Learning Non-Linear Functions for Text Classification

Cohan Sujay Carlos
Aiaioo Labs
Bangalore
India
cohan@aiaioo.com

Geetanjali Rakshit
IITB-Monash Research Academy
Mumbai
India
geet@cse.iitb.ac.in

Abstract

In this paper, we show that generative classifiers are capable of learning non-linear decision boundaries and that non-linear generative models can outperform a number of linear classifiers on some text categorization tasks.

We first prove that 3-layer multinomial hierarchical generative (Bayesian) classifiers, under a particular independence assumption, can only learn the same linear decision boundaries as a multinomial naive Bayes classifier.

We then go on to show that making a different independence assumption results in nonlinearization, thereby enabling us to learn non-linear decision boundaries.

We finally evaluate the performance of these non-linear classifiers on a series of text classification tasks.

1 Introduction

Probabilistic classifiers predict a *class* c given a set of *features* F of the data point being classified, by selecting the most probable class given the features, as shown in Equation 1.

$$\arg\max_{c} P(c|F) \tag{1}$$

Bayesian classifiers, also known as generative classifiers, are those that learn a model of the joint probability $P(F, c)$ of the inputs F and labels $c \in C$ (Ng and Jordan, 2001), and use the *Bayes rule* (2) to invert probabilities of the features F given a class c into a prediction of the class c given the features F.

$$P(c|F) = \frac{P(F|c) \times P(c)}{P(F)} \tag{2}$$

2-layer classifiers such as the perceptron and the naive Bayes classifier are only capable of learning linear decision boundaries (Minsky and Papert, 1988) passing through the origin (if learning a multinomial probability distribution), or quadratic decision boundaries (with a Gaussian probability distribution or kernel).

The multilayer perceptron, a 3-layer classifier with an input layer, a hidden layer and an output layer, is capable of learning non-linear decision boundaries (Cybenko, 1989; Andoni et al, 2014). It is therefore only to be expected that 3-layer generative models should also be capable of learning the same. However, it turns that under certain conditions, 3-layer generative models for classification are only as powerful as 2-layer models, and under other conditions, are capable of learning non-linear decision boundaries.

In this paper, we study 3-layer hierarchical multinomial generative models, and define the conditions under which their decision boundaries are linear or non-linear. The main contributions of the paper are the following:

- We prove that a certain set of independence assumptions results in linear decision boundaries and identical classification performance to a naive Bayes classifier.

- We demonstrate that a different set of independence assumptions makes models with the same parameters capable of learning non-linear decision boundaries and certain non-linear functions, for instance, a scaled and shifted XOR function.

- We show through experiments that such **non-linear hierarchical Bayesian models** can outperform many commonly used machine learning models at certain text classification tasks.

2 Related Work

Deep learning is a term used to describe any method of machine learning involving the use of multiple processing layers to learn representations of data (LeCun et al., 2015; Bengio, 2009).

2.1 Neural Network Models

Two machine learning algorithms used in deep learning are the multilayer perceptron (Rumelhart et al, 1986) and the autoencoder (Alex et al, 2011), both of which are neural networks (Wang and Yeung, 2016).

2.2 Probabilistic Graphical Models

Probabilistic graphical models are concise graph-based representations of probability distributions over possibly complex and high-dimensional spaces, with the nodes representing variables of interest and various properties of the graph representing functions interpretable as probabilities (Koller and Friedman, 2009). Probabilistic graphical models are usually of one of two types, depending on whether their edges are directed or undirected.

Undirected probabilistic graphical models (also known as Markov random fields) have undirected edges and are models of probability distributions whose factorization is a function of the cliques in the graph (Wainwright and Jordan, 2008). They represent discriminative classifiers, which learn the probability of the labels given the features $P(c|F)$ directly (Ng and Jordan, 2001) without first learning $P(F|c)$ and then applying Bayesian inversion. Two machine learning models used in deep learning - Boltzmann machines (Ackley et al, 1986) and restricted Boltzmann machines (Smolensky, 1986) - are both undirected probabilistic graphical models (Fischer and Igel, 2012).

Directed probabilistic graphical models (also known as Bayesian networks) are on the other hand directed acyclic graphs where the factorization of the probability distribution is a function of the parent-child relationships in the graph.

2.3 Directed Probabilistic Graphical Models

For each vertex $i \in 1 \dots n$ in a directed probabilistic graphical model and the set of ancestor vertices $a(i)$ of i (vertices with a sequence of directed edges leading to i), the random variables x_1, x_2 $\dots x_n$ attached to the nodes represent a family of probabilistic distributions that factorise according to Equation 3.

$$P(x_1, x_2 \dots x_n) = \left(\prod_{1 \leq i \leq n} P(x_i | x_{a(i)}) \right) \quad (3)$$

In this paper, we focus on a subset of directed probabilistic graphical models known as hierarchical Bayesian models (Wainwright and Jordan, 2008).

