IJCAI'99. Best paper award.

Learning in Natural Language

Dan Roth[*]
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
danr@cs.uiuc.edu

Abstract

Statistics-based classifiers in natural language are developed typically by assuming a generative model for the data, estimating its parameters from training data and then using Bayes rule to obtain a classifier. For many problems the assumptions made by the generative models are evidently wrong, leaving open the question of why these approaches work.

This paper presents a learning theory account of the major statistical approaches to learning in natural language. A class of *Linear Statistical Queries (LSQ)* hypotheses is defined and learning with it is shown to exhibit some robustness properties. Many statistical learners used in natural language, including naive Bayes, Markov Models and Maximum Entropy models are shown to be LSQ hypotheses, explaining the robustness of these predictors even when the underlying probabilistic assumptions do not hold. This coherent view of when and why learning approaches work in this context may help to develop better learning methods and an understanding of the role of learning in natural language inferences.

1 Introduction

Generative probability models provide a principled way to the study of statistical classification in complex domains such as natural language. It is common to assume a generative model for such data, estimate its parameters from training data and then use Bayes rule to obtain a classifier for this model. In the context of natural language most classifiers are derived from probabilistic language models which estimate the probability of a sentence s, say, using Bayes rule, and then decompose this probability into a product of conditional probabilities according to the generative model assumptions.

$$\begin{aligned} Pr(s) &\doteq Pr(w_1, w_2, \dots w_n) = \Pi_{i=1}^{n} Pr(w_i|w_1, \dots w_{i-1}) \\ &\doteq \Pi_{i=1}^{n} Pr(w_i|h_i) \end{aligned}$$

where h_i is the relevant *history* when predicting w_i. Note that in this representation s can be any sequence of tokens, words, part-of-speech (pos) tags. etc.

This general scheme has been used to derive classifiers for a variety of natural language applications including speech applications [Rabiner, 1989], part of speech tagging [Kupiec, 1992; Schütze, 1995], word-sense disambiguation [Gale *et al.*, 1993], context-sensitive spelling correction [Golding, 1995] and others. While the use of Bayes rule is harmless, most of the work in statistical language modeling and ambiguity resolution is devoted to estimating terms of the form $Pr(w|h)$. The generative models used to estimate these terms typically make Markov or other independence assumptions. It is evident from looking at language data that these assumptions are often patently false and that there are significant global dependencies both within and across sentences. For example, when using (Hidden) Markov Model (HMM) as a generative model for the problem of part-of-speech tagging, estimating the probability of a sequence of tags involves assuming that the part of speech tag t_i of the word w_i is independent of other words in the sentence, given the preceding tag t_{i-1}. It is not surprising therefore that making these assumptions results in a poor estimate of the probability distribution density function[1]. However, classifiers built based on these false assumptions nevertheless seem to behave quite robustly in many cases.

In this paper we develop a learning theory account for this phenomenon. We show that a variety of models used for learning in Natural Language make their prediction using *Linear Statistical Queries (LSQ)* hypotheses. This is a family of linear predictors over a set of fairly simple features which are directly related to the independence assumptions of the probabilistic model assumed. We claim that the success of classification methods which are derived from incorrect probabilistic density estimation is due to the combination of two factors:

- The low expressive power of the derived classifier.
- Robustness properties shared by all linear statistical queries hypotheses.

[*]Research supported by NSF grants IIS-9801638 and SBR-9873450.

[1]Experimental evidence to that effect will not be included in this extended abstract for lack of space (but see Sec. 5).

We define the class of LSQ hypotheses and prove these claims. Namely, we show that since the hypotheses are computed over a feature space chosen so that they perform well on training data, learning theory implies that they perform well on previously unseen data, irrespective of whether the underlying probabilistic assumptions hold. We then show how different models used in the literature can be cast as LSQ hypotheses by selecting the statistical queries (i.e., set of features) appropriately and how this affects the robustness of the derived hypothesis.

The main contribution of the paper is in providing a unified learning theory account for a variety of methods that are widely used in natural language applications. The hope is that providing a coherent explanation for when and why learning approaches work in this context could help in developing better learning methods and a better understanding of the role of learning in natural language inferences.

We first present learning theory preliminaries and then define the class of linear statistical queries hypotheses and discuss their properties. In Sec. 4 we show how to cast known probabilistic predictors in this framework, and in Sec. 5 we discuss the difference between LSQ learning and other learning algorithms.

