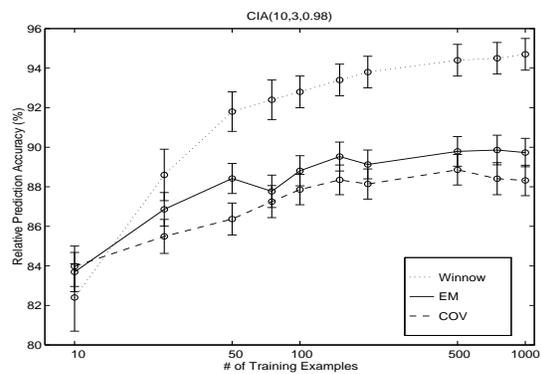


Fig. 10: CIA(10,3,0.98) (U)

These patterns were confirmed when we tried to compare the approaches on real data. We have used data that originates from a problem in which assuming a hidden “context” variable seems somewhat plausible. The data is taken from the context-sensitive spelling correction domain. We used one data set from those that were used in [GR99]. For example, given sentences in which the word *passed* or *past* appear, the task is to determine, for each such occurrence, which of the two it should be. This task may be modeled by thinking of the “context” as a hidden variable in our sense. Yet when we tried to learn in this case under the CIA model, with a binary valued hidden variable, the results were no better than just predicting the most likely classification (around 70%). Winnow, in contrast, performed extremely well and exceeds 95% on this task. We hesitate to read

much into our limited real-data experiments, other than to note that so far they are consistent with the more careful experiments on synthetic data.

8 Conclusion

By restricting to a binary hidden variable, we have been able to consider a “fair” comparison between unsupervised learning methods—probabilistic model construction, and more traditional algorithms that directly learn a classification—at least in the sense that both have the same expressive power. Our results illustrate the dangers of predicting using a model that is even “slightly” simpler than the distribution actually generating the data, vs. the relative robustness of directly searching for a good predictor. Our conclusions concerning the fragility of the former should not be surprising but we believe that given the importance of the problem it is valuable to have some idea of the true significance of the effect.

As we have indicated, in many real-world cases, where a model of the sort we have considered here seems plausible, it is impossible to nail down more specific characterizations of the probabilistic model. Our results exhibit how important it is to use the correct model and how sensitive are the results to deviations from it, when attempting to learn using model construction. One possible conclusion may be that in cases when the probabilistic model is not well understood but the tasks to be performed with the learned model are – one should prefer a more direct learning approach.

The purpose of this paper is not, of course, to advocate that in practice one should use either Winnow or binary CIA in exactly the form considered here. A richer probabilistic model should be used along with a model selection phase. However, studying the problem in a restricted and controlled environment is crucial so as to understand the nature and significance of this fundamental problem. The simple parametric approach can also be improved using various voting techniques, for instance. The success of Winnow may indicate that, when performing a prediction task in the presence of a rich probabilistic model, exploring nonparametric techniques may be a legitimate contender to trying to construct an approximate model. A careful comparison between such algorithms could be productive.

References

- [Blu97] A. Blum. Empirical support for winnow and weighted majority based algorithms: results on a calendar scheduling domain. *Machine Learning*, 26:1–19, 1997.
- [Bre96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [CGG98] M. Cryan, L. A. Goldberg, and P. W. Goldberg. Evolutionary trees can be learned in polynomial time in the two-state general markov model. In *39th Symp. on Foundations of Computer Science*, pages 436–445, 1998.
- [CV95] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society B*, 39:1–38, 1977.
- [FM99] Y. Freund and Y. Mansour. Estimating the mixture of two product distributions. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 53–62. ACM Press, New York, NY, 1999.

- [GR99] A. R. Golding and D. Roth. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999.
- [HC95] R. V. Hogg and A. T. Craig. *Introduction to Mathematical Statistics*. Prentice-Hall, 1995.
- [HS92] K. Höffgen and H. Simon. Robust trainability of single neurons. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 428–439, New York, New York, 1992. ACM Press.
- [KS94] M. Kearns and R. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48:464–497, 1994.
- [Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [Lit91] N. Littlestone. Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 147–156, San Mateo, CA, 1991. Morgan Kaufmann.
- [MP69] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.
- [Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
- [Vap95] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.