2.4 Hierarchical Bayesian Models

Hierarchical Bayesian models are directed probabilistic graphical models in which the directed acyclic graph is a tree (each node that is not the root node has only one immediate parent). In these models the ancestors $a(i)$ of any node i do not contain any siblings (every pair of ancestors is also in an ancestor-descendent relationship). The factorization represented by a hierarchical Bayesian model is just a special case of Equation 3 (Wainwright and Jordan, 2008). Such models naturally lend themselves to classification tasks because the root node can be used to represent the set of classes C and the leaf nodes the set of features F.

The naive Bayes classifier is a special case of a hierarchical Bayesian classifier because it consists of a tree that is only two layers deep (a root node and a layer of leaf nodes). So, for the naive Bayes classifier, Equation 3 reduces to Equation 4.

$$P(F, c) = P(F|c)P(c) = \prod_{f \in F} P(f|c)P(c) \quad (4)$$

Naive Bayes classifiers are easy to train because all the nodes in the directed graph representation of a naive Bayes classifier are visible nodes. Since all the edge probabilities that need to be estimated in the learning step are between visible variables, they are easy to estimate.

With hierarchical Bayesian models, there are also "hidden" nodes which do not correspond to either labels or features in the data used in training. The estimation of the probabilities involving such hidden nodes requires the use of a training method such as Gibbs' sampling, variational method or expectation maximization (which is described below).

Though hierarchical support vector machine models (Sun and Lim, 2001) and hierarchical neural networks (Ruiz and Srinivasan, 2002) have been explored in hierarchical text categorization tasks, there has been relatively little work on hierarchical generative models. There have been evaluations of the performance of hierarchical generative models on hierarchical classification tasks (Veeramachaneni et al., 2005), on images (Liao and Carneiro, 2015) and also on text classification tasks (Vaithyanathan et al., 2000). But none of the earlier papers attempt formal characterization of the linear and non-linear aspects of hierarchical generative models. So, to our knowledge, there has been no earlier study establishing the conditions for linearity and non-linearity in such models along with a demonstration of superior performance in a typical text categorization task.

2.5 Expectation Maximization

The *expectation maximization* (EM) algorithm is an iterative algorithm for estimating the parameters of a directed graphical model with hidden variables. It consists of two steps: *expectation* (E-step) and *maximization* (M-step) (Dempster et al., 1977).

The parameters of the directed graphical model are at first randomly initialized. In the E-step, expected counts of features are calculated at each hidden variable using the current estimate of the parameters conditional on the observations. These expected counts are then used to re-estimate the parameters, which is the M-step. The E-step and M-step are iteratively computed until convergence (Do and Batzoglou, 2008).

Let X be the observations and Z be the underlying hidden variables. EM tries to solve for the parameters θ that maximize the log likelihood of the observed data.

$$log\ P(X|\theta) = log \sum_{z} P(X, Z|\theta) \tag{5}$$

In the E-step, the algorithm finds a function that lower bounds $logP(X|\theta)$ (Jordan and Bishop, 2004).

E-Step:

$$Q(\theta|\theta^{(t)}) = E_{P(Z|X,\theta^{(t)})}[log\ P(X, Z|\theta)] \tag{6}$$

The M-step calculates the new parameters from the maximized Q function in Equation 6.

M-Step:

$$\theta^{(t+1)} = argmax_{\theta}\ Q(\theta|\theta^{(t)}) \tag{7}$$

2.6 Repeated Expectation Maximization

Since the objective function optimized by the expectation maximization algorithm is non-convex, expectation maximization can lead to suboptimal learning on account of local maxima. The accuracy of any resultant model consequently varies widely. Since the final state reached depends on the initial set of parameters used (Laird et al, 1987), repeating expectation maximization with different initial parameters (selected at random) and selecting the best performing models (using a validation set) yields better overall accuracy and lowers the variance of the same (Zhao et al., 2011).

3 Three-Layer Models

In a 3-layer model, the root node represents the set of class labels $C = \{c_1, c_2 \ldots c_k\}$ and the leaf nodes represent the set of features $F = \{f_1, f_2 \ldots f_i\}$. And sandwiched between the two is a set of hidden nodes $H = \{h_1, h_2 \ldots h_j\}$.