2 Preliminaries

In an instance of pac learning [Valiant, 1984], a learner needs to determine a close approximation of an unknown target function from labeled examples of that function. The unknown $\{0,1\}$-valued function g is assumed to be an element of a known function class $\mathcal{G}$ over the *instance space* $X = \{0,1\}^n$, and the examples are distributed according to some unknown probability distribution D on X. A learning algorithm draws a sample of labeled examples according to D and eventually outputs a hypothesis h from some hypothesis class $\mathcal{H}$. The error rate of the hypothesis h is defined to be $error(h) = Pr_D[h(x) \neq g(x)]$. The learner goal is to output, with probability at least $1-\delta$, a hypothesis $h \in \mathcal{H}$ whose error rate is at most ε, for the given error parameter ε and confidence parameter δ, using sample size that is polynomial in the relevant parameters. The learner is then called a *pac learner*. The algorithm is an *efficient pac learning algorithm* if this is done in time polynomial in the relevant parameters.

A more realistic variant of the pac-learning model, the *agnostic learning* model [Haussler, 1992; Kearns *et al.*, 1992], applies when we do not want to assume that the labeled training examples (x,l) arise from a target concept of an a-priori simple structure $g \in \mathcal{G}$. In this model one assumes that data elements (x,l) are sampled according to some arbitrary distribution D on $X \times \{0,1\}$. D may simply reflect the distribution of the data as it occurs "in nature" (including contradictions) without assuming that the labels are generated according to some "rule". The true error of the hypothesis h is defined to be $error_D(h) = Pr_{(x,l)\in D}[h(x) \neq l]$, and the goal of the learner is to compute, with high probability, a hypothesis $h \in \mathcal{H}$ with true error not larger than $\varepsilon + inf_{h\in\mathcal{H}} error_D(h)$. A learner that can carry out this task for any distribution D is called an *efficient (agnostic) pac learner* for hypothesis class $\mathcal{H}$ if the sample size $m(\varepsilon,\delta)$ and the time required to produce the hypothesis are polynomial in the relevant parameters.

In practice, one cannot compute the true error $error_D(h)$, and in many cases it may be hard to guarantee that the computed hypothesis h is close to optimal in $\mathcal{H}$. Instead, the input to the learning algorithm is a sample $S = \{(x^i, l^i)\}_{i=1}^m$ of m labeled examples, the learner tries to find a hypothesis h with a small *empirical error*

$$error_S(h) = |\{x \in S | h(x) \neq l\}| / |S|,$$

and hopes that it behaves well on future examples.

The hope that a classifier learned from a training set will perform well on previously unseen examples is based on the basic theorem of learning theory [Valiant, 1984; Vapnik, 1995] which, stated informally, guarantees that if the training data and the test data are sampled from the same distribution, good performance on large enough training sample guarantees good performance on the test data (i.e., good "true" error). This is quantified in the following *uniform convergence* result:

Theorem 2.1 ([Haussler, 1992]) *If the size of the training sample S is at least $m(\varepsilon,\delta) = \frac{1}{\varepsilon^2}(kVC(\mathcal{H}) + \ln\frac{1}{\varepsilon} + \ln\frac{1}{\delta})$ then with probability at least $1-\delta$,*

$$error_D(h) < error_S(h) + \varepsilon,$$

where k is some constant and $VC(\mathcal{H})$ is the VC-dimension of the class $\mathcal{H}$ [Vapnik, 1982], a combinatorial parameter which measures the richness of $\mathcal{H}$.

In practice the sample size $|S|$ may be fixed, and the result simply indicates how much can one count on the true accuracy of a hypothesis selected according to its performance on S. Also, the computational problem of choosing a hypothesis which minimizes the empirical error on S (or even approximates a minimum) may be hard [Kearns *et al.*, 1992; Höffgen and Simon, 1992]. Finally, notice that the main assumption in the theorem above is that the training data and the test data are sampled from the same distribution; Golding and Roth [1999] discuss this assumption in the NLP context.

3 Robust Learning

This section defines a learning algorithm and a class of hypotheses with some generalization properties, that we later show to capture many probabilistic learning methods used in NLP. The learning algorithm discussed here is a *Statistical Queries(SQ)* algorithm [Kearns, 1993]. An SQ algorithm can be viewed as a learning algorithm that interacts with its environment in a restricted way. Rather than viewing *examples*, the algorithm only requests the values of various *statistics* on the distribution of the labeled examples to construct its hypothesis. (E.g. "What is the probability that a randomly chosen labeled example (x,l) has $x_i = 0$ and $l = 1$"?) A *feature* is an indicator function $\chi : X \rightarrow \{0,1\}$ which defines a subset of the instance space - all those elements which are mapped to 1 by χ. $\mathcal{X}$ denotes a class of such indicator functions. $\mathcal{X}$ can be viewed as a transformation of the instance space X; each example $(x_1, \dots x_n) \in X$ is mapped to an example $(\chi_1, \chi_2, \dots \chi_{|\mathcal{X}|})$ in the new space. To simplify notation, we sometimes view a feature as an indicator function over the *labeled* instance space $X \times \{0,1\}$ and say that $\chi(x,l) = 1$ for examples $x \in \chi(X)$ with label l.

A *statistical query* has the form $[\chi, l, \tau]$, where $\chi \in \mathcal{X}$ is a feature, $l \in \{0,1\}$ is a further (optional) restriction imposed on the query and τ is an error parameter. A call to the SQ oracle returns an estimate $\hat{P}^D_{[\chi,l,\tau]}$ of

$$P^D_{[\chi,l]} = Pr_D\{(x,i) | \chi(x) = 1 \wedge i = l\} \qquad (1)$$

which satisfies $|\hat{P}_\chi - P_\chi| < \tau$. (When clear from the context we omit τ and/or l from this notation.) A *statistical queries algorithm* is a learning algorithm that constructs its hypothesis only using information received from an SQ oracle. An algorithm is said to use a query space $\mathcal{X}$ if it only makes queries of the form $[\chi, l, \tau]$ where $\chi \in \mathcal{X}$. As usual, an *SQ* algorithm is said to be a good learning algorithm if, with high probability, it outputs a hypothesis h with small error, using sample size that is polynomial in the relevant parameters.

In order to simulate the behavior of the SQ oracle on a query $[\chi, l, \tau]$ on the distribution D, we simply draw a large sample S of labeled examples (x, l) according to D and evaluate

$$Pr_S[\chi(x,l)] = \frac{|\{(x,l) : \chi(x,l) = 1|\}}{|S|}.$$

Chernoff bounds guarantee that the number of examples required to achieve tolerance τ with probability at least $1 - \delta$ is polynomial in $1/\tau$ and $\log 1/\delta$.

The SQ model of learning was introduced by Kearns [1993] and studied in [Decatur, 1993; Aslam and Decatur, 1995]. It was viewed as a tool for demonstrating that a pac learning algorithm is noise-tolerant. In particular, it was shown that learning with an SQ algorithm allows the learner to tolerate examples with noisy labels – labels which, with some probability $< 1/2$, are inconsistent with the target function (*classification noise*), as well as various forms of corrupted examples (malicious noise, attribute noise). A typical noise-tolerant learning result in the pac model states that if an SQ algorithm learns a concept class $\mathcal{G}$, in the sense that it produces a hypothesis with small true error, then this algorithm is noise tolerant in that it would have produces a hypothesis with small true error even when trained with noisy data. The running time of the algorithm and the number of data samples required depend on the tolerance required from the statistical queries. This, in turn, is determined by the noise level the algorithm is required to tolerate (up to some bounds; see the references for details). In the next section we consider a special SQ algorithm and prove a robustness result that may be more satisfying from a practical point of view.

3.1 Linear Statistical Queries Hypotheses

Let $\mathcal{X}$ be a class of features and $f : \{0,1\} \to \Re$ a function that depends only on the values $\hat{P}^D_{[\chi,l]}$ for $\chi \in \mathcal{X}$. A *Linear Statistical Queries (LSQ)* hypothesis predicts $l \in \{0,1\}$ given $x \in X$ when

$$l = argmax_{l \in \{0,1\}} \sum_{\chi \in \mathcal{X}} f_{[\chi,l]}(\{\hat{P}^D_{[\chi,l]}\}) \cdot \chi(x)$$

Clearly, the LSQ is a linear discriminator over the feature space $\mathcal{X}$, with coefficients f that are computed given (potentially all) the values $\hat{P}^D_{[\chi,l]}$. The definition generalizes naturally to non-binary classifiers by allowing l to range over a larger set of labels; in this case, the discriminator between predicting l and other values is linear. A learning algorithm that outputs an LSQ hypothesis is called an *LSQ algorithm*.