$P(F|c)$ can be obtained from $P(F, H|c)$ by marginalizing over the hidden nodes, as follows.

$$P(F|c) = \sum_{h} P(F, h|c) \tag{8}$$

The chain rule gives us the following equation.

$$P(F, h|c) = P(F|h, c) \times P(h|c) \tag{9}$$

Substituting Equation 9 in Equation 8, we get Equation 10.

$$P(F|c) = \sum_{h} P(F|h, c)P(h|c) \tag{10}$$

The corresponding rule for individual features[1] can be similarly derived.

$$P(f|c) = \sum_{h} P(f|h, c)P(h|c) \tag{11}$$

By substituting Equation 10 in Equation 2, we arrive at Equation 12.

$$P(c|F) = \sum_{h} P(F|h, c)P(h|c)\frac{P(c)}{P(F)} \tag{12}$$

So the parameters that need to be learnt for classification are $P(F|h, c)$, $P(h|c)$ and $P(c)$. $P(F)$

[1]In the rest of the paper, we use capital letters to denote sets of variables (features, hidden nodes and classes), and lowercase letters for a specific variable (a feature, a hidden node or a class).

does not need to be learnt as it is independent of the class c and so is merely a normalization constant.

Once these parameters have been learnt, the probability of a class c given the features F can be computed in one of two ways:

1. Product of sums.
2. Sum of products.

We prove below that using the product of sums to compute $P(c|F)$ in a directed 3-layer multinomial hierarchical model trained using expectation maximization yields a linear decision boundary whereas using the sum of products results in a non-linear decision boundary.

3.1 Product of Sums

If you assume that each feature f is independent of every other feature conditional on a class c, then the joint probability of a set of features F given a class c can be written as a product of probabilities of individual features given the class.

$$P(F|c) = \prod_f P(f|c) \tag{13}$$

Substituting Equation 11 into Equation 13, we get Equation 14.

$$P(F|c) = \prod_f \sum_h P(f|h,c)P(h|c) \tag{14}$$

Substituting the above for $P(F|c)$ in Equation 2, we get

$$P(c|F) = \prod_f \left(\sum_h P(f|h,c)P(h|c) \right) \frac{P(c)}{P(F)} \tag{15}$$

3.2 Proof of Naive Bayes Equivalence

In this subsection, we show that expectation maximization over any data-set results in the learning of multinomial parameters that when combined using Equation 15 yield a classifier with the same classification performance as a multinomial naive Bayes classifier trained on the same data-set.

The equation for $P(c|F)$ for a naive Bayes classifier can be obtained by dividing both sides by $P(F)$ in Equation 4.

$$P(c|F) = P(F|c)\frac{P(c)}{P(F)} \tag{16}$$

By inspection of Equation 16 and Equation 15, we see that the right hand sides of these equations must be equal for both classifiers (the 3-layer hierarchical Bayesian and naive Bayes classifiers) to perform identically.

By equating the right sides of Equation 16 and Equation 15, we obtain Equation 17[2].

$$P_{nb}(F|c) = \prod_f \left(\sum_h P_{hb}(f|h,c)P_{hb}(h|c) \right) \tag{17}$$

By substituting Equation 13 in Equation 17 we get Equation 18.

$$\prod_f P_{nb}(f|c) = \prod_f \left(\sum_h P_{hb}(f|h,c)P_{hb}(h|c) \right) \tag{18}$$

Equation 18 is satisfied if each of the factors on the right hand side is equal to each of the corresponding factors on the left hand side, as shown in Equation 19.

$$P_{nb}(f|c) = \sum_h P_{hb}(f|h,c)P_{hb}(h|c) \tag{19}$$

So, to prove that the multinomial "product of sums" hierarchical Bayes classifier is equivalent to the multinomial naive Bayes classifier (and therefore is only capable of learning a linear hyperplane separator between classes), all we have to establish is that Equation 19 holds for a given learning procedure.

Let the count of a feature f in a class c be $C(f)$, and let the distribution learnt be a multinomial distribution. For any class c and features $\phi \in F$, the class-conditional feature probability learnt by a naive Bayes classifier is that given by the maximum likelihood estimator as shown in Equation 20.

$$P_{nb}(f|c) = \frac{C(f)}{\sum_{\phi \in F} C(\phi)} \tag{20}$$

[2]The subscripts in P_{nb} and P_{hb} indicate that these were probabilities estimated for the naive Bayes and the hierarchical Bayesian classifiers respectively.

For the hierarchical Bayesian classifier, the expectation maximization algorithm learns P_{hb} differently depending on whether hard expectation maximization or soft expectation maximization is used.

3.2.1 Hard Expectation Maximization

In hard expectation maximization, each data point is assigned to one of the hidden nodes $h \in H$ in the E-step. Let $C_h(f)$ of a feature $f \in F$ denote the count of the feature in the data points assigned to a hidden node h of a class c, and the set F_h denote the set of features assigned to that hidden node.

For any feature f, $C(f)$ and $C_h(f)$ are related as follows.

$$C(f) = \sum_{h \in H} C_h(f) \tag{21}$$

It is now possible to write P_{hb} in terms of $C(f)$ as shown in Equation 22 and Equation 23.

$$P_{hb}(h|c) = \frac{\sum_{\theta \in F_h} C_h(\theta)}{\sum_{\phi \in F} C(\phi)} \tag{22}$$

$$P_{hb}(f|h,c) = \frac{C_h(f)}{\sum_{\theta \in F_h} C_h(\theta)} \tag{23}$$

By plugging Equation 22 and Equation 23 into the right hand side of Equation 19 we get

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \sum_h \frac{C_h(f)}{\sum_{\phi \in F} C(\phi)} \tag{24}$$

Since the sum of counts $C_h(f)$ over hidden nodes $h \in H$ is equal to $C(f)$ (Equation 21), the above equation is equal to

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \frac{C(f)}{\sum_{\phi \in F} C(\phi)} \tag{25}$$

Since the right hand sides of Equation 20 and Equation 25 are equal, we have proved that Equation 19 holds when the probabilities P_{nb} and P_{hb} are learnt using hard expectation maximization.

3.2.2 Soft Expectation Maximization

In soft expectation maximization, each data point is shared by all of the hidden nodes $h \in H$ of a class c in the E-step, with each hidden node's share of ownership being $m_h(d)$. Since $m_h(d)$ represents a probability distribution over h, ($m_h(d) = P(h|d, \Theta)$ where Θ are the parameters of the model), the sum of $m_h(d)$ over all hidden nodes is 1.

$$\sum_{h \in H} m_h(d) = 1 \tag{26}$$

Let $C_d(f)$ of a feature $f \in F_d$ denote the count of the feature f in data point d belonging to a class c, where F_d is the set of features in data point d.

Now the probabilities P_{hb} can be computed as follows.

$$P_{hb}(h|c) = \frac{\sum_d \left(m_h(d) \sum_{\theta \in F_d} C_d(\theta) \right)}{\sum_{\phi \in F} C(\phi)} \tag{27}$$

$$P_{hb}(f|h,c) = \frac{\sum_d m_h(d) C_d(f)}{\sum_d \left(m_h(d) \sum_{\theta \in F_d} C_d(\theta) \right)} \tag{28}$$

By plugging Equation 27 and Equation 28 into the right hand side of Equation 19 we get

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \sum_h \frac{\sum_d m_h(d) C_d(f)}{\sum_{\phi \in F} C(\phi)} \tag{29}$$

Taking the summation over h inside the summation over d we obtain Equation 30.

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \frac{\sum_d \sum_h m_h(d) C_d(f)}{\sum_{\phi \in F} C(\phi)} \tag{30}$$

Since $\sum_h m_h(d) = 1$ (26), the above equation reduces to Equation 31.

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \frac{\sum_d C_d(f)}{\sum_{\phi \in F} C(\phi)} \tag{31}$$

However, $\sum_d C_d(f) = C(f)$, so we obtain

$$\sum_h P_{hb}(f|h,c) P_{hb}(h|c) = \frac{C(f)}{\sum_{\phi \in F} C(\phi)} \tag{32}$$

Since the right hand sides of Equation 20 and Equation 32 are equal, we have proved that Equation 19 holds when the probabilities P_{nb} and P_{hb} are learnt using soft expectation maximization.

Thus we have proved that a hierarchical Bayes classifier that computes $P(c|F)$ by taking the product of the sums of $P_{hb}(f|h,c) \times P_{hb}(h|c)$ is equivalent to a naive Bayes classier and is therefore, like a naive Bayes classifier, only capable of learning a linear decision boundary.

3.3 Sum of Products

If you assume that each feature f is independent of other features conditional on any hidden node h belonging to any class c of the set C, then the joint probability of a set of features F given a hidden node h and its class c, denoted as $P(F|h,c)$, can be written as a product of probabilities of individual features given the hidden node and class.

$$P(F|h,c) = \prod_f P(f|h,c) \tag{33}$$

Substituting Equation 33 for $P(F|h,C)$ in Equation 12, we get Equation 34.

$$P(c|F) = \sum_h \left(\prod_f P(f|h,c) \right) P(h|c) \frac{P(c)}{P(F)} \tag{34}$$

Equation 34 allows you to compute the posterior probability $P(c|F)$ by taking the sum over the hidden nodes of the products of $P(f|h,c)$.

We show below that this method of computing the posterior probability allows a multinomial hierarchical Bayesian classifier to learn a non-linear decision boundary.

3.4 Demonstration of Non-Linearity

When the XOR function is scaled by 4 and shifted by 1 unit along the x and y axes, the resultant function maps the tuples $(x = 5, y = 5)$ and $(x = 1, y = 1)$ to 0 and the tuples $(x = 5, y = 1)$ and $(x = 1, y = 5)$ to 1 as shown in Figure 1.

We can verify by observation that the convex hulls around the data points in each of the categories intersect. The convex hull around category 1 is the line between $\{5,1\}$ and $\{1,5\}$. The convex hull around category 0 is the line between $\{1,1\}$ and $\{5,5\}$. These lines intersect at $\{3,3\}$, and therefore the categories are not linearly separable.