Example 3.1 *The naive Bayes predictor [Duda and Hart, 1973] is derived using the assumption that given the label $l \in L$ the features' values are statistically independent. Consequently, the Bayes optimal prediction is given by:*

$$h(x) = argmax_{l \in L} \Pi^m_{i=1} Pr(x_i|l) Pr(l),$$

where $Pr(l)$ denotes the prior probability of l (the fraction of examples labeled l) and $Pr(x_i|l)$ are the conditional feature probabilities (the fraction of the examples labeled l in which the ith feature has value x_i).

The LSQ definition implies:

Claim 3.1 *The naive Bayes algorithm is an LSQ algorithm over a set $\mathcal{X}$ which consists of $n+1$ features: $\chi_0 \equiv 1$, $\chi_i = x_i$ for $i = 1, \dots, n$ and where*

$$\begin{aligned} f_{[1,l]}() &= \log \hat{P}^D_{[1,l]} \\ &\textit{and} \\ f_{[x_i,l]}() &= \log \hat{P}^D_{[x_i,l]} / \hat{P}^D_{[1,l]}, \quad i = 1, \dots, n. \end{aligned}$$

We note that formalized this way, as is commonly done in NLP (e.g., [Golding, 1995]), features which are not active in the example are assumed unobserved *and are not taken into account in the naive Bayes estimation. It is possible to assume instead that features which are not active are* false *($\chi = 0$); this assumption yields an LSQ algorithm over the same features, with different coefficients (see, e.g., [Roth, 1998] for the derivation).*

The observation that the LSQ hypothesis is linear over $\mathcal{X}$ yiedls the first robustness result. VC dimension theory implies (via a variation of, e.g., [Anthony and Holden, 1993], and due to the restriction LSQ imposes on the coefficients) that:

Claim 3.2 *The VC dimension of the class of LSQ hypotheses is bounded above by $|\mathcal{X}|$. If all the features used are polynomials of the form $\chi = \Pi x_{i_1} x_{i_2} \dots x_{i_k}$ (which are all linearly independent), the VC dimension of the LSQ hypotheses is exactly $|\mathcal{X}|$.*

This, together with the basic theorem 2.1 implies:

Corollary 3.3 *For any LSQ learning algorithm and given a performance guarantee on previously unseen examples (sampled from the same distribution), the number of training examples required in order to maintain this performance scales linearly with the number of features used.*

3.2 Distributional Robustness

LSQ hypotheses are computed by an SQ algorithm; it calls the SQ oracle to estimate $P^D_{[\chi,l]}$ for all elements in $\mathcal{X}$, and applies f. Then, given an example $x \in X$ it evaluates the hypothesis to make the prediction.

As discussed above, the SQ model was introduced as a tool for demonstrating that a pac learning algorithm is noise-tolerant. In a practical learning situation we are given a *fixed* sample of data to use in order to generate a hypothesis, which will be tested on a different sample, hopefully "similar" to the training sample. Our goal is to show that an algorithm that is able to learn under these restrictions (namely, interacting only via SQs) is guaranteed to produce a *robust* hypothesis. Intuitively, since the SQ predictor does not really depend on specific examples, but only on some statistical properties of the data, and only within some tolerance, it should still perform well in the presence of data sampled according to a different distribution, as long as it has the same statistical properties within this tolerance.

Next we formalize this intuition and show that the basic results proved for the SQ model can be adapted to this situation. Formally, we assume that there is a *test sample*, S_{test}, generated according to some distribution D', and that we care about our hypothesis' performance on this sample. However, training is performed on a different sample, S_{train}, generated according to a different distribution D. We show that an SQ hypothesis trained over S_{train} can still perform well on S_{test}, if D and D' are not too different. A common distance measure between distributions is the *variation distance*[2] $d(D, D') = \frac{1}{2}\sum_{x\in X} |D(x) - D'(x)|$. It is easy to see that this measure is equivalent to $d(D, D') = \max_{A\subseteq X} |D(A) - D'(A)|$ which is more convenient for our purposes. A more biased measure, strictly smaller than the standard measures may also be used in this case:

$$d_{\mathcal{X}}(D, D') = \max_{\chi\in\mathcal{X}} |D(\chi) - D'(\chi)|,$$

yielding better robustness guarantees for the algorithm.