Since the categories are not linearly separable, if a classifier is capable of learning (correctly classifying the points of) this XOR function it must be a non-linear classifier.

In this subsection, we demonstrate experimentally that a 3-layer multinomial hierarchical Bayesian classifier with a "sum-of-products" posterior and 4 hidden nodes (2 per class) can learn this function.

Figure 2 shows the outputs of various classifiers trained on the XOR function described in Figure 1 when run on random points in the domain of the function. The small dots were placed at all points classified as 1.

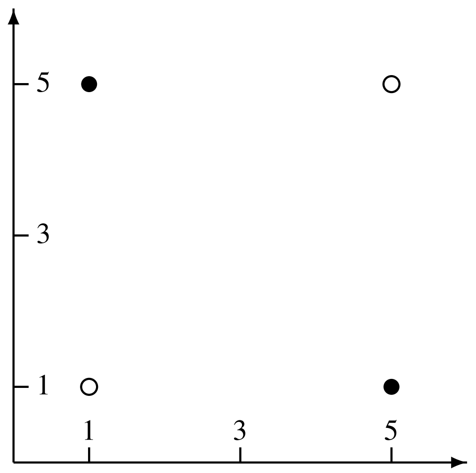


Figure 1: A 2-D plot of the shifted and scaled XOR function, where the filled circles stand for points that map to 1 and the unfilled circles for points that map to 0.

3.4.1 Naive Bayes

It can be seen from Figure 2a that the boundary learnt by the naive Bayes classifier is a straight line through the origin.

It can be seen that no matter what the angle of the line is, at least one point of the four will be misclassified.

In this instance, it is the point at $\{5,1\}$ that is misclassified as 0 (since the clear area represents the class 0).

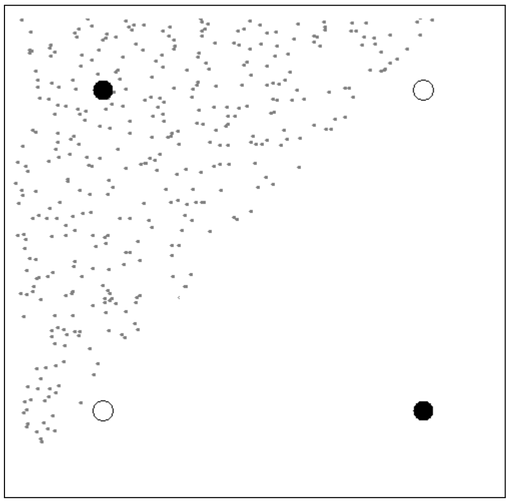
3.4.2 Multinomial Non-Linear Hierarchical Bayes

The decision boundary learnt by a **multinomial** non-linear hierarchical Bayes classifier (one that computes the posterior using a sum of products of the hidden-node conditional feature probabilities) is shown in Figure 2b.

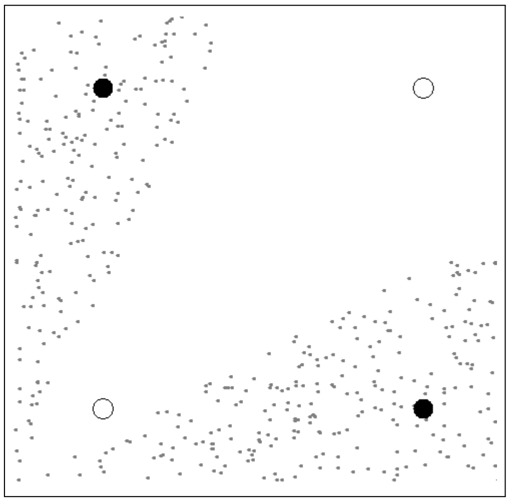
The boundary consists of two straight lines passing through the origin. They are angled in such a way that they separate the data points into the two required categories.

All four points are classified correctly since the points at $\{1,1\}$ and $\{5,5\}$ fall in the clear conical region which represents a classification of 0 whereas the other two points fall in the dotted region representing class 1.

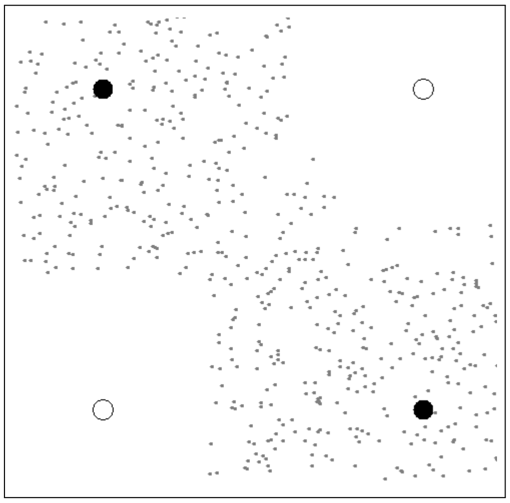
Therefore, we have demonstrated that the multinomial non-linear hierarchical Bayes classifier can learn the non-linear function of Figure 1.



(a) Naive Bayes



(b) Multinomial Non-Linear Hierarchical Bayes



(c) Gaussian Non-Linear Hierarchical Bayes

Figure 2: Decision boundaries learnt by classifiers trained on the XOR function of Figure 1. The dotted area represents what has been learnt as the category 1. The clear area represents category 0.

3.4.3 Gaussian Non-Linear Hierarchical Bayes

The decision boundary learnt by a **Gaussian** non-linear hierarchical Bayes classifier is shown in Figure 2c.

The boundary consists of two quadratic curves separating the data points into the required categories.

Therefore, the Gaussian non-linear hierarchical Bayes classifier can also learn the function depicted in Figure 1.

Thus, we have demonstrated that 3-layer hierarchical Bayesian classifiers (multinomial and Gaussian) with a "sum-of-products" posterior and 4 hidden nodes (2 per class) can learn certain non-linear functions.

3.5 Intuition on Learning Arbitrary Non-Linear Functions

Non-linear hierarchical Bayesian models based on the multinomial distribution are restricted to hyperplanes that pass through the origin. Each hidden node represents a linear boundary passing through the origin. Models based on the Gaussian distribution are not so restricted. Each hidden node in their case is associated with a quadratic boundary.

Assuming that the hidden nodes act as cluster centroids and fill the available hyperspace of a given class (for a suitable probability distribution), one might expect the dominance of the closest hidden nodes - dominance of one hidden node for text datasets is likely on account of extreme posterior probabilities (Eyheramendy et al, 2003) - to result in the piece-wise reconstruction of arbitrary boundaries in the case of Gaussian models. The dominance of the nearest hidden nodes (in terms of angular distances) is likely to result in appropriate radial boundaries in the case of multinomial models.

We now demonstrate empirically that a classifier capable of learning non-linear boundaries can outperform linear classifiers on certain text and tuple classification tasks.

4 Experimental Results

We conducted four sets of experiments to evaluate the performance of non-linear hierarchical Bayesian classifiers trained using repeated soft expectation maximization. In the first three experiments, the multinomial version was used and in the last, the Gaussian version was used.

With both the naive Bayes classifier and the hierarchical Bayesian models, the smoothing used was Laplace smoothing. The SVM classifier used a linear kernel.

4.1 Large Movie Review Dataset

The first experiment was run on the large movie review dataset (Maas et al, 2011) which consists of $50,000$ movie reviews labelled as either positive or negative (according to whether they expressed positive or negative sentiment about a movie).

The training set size was varied between $5,000$ and $35,000$ in steps of $5,000$ (with half the reviews drawn from positively labelled data and the other half drawn from negatively labelled data). Of the remaining data, we used $5,000$ reviews ($2,500$ positive and $2,500$ negative) as a validation set. The rest were used as test data.

The accuracies for different training set sizes and hidden nodes are as shown in Figure 3. The accuracies obtained when training on $25,000$ reviews and testing on $20,000$ (with the remaining 5000 documents of the test set used for validation) are shown in Table 1 .

Classifier	**Accuracy**
Naive Bayes	0.7964 ± 0.0136
MaxEnt	0.8355 ± 0.0149
SVM	0.7830 ± 0.0128
Non-linear Hierarchical Bayes	$\mathbf{0.8438} \pm 0.0110$

Table 1: Accuracy on the movie reviews dataset when trained on 25,000 reviews.

4.2 Single-Label Text Categorization Datasets

We tested the performance of the various classifiers (the hierarchical Bayes classifier was configured with 2 hidden nodes per class) on the Reuters R8, R52 and 20 Newsgroups datasets preprocessed and split as described in (Cardoso-Cachopo, 2007). 10% of the training data was held back for validation. The results are shown in Table 2.

4.3 Query Classification Dataset

For this experiment, we used a smaller corpus of 1889 queries classified into 14 categories (Thomas Morton, 2005). Five different random orderings of