Theorem 3.4 *Let $\mathcal{A}$ be an $SQ(\tau, \mathcal{X})$ learning algorithm for a function class $\mathcal{G}$ over the distribution D and assume that $d_{\mathcal{X}}(D, D') < \gamma$ (for γ inversely polynomial in τ). Then $\mathcal{A}$ is also an $SQ(\tau, \mathcal{X})$ learning algorithm for $\mathcal{G}$ over D'.*

Proof: Since $\mathcal{A}$ is an $SQ(\tau, \mathcal{X}, D)$ algorithm, a hypothesis generated with estimates $\hat{P}^D_\chi$ that are τ close to P^D_χ, behaves well on D. In order for the hypothesis to behave well on D' we can simply run $\mathcal{A}$ with oracle calls of tolerance $\tau - \gamma$ to get $\hat{P}^D_\chi$ that is within $\tau - \gamma$ of P^D_χ. By the definition of the distance measure, the difference between the probability of events of interest occurring under D and D' is no more than γ and therefore

$$|\hat{P}^D_\chi - P^{D'}_\chi| \le |\hat{P}^D_\chi - P^D_\chi| + |P^D_\chi - P^{D'}_\chi| \le (\tau - \gamma) + \gamma.$$

Thus, this procedure simulates estimates that are within τ on D', implying a well behaved hypothesis on D'. To simulate the more accurate estimates, with tolerance $\tau - \gamma$, there is a need to use more samples, but only polynomially more due to the relation between γ and τ. ■

For the final point regarding robustness, observe that the proof shows the importance of the tolerance. In practice, the sample is given, and the robustness of the algorithm to different distributions depends on the sample's size. The richness of the feature class $\mathcal{X}$ plays an important role here. To ensure tolerance of τ for all features in χ one needs to use $O(\frac{\log|\mathcal{X}|}{\tau})$ samples (more generally, $O(\frac{VC(\mathcal{X})}{\tau})$ samples). For a given size sample, therefore, the use of simpler features in the LSQ representation provides better robustness. This, in turn, can be traded off with the ability to express the learned function with an LSQ over a simpler set of features.

4 Applications: Probabilistic Classifiers

In this section we cast a few widely used probabilistic classifiers as LSQ hypotheses, implying that they are subject to the properties discussed in Sec. 3.

4.1 Naive Bayes

As shown in Example 3.1, the naive Bayes predictor is an LSQ hypothesis. The implication is that this method, which has been widely used for natural language tasks [Gale *et al.*, 1993; Golding, 1995] satisfies:

Corollary 4.1 *The naive Bayes algorithm is a robust learning method in the sense of Cor. 3.3 and Thm 3.4.*

4.2 General Bayesian Classifier

The naive Bayes predictor can be generalized in several ways. First, one can consider a generalization of the naive model, in which *hidden variables* are allowed. The simplest generalization in this direction would assume that the distribution is generated by postulating a "hidden" random variable z. Having randomly chosen a value z for the hidden variable, we choose the value for each observable x_i independently of each other.

As a motivating example consider the case of disambiguation or tagging in natural language tasks, where the "context" or a tag may be viewed as hidden variables.

The prediction problem here is to predict the value of one variable, given known values for the remaining variables. One can show that the predictor in this setting is again an LSQ that uses essentially the same features as in the simple (observable) naive Bayes case discussed above (e.g., [Grove and Roth, 1998]).

[2] [Yamanishi, 1992] shows that this measure is equivalent to other well known measures such as the KL divergence.

A second generalization of naive Bayes is to assume that the independence structure around the predicted variable x is more complicated and is represented, for example, using a Bayesian network that is not as trivial as the naive Bayes network. In this case too, since we care only about predicting a value for a single variable having observed the other variables, it is not hard to verify that the Bayes optimal predictor is an LSQ hypothesis. The features in this case are polynomials $\chi = \Pi x_{i_1} x_{i_2} \dots x_{i_k}$ of degree that depends on the number of neighbors of x in the network. The details of this analysis are not included for lack of space. Instead, we concentrate on a special case that is of great interest in natural language.

4.3 Markov Models

Markov Models and Hidden Markov Models (HMMs) are a standard model used in language modeling in general and in disambiguation tasks such as part-of-speech (pos) tagging in particular. (See, e.g.,[Rabiner, 1989]; we will assume familiarity with them here). For a bi-gram pos tagger, the states of the HMM are pos tags. Transition probabilities are probabilities of a tag given the previous tag, and emission probabilities are probabilities of a word given a tag. The standard algorithms first estimate the transition and emission probabilities from training data; then they use the Markov model assumption in order to evaluate the probability of a particular part of speech sequence given a sentence. For example, for the part-of-speech sequence of the sentence `this can will rust` one computes

$$\begin{aligned} P(the = DT, can = NN, will = MD, rust = VB) \quad = \\ P(DT)P(NN|DT)P(MD|NN)P(VB|MD) \cdot \\ P(the|DT)P(can|NN)P(will|MD)P(rust|VB). \end{aligned}$$

Of course, this is correct only under the Markov assumption; namely, one assumes that given a tag for the ith word in the sentence, the value of the word is independent of other words or tags (yielding the term $P(can|NN)$), and the tag of the $i+1$ word is independent of preceding tags (yielding $P(MD|NN)$). In order to tag a sentence for pos one needs to maximize this term over all possible assignments of pos tags. This is usually done using a dynamic programming technique, e.g., a variation of the Viterbi algorithm.

Here we will be interested in the main computational step in this process – predicting the pos of a single word, given the sentence and assuming the pos of all the neighboring words as input. We will disregard the global optimization of finding the best simultaneous pos assignment to all words in the sentence[3]. We call this predictor the Markov predictor. For this prediction, it is easy to see that given an example

$$< (w_1 : t_1), \dots (w_{i-1} : t_{i-1}), (w_i : ?), (w_{i+1} : t_{i+1}), \dots >$$

[3]Interestingly, experimental studies have shown that sometimes the global optimization is not required and the local predictions actually produce better results [Delcher *et al.*, 1993; Roth and Zelenko, 1998].

we predict $pos(w_i) = t$ iff $t \in T$ maximizes

$$P(t|t_{i-1})P(w_i|t)P(t_{i+1}|t)$$

or, equivalently,

$$\log P(t|t_{i-1}) + \log P(w_i|t) + \log P(t_{i+1}|t)$$

where T is the set of all possible pos tags. In the above example, for the word `'can'` we will select the tag t that maximizes $P(t|DT) \cdot P(can|t) \cdot P(MD|t)$.

Let T, W denote the sets of all tags t and all words w, resp., which occur in the examples. In order to represent the Markov predictor as an LSQ algorithm we use $\mathcal{X} = \{T, W, 1\} \times \{T, W, 1\}$, the set of all singletons and pairs of words and tags. With this set of features and the notation introduced in Eq. 1 it is clear that the Markov predictor above can be written as an LSQ hypothesis:

$$t = argmax_{t \in T} \sum_{\chi \in \mathcal{X}} f_{[\chi,t]}(\{\hat{P}^D_{[\chi,t]}\}) \times \chi(x)$$

where

$$\begin{aligned} f_{[t_1 \times t_2, t_1]}() &= \log \hat{P}^D_{[t_1,t_2]} / \hat{P}^D_{[1,t_1]}, \\ f_{[t_1 \times t_2, t_2]}() &= \log \hat{P}^D_{[t_1,t_2]} / \hat{P}^D_{[1,t_2]}, \\ f_{[w \times t, t]}() &= \log \hat{P}^D_{[w,t]} / \hat{P}^D_{[1,t]}, \end{aligned}$$

and otherwise, $f_{[\chi,t]}() \equiv 0$.

Notice that in this formalization each word in the sentence yields a single training example and features are computed with respect to the word for which the tag is to be predicted. We get that:

Claim 4.2 *The Markov predictor is an LSQ algorithm over a set $\mathcal{X}$ of singletons and pairs of words and tags. Therefore, the Markov predictor is a robust learning method in the sense of Cor. 3.3 and Thm 3.4. This fact naturally generalizes to higher order Markov predictors by increasing the expressivity of the features.*

4.4 Maximum Entropy Models

Maximum Entropy (ME) models [Jaynes, 1982] are exponential distributions which satisfy given constraints. Constraints are what we have called *features* and the distribution is defined in terms of the *expected value* of the feature over the training set. These models have been used successfully to generate predictors for several disambiguation tasks [Ratnaparkhi *et al.*, 1994; Ratnaparkhi, 1997], typically by estimating the ME probability model for the conditional probability $p(w|h)$ (as in Sec. 1).

It can be shown that the distribution that maximizes the entropy and is consistent with a set of constraints (features) $\chi \in \mathcal{X}$ must have the form $P_{ME} = c\Pi_{j=1}^{k} \alpha_j^{\chi_j(x)}$, where c is a normalization constant and the α_j are the model parameters. Each parameter α_j corresponds to exactly one feature χ_j and can be viewed as the weight of that feature.

Predictions using the ME model are performed as follows. Given that the probability distribution P_{ME} has been estimated as above, and given the observed sentence s we would predict

$$h(s) = argmax_{l \in L} \Pi_{j=1}^{k} \alpha_j^{\chi_j(s,l)}.$$

Equivalently, by taking logarithms, we get a linear decision surface over the feature space.

However, while the optimal coefficients for an ME predictor are also a function of the features' statistics over the training data, they cannot be written explicitly. Instead, an iterative algorithm (e.g., [Darroch and Ratcliff, 1972]) that approximates the optimal coefficients (but is not known to converge efficiently) is required. The ME predictor is thus a *Linear* predictor but, strictly speaking, in its optimal form is not an SQ algorithm.

5 LSQ vs. Consistent Learning

In this section we discuss an important difference between an LSQ algorithm and "standard" learning algorithms, using the naive Bayes classifier as a running example. In most cases, algorithms studied in Machine Learning search for a hypothesis that is (almost) consistent with the training data and then rely on Thm 2.1 to guarantee good performance in the future. An LSQ algorithm, on the other hand, need not seek consistency. It simply computes a hypothesis from the responses of the *SQ* oracle and uses that hypothesis to make future predictions. Indeed, it is easily verified that the hypothesis produced by, say, a naive Bayes algorithm may not be consistent with the data that was used to compute it. Nevertheless, when the data satisfies the probabilistic assumptions used to derive the predictor, namely features' values are independent given the label, this hypothesis is the optimal predictor for future examples (although it may still make many mistakes). However, when the probabilistic assumptions do not hold, the success of the predictor is directly related to its success on the training data.

Consistency (or almost consistency) is not a trivial requirement. In fact, it is hard to come up with structural properties of classes for which naive Bayes is an optimal predictor. Even when the target concept is very simple - the class of Boolean conjunctions - consistency is not achieved in general, and the naive Bayes algorithm is not a good predictor for this class[4].

Example 5.1 *Consider the Boolean conjunction* $f = x_1 \wedge x_2$ *over over* $\{x_1, x_2, \ldots x_n\}$. *Assume that examples are generated according to the following distribution: Exactly half of the examples are positive. Over the positive examples necessarily* $x_1 = x_2 = 1$ *and we define* $x_3 = x_4 = x_5$*, and that this value is* 1 *with probability* $1/2$. $x_6, \ldots x_n$ *are* 1 *with probability* $1/2$*, independently of anything else. Over the negative examples* x_1 *is* 1 *with probability* $1/2$ *and* $x_2 \neq x_1$. $x_3 = x_4 = x_5$ *and this value is* 1 *with probability* $7/8$. $x_6, \ldots x_k$ *are* 1 *with probability* $1/2$*, independently of anything else.*

In this way we have that $p(x_1|l = 1) = p(x_2|l = 1) = 1$ *and* $p(x_i = 1|l = 1) = 1/2$ *for* $i = 3, \ldots k$. *For the negative examples we have that* $p(x_i = 1|l = 0) = 7/8$ *for* $i = 3, 4, 5$ *and* $p(x_i|l = 0) = 1/2$ *for* $i \neq 3, 4, 5$.

It is now easy to see that while the naive Bayes algorithm does not make mistakes on negative examples of a conjunction, it will predict incorrectly on half of the positive examples, all those for which $x_3 = x_4 = x_5 = 1$.

Clearly, this behavior is indicative of the performance of this predictor on any data presented to it which is sampled according to this distribution.

Practitioners, however, do not simply compute the LSQ hypothesis and use it, confident that it performs well since (when the assumptions of the underlying model hold) it represents the maximum likelihood hypothesis. Instead, they spend time choosing a feature set with which the predictor performs well on the training set. Once this is achieved, according to the results in Sec. 3, the predictor behaves well on previously unseen examples, but this happens *regardless* of whether the underlying probabilistic assumptions hold[5]. The only limitation on growing the feature set is the relation between the expressiveness of these features and the sample size required to achieve small tolerance, which we have alluded to. The role of the probabilistic assumptions in this case may be viewed as supplying guidance for selecting good features.

The question of why it is that even in seemingly hard NLP tasks, the search for good features is fairly simple, making it unnecessarily to resort to very expressive features, is an orthogonal question that we address in a companion paper. For the sake of the argument developed here, it is sufficient to realize that all the algorithms presented rely on their good behavior on the training set, and thus behave well even if the probabilistic assumptions that were used to derive the predictor do not hold.

6 Conclusion

In the last few years we have seen a surge of empirical work in natural language. A significant part of this work is done by using statistical machine learning techniques. Roth [1998] has investigated the relations among some of the commonly used methods and taken preliminary steps towards developing a better theoretical understanding for why and when different methods work. This work continues to develop the theoretical foundations of learning in natural language, focusing on the study of

[4] Domingo&Pazzani claim in a 1998 MLJ paper that naive Bayes is optimal for Boolean conjunctions and disjunctions. Their proof, however, holds only for product distributions, for which the naive Bayes assumptions hold. The following example may be viewed as a counterexample to a more general interpretation of their claim.

[5] Experimental evidence to this effect is presented in [Golding and Roth, 1999]. Two sets of features are compared. The one which provides a better conditional probability estimation, by virtue of better satisfying the independence assumption, nevertheless supports significantly worse performance on the prediction task.

probabilistic density estimation models and their use in performing natural language predictions.

In addition to providing better learning techniques, developing an understanding for when and why learning works in this context is a necessary step in studying the role of learning in higher-level natural language inferences.

Acknowledgments

I am grateful to Jerry DeJong, Lenny Pitt and anonymous referees for their useful comments on an earlier version of this paper.

References

[Anthony and Holden, 1993] M. Anthony and S. Holden. On the power of polynomial discriminators and radial basis function networks. In *Proc. 6th Annu. Workshop on Comput. Learning Theory*, pages 158–164. ACM Press, New York, NY, 1993.

[Aslam and Decatur, 1995] J. A. Aslam and S. E. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 437–446. ACM Press, New York, NY, 1995.

[Darroch and Ratcliff, 1972] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[Decatur, 1993] S. E. Decatur. Statistical queries and faulty PAC oracles. In *Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*, pages 262–268. ACM Press, 1993.

[Delcher *et al.*, 1993] A. Delcher, S. Kasif, H. Goldberg, and W. Xsu. Application of probabilistic causal trees to analysis of protein secondary structure. In *National Conference on Artificial Intelligence*, pages 316–321, 1993.

[Duda and Hart, 1973] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[Gale *et al.*, 1993] W. Gale, K. Church, and D. Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1993.

[Golding and Roth, 1999] A. R. Golding and D. Roth. A winnow based approach to context-sensitive spelling correction. *Machine Learning*, 1999. Special issue on Machine Learning and Natural Language; . Preliminary version appeared in ICML-96.

[Golding, 1995] A. R. Golding. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the 3rd workshop on very large corpora, ACL-95*, 1995.

[Grove and Roth, 1998] A. Grove and D. Roth. Linear concepts and hidden variables: An empirical study. In *Neural Information Processing Systems*. MIT Press, 1998.

[Haussler, 1992] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, September 1992.

[Höffgen and Simon, 1992] 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.

[Jaynes, 1982] E. T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, September 1982.

[Kearns *et al.*, 1992] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 341–352. ACM Press, New York, NY, 1992.

[Kearns, 1993] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 392–401, 1993.

[Kupiec, 1992] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6:225–242, 1992.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[Ratnaparkhi *et al.*, 1994] A. Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *ARPA*, Plainsboro, NJ, March 1994.

[Ratnaparkhi, 1997] A. Ratnaparkhi. A linear observed time statistical parser based on maximum entropy models. In *EMNLP-97, The Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10, 1997.

[Roth and Zelenko, 1998] D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th International Conference on Computational Linguistics*, pages 1136–1142, 1998.

[Roth, 1998] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. National Conference on Artificial Intelligence*, pages 806–813, 1998.

[Schütze, 1995] H. Schütze. Distributional part-of-speech tagging. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics*, 1995.

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[Vapnik, 1982] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[Vapnik, 1995] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[Yamanishi, 1992] K. Yamanishi. A learning criterion for stochastic rules. *Machine Learning*, 1992.