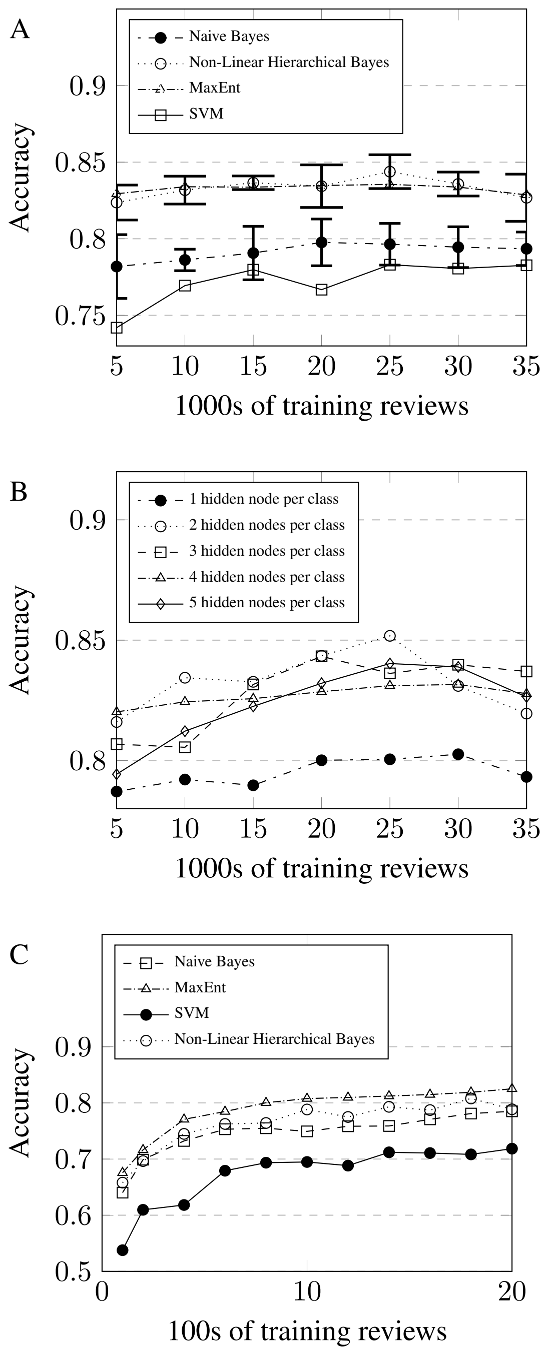


Figure 3: Evaluations on the movie reviews dataset. Plot A shows the performance of various classifiers on the dataset (the multinomial non-linear hierarchical Bayes classifier being configured with 2 hidden nodes per class and trained using 50 repetitions of 4 iterations of soft expectation maximization). Plot B compares the accuracies of multinomial non-linear hierarchical Bayes classifiers with different numbers of hidden nodes. Plot C charts the performance of the classifiers on much smaller quantities of data than in plot A (the 2-hidden-node multinomial non-linear hierarchical Bayes classifier being trained with 10 repetitions of 5 expectation maximization iterations).

Classifier	R8	R52	20Ng
Naive Bayes	0.955	0.916	0.797
MaxEnt	0.953	0.866	0.793
SVM	0.946	0.864	0.711
Non-Lin Hier Bayes	**0.964**	**0.921**	**0.808**

Table 2: Accuracy on Reuters R8, R52 and 20 Newsgroups.

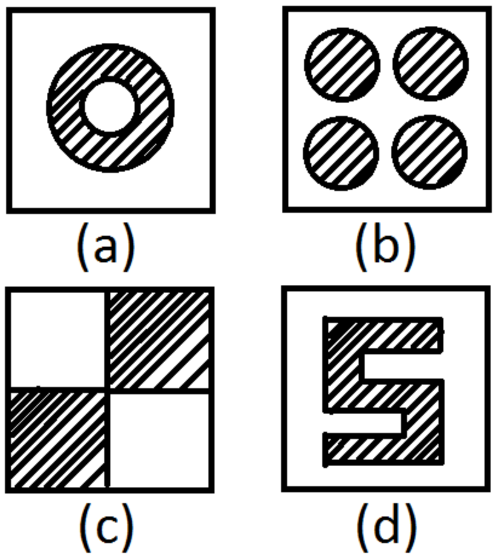


Figure 4: Artificial 2-dimensional dataset: a) ring b) dots c) XOR d) S

the data in the corpus were used, each with 1400 training, 200 validation and 289 test queries.

The accuracies of different classifiers (including a hierarchical Bayes classifier with 4 hidden nodes per class trained through 50 repetitions of 10 iterations of expectation maximization) on the query dataset are as shown in Table 3. The error margins are large enough to render all comparisons on this dataset devoid of significance.

Classifier	Accuracy
Naive Bayes	0.721 ± 0.018
MaxEnt	0.667 ± 0.028
SVM	0.735 ± 0.032
Non-Linear Hier Bayes	0.711 ± 0.032

Table 3: Accuracy on query classification.

4.4 Artificial 2-Dimensional Dataset

A total of 2250 points were generated at random positions (x, y) inside a square of width 500. Those falling inside each of the four shaded shapes shown in Figure 4 were labelled 1 and the rest of the points (falling outside the shaded areas of the square) were labelled 0.

For each shape, 1000 of the points were used for training, 250 as the validation set and the remaining 1000 as the test set. Naive Bayes and non-linear hierarchical Bayes classifiers that assumed the Gaussian distribution were used.

The Gaussian naive Bayes (GNB) classifier and the non-linear Gaussian hierarchical Bayes (GHB) classifier (10 hidden nodes per class), trained with 10 repetitions of 100 iterations of expectation maximization were tested on the artificial dataset. Their accuracies are as shown in Table 4.

Shape	GNB	GHB
ring	0.664	0.949
dots	0.527	0.926
XOR	0.560	0.985
S	0.770	0.973

Table 4: Accuracy on the artificial dataset.

4.5 Discussion

We see from Table 1 that the multinomial non-linear hierarchical Bayes classifier significantly outperforms the naive Bayes and SVM classifiers on the movie reviews dataset. We see from Table 2 that its performance compares favourably with that of other classifiers on the Reuters R8, the Reuters R52 and the 20 Newsgroups datasets.

We also observe from Plot B of Figure 3 that multinomial non-linear hierarchical Bayes classifiers with 2, 3, 4 and 5 hidden nodes outperform 1 hidden node on that dataset.

Finally, we observe that the artificial dataset is modeled far better by a Gaussian non-linear hierarchical Bayes classifier than by a Gaussian naive Bayes classifier.

5 Conclusions

We have shown that generative classifiers with a hidden layer are capable of learning non-linear decision boundaries under the right conditions (independence assumptions), and therefore might be said to be capable of deep learning. We have also shown experimentally that multinomial non-linear hierarchical Bayes classifiers can outperform some linear classification algorithms on some text classification tasks.

References

Aixin Sun and Ee-Peng Lim. 2001. Hierarchical Text Classification and Evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, 521–528.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL-HLT2011*, 142–150.

Alex Krizhevsky, and Geoffrey E. Hinton. 2011. Using very deep autoencoders for content-based image retrieval. *ESANN*.

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. 2014. Learning polynomials with neural networks. *ICML*.

Ana Cardoso-Cachopo. 2007. Improving Methods for Single-label Text Categorization. *PhD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa*.

Asja Fischer and Christian Igel. 2012. An Introduction to Restricted Boltzmann Machines. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings, 7441:14–36.

Andrew Y. Ng and Michael I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14 (NIPS'01)*, 841–848.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*. 1–38. JSTOR.

Chuong B. Do, and Serafim Batzoglou. 2008. What is the expectation maximization algorithm?. *Nature biotechnology*. 26(8):897-899. Nature Publishing Group.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT Press, Cambridge, MA.

David E. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. 1986. Learning Internal Representations by Error Propagation. In *Parallel distributed processing: explorations in the microstructure of cognition*. 1:318-362. MIT Press, Cambridge, MA.

David H. Ackley, Geoffrey E. Hinton, Terrence J. Sejnowski. 1985. A learning algorithm for Boltzmann machines. In *Cognitive science*. 9(1):147-169.

George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. In *Mathematics of control, signals and systems*. 2(4):303–314

Hao Wang and Dit-Yan Yeung. 2016. Towards Bayesian Deep Learning: A Survey. *CoRR*.

Martin J.Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Marvin Minsky and Seymour Papert. 1988. *Perceptrons: an introduction to computational geometry (expanded edition)*. MIT Press, Cambridge, MA.

Michael I. Jordan, and Chris Bishop. 2004. *An introduction to graphical models*. Progress.

Miguel E. Ruiz, and Padmini Srinivasan. 2002. Hierarchical Text Categorization Using Neural Networks. In *Information Retrieval*, 5(1):87–118.

Nan Laird, Nicholas Lange, and Daniel Stram. 1987. Maximum likelihood computations with repeated measures: application of the EM algorithm. *Journal of the American Statistical Association*, 82(397):97–105.

Paul Smolensky. 1986. Chapter 6: Information Processing in Dynamical Systems: Foundations of Harmony Theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 1:194281. MIT Press, Cambridge, MA.

Susana Eyheramendy, David D. Lewis, and David Madigan. 2003. On the Naive Bayes Model for Text Categorization. In *9th International Workshop on Artificial Intelligence and Statistics*.

Thomas Morton. 2005. Using semantic relations to improve information retrieval. *Dissertations available from ProQuest*. Paper AAI3197718.

Qinpei Zhao, Ville Hautamaki, and Pasi Franti. 2011. RSEM: An Accelerated Algorithm on Repeated EM. *2011 Sixth International Conference on Image and Graphics (ICIG)*. IEEE.

Shivakumar Vaithyanathan, Jianchang Mao, and Byron Dom. 2000. Hierarchical Bayes for Text Classification. *PRICAI Workshop on Text and Web Mining*, 36-43.

Sriharsha Veeramachaneni, Diego Sona, and Paolo Avesani. 2005. Hierarchical Dirichlet Model for Document Classification. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, 928–935.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521:436-444.

Yoshua Bengio. 2009. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1-127.

Zhibin Liao and Gustavo Carneiro. 2015. The use of deep learning features in a hierarchical classifier learned with the minimization of a non-greedy loss function that delays gratification. *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE.