2006

Using Query-Specific Variance Estimates to Combine Bayesian Classifiers

Chi-Hoon Lee

Russell Greiner

Shaojun Wang
Wright State University - Main Campus, shaojun.wang@wright.edu

Using Query-Specific Variance Estimates to Combine Bayesian Classifiers

Chi-Hoon Lee CHIHOON@CS.UALBERTA.CA
Russ Greiner GREINER@CS.UALBERTA.CA
Shaojun Wang SWANG@CS.UALBERTA.CA
Dept of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E1, Canada

Abstract

Many of today's best classification results are obtained by combining the responses of a set of base classifiers to produce an answer for the query. This paper explores a novel "query specific" combination rule: After learning a set of simple belief network classifiers, we produce an answer to each query by combining their individual responses, using weights based inversely on their respective *variances* around their responses. These variances are based on the uncertainty of the network parameters, which in turn depend on the training datasample. In essence, this variance quantifies the base classifier's confidence of its response to this query. Our experimental results show that these "mixture-using-variance belief net classifiers" MUVs work effectively, especially when the base classifiers are learned using balanced bootstrap samples and when their results are combined using James-Stein shrinkage. We also found that our variance-based combination rule performed better than both bagging and AdaBoost, even on the set of base classifiers produced by AdaBoost itself. Finally, this framework is extremely efficient, as both the learning and the classification components require only straight-line code.

1. Introduction

Many tasks — including fault diagnosis, pattern recognition and forecasting — can be viewed as *classification*, as each requires assigning the class ("label") to a given instance (Mitchell, 1997; Hastie et al., 2002). In many cases, the classification function is not known *a priori*. If there are labeled data sets corresponding to this function, we can then try to *learn* the classifier.

Ensemble methods, which involves first learning a number of simple classifiers then combining their responses in some principled manner to answer a query, have proven to be very effective (Bauer & Kohavi, 1999; Opitz & Maclin, 1999). These ensemble systems can use belief nets (Pearl, 1988; Inza et al., 2005) as the base classifiers. This paper proposes a new way to combine the responses of these belief-net based classifiers to a given query, based on their respective *variances* around their responses to the given query, which in turn are based on the datasamples used to learn these classifiers. Informally, if a classifier is learned from a datasample that includes instances that resemble the current query, we expect the classifier's response to be fairly certain, and so have a small variance. By contrast, this variance would be large from a classifier that was based on instances that were different from the query. (We provide a more formal description in Section 3.2.1.) Our combination rule then uses to these "confidences" to weigh the responses.

As an example, imagine classifier H_A states the probability of $+c$ (given some evidence) is 0.3 ± 0.1 and of $-c$ is 0.7 ± 0.1, while H_B thinks the answer should be $+c$ with probability 0.6 ± 0.005 and $-c$ with probability 0.4 ± 0.005. While H_A prefers $-c$, H_B is much more confident in its response, which favors $+c$. A simple unweighted average of their responses $\frac{1}{2}(0.3 + 0.6)$ vs $\frac{1}{2}(0.7 + 0.4)$ would return $-c$. However, our ensemble classifier essentially computes scores of $0.3/0.1^2 + 0.6/0.005^2 = 24,030$ for $+c$ and $0.7/0.1^2 + 0.4/0.005^2 = 16,070$ for $-c$, and so returns $+c$. Note that the variances (and hence the weightings of the classifiers) are *query-specific*: for a different query, H_A's variance might be smaller that H_B's; this would mean that H_A would have the dominant influence in the decision.

Section 2 provides a quick literature review, to place our results. Section 3 then provides the foundations:

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006.

specifying how to use variances to combine responses in general, then defining the two classes of belief nets we are considering as base classifiers (naïve Bayes and tree-augmented naïve Bayes), and showing how to approximate the variance of their responses. Section 4 then considers a number of ways to produce a variety of base classifiers to combine to form so-called MUVs[1], and shows empirically that they work effectively, across a number of real world examples. In particular, we show that the MUV formed from a set of base classifiers typically works (significantly) better than just using any of those base classifiers. We also show significant improvements by (1) combining using *James-Stein shrinkage*, not simple maximum likelihood; (2) using *bootstrap samples* to train different classifiers, not disjoint samples; and (3) using a *balanced sample*, not one that is class-wise disjoint. This section also shows that our "variance-based combination rule" improves on AdaBoost's simple combination rule, and that it works better than several other standard ways to combine probabilistic classifiers. Finally, we consider ways to combine a set of these MUV ensembles, and show this can produce further performance improvements. The webpage (Web, A) presents complete details of our experiments, as well as other studies, and also provides proofs of the claims presented here.

2. Related Work

Our MUV approach is an ensemble method (Bauer & Kohavi, 1999; Opitz & Maclin, 1999), as it first produces a set of classifiers, and then combines them to answer each query. There are many current ensemble approaches, including several (Perrone & Cooper, 1993; Krogh & Vedelsby, 1995; Sollich & Krogh, 1996; Taniguchi & Tresp, 1997) that use variance terms when combining the base classifiers. Our approach is significantly different. (1) Most identify a "variance" quantity with each *classifier* and so use the same weight for each test/training instance; by contrast, we (and some others) compute an "*classifier+instance variance*", which means our system can assign different weights to a classifier for different instances. (2) While many estimate a classifier-variance *empirically*, we can compute an instance-classifier-variance *analytically* (see Eqn 8), based on the sample used to train the classifiers. (3) While we use variance to weight each individual classifier-instance at *performance time*, some others (including (Krogh & Vedelsby, 1995; Sollich & Krogh, 1996)) use it during the training stage to help minimize the ensemble generalization error. (4) As our derivation is based on *MLE* (not least squares (Perrone & Cooper, 1993; Taniguchi & Tresp, 1997)), we were able to easily obtain a James-Stein estimator Eqn 4, which we found to be very effective.

There are yet other systems that compute and use *classifier-instance* weights at performance time. For example, the "mixture of experts" approach (Jacobs et al., 1991) weights each classifier-instance $\langle m, \mathbf{e}\rangle$ by the probability that classifier m is appropriate for instance $\mathbf{e}$, $P(\, m \,|\, \mathbf{e}\,)$, which is estimated from the partitioned datasample. Our model differs by using a term based on variance to weight each $\langle m, \mathbf{e}\rangle$ pair (Eqn 3) and by using an analytic form (here for variance), rather than an empirical estimate. Also, our MUV system produces a *classifier* by blurring together the different classifiers, while (Jacobs et al., 1991) selects just one of the *regressors* stochastically.

[1] "Mixture Using Variance" belief net classifiers

3. Foundations

3.1. Using Variance to Combine Responses

In general, a "probability-variance classifier" will return both a mean $\mu \in [0,1]$ and a variance σ^2 to each specific question — *e.g.*, its response to "What is $P(\,$Cancer=true $|$ Gender=male, Smoke=true, Age=old$\,)$?" might be "0.3 ± 0.1". (We will discuss the source of these variances in Section 3.2.1.) Now imagine we have a set of k such classifiers, each producing a $\langle \mu_i, \sigma_i^2 \rangle$ pair for a given query. What is the best single response to this query, based on this set of $\{\langle \mu_i, \sigma_i^2 \rangle\}_i$ values?

We formulate this as a statistical estimation problem. Assume the correct answer is λ, and view the result produced by classifier i as an estimate of λ with noise ϵ_i — *i.e.*, $\mu_i = \lambda + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ and the ϵ_is are independent of one another and the response λ. Then the maximum likelihood estimate of λ,

$$\hat{\lambda}_{\text{MLE}} \;=\; \frac{\sum_{i=1}^{k} \frac{\mu_i}{\sigma_i^2}}{\sum_{i=1}^{k} \frac{1}{\sigma_i^2}} \qquad (1)$$

is an unbiased estimator, whose variance is

$$\sigma^2\,[\,\lambda_{\text{MLE}}\,] \;=\; \sigma^2 \left[\, \frac{\sum_{i=1}^{k} \frac{\mu_i}{\sigma_i^2}}{\sum_{i=1}^{k} \frac{1}{\sigma_i^2}} \,\right] \;=\; \frac{1}{\sum_{i=1}^{k} \frac{1}{\sigma_i^2}} \qquad (2)$$

To translate to our context: given a fixed set of evidence $\mathbf{E} = \mathbf{e}$, let $\mu_i(c, \mathbf{e}) = E\,[\,P_i(\,c \,|\, \mathbf{e}\,)\,]$ be the expected response produced by the i^{th} probabilistic classifier for label c and $\sigma_i^2(c, \mathbf{e}) = \sigma^2\,[\,P_i(\,c \,|\, \mathbf{e}\,)\,]$ be the associated variance. The MUV-classifier would then com-

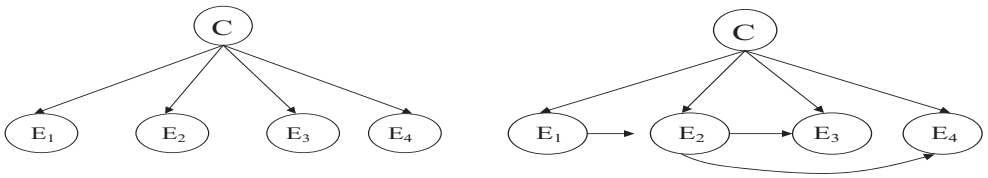


Figure 1. Structures of two simple Belief Nets: (a) Naïve Bayes (b) Tree-Augmented Naïve Bayes

pute

$$q(c, \mathbf{e}) \quad = \quad \frac{1}{\sum_i \frac{1}{\sigma_i^2(c,\mathbf{e})}} \sum_i \frac{\mu_i(c, \mathbf{e})}{\sigma_i^2(c, \mathbf{e})} \qquad (3)$$

for each $c \in C$, then return the largest $c^* = \text{argmax}_c\, q(c, \mathbf{e})$. Hence, we weight each probabilistic estimate with the reciprocal of the associated variance.[2]

We also consider the James-Stein estimator, which is biased but with smaller variance (Lehmann & Casella, 2003). Here we assume a prior distribution $\lambda \sim \mathcal{N}(0, \tau^2)$ with unknown parameter τ^2 and want to find the estimate $\lambda_{JS}(\bar{\mu}, \bar{\sigma})$ that minimizes square error loss, which is the posterior mean,

$$\hat{\lambda}_{Bayes}(\bar{\mu}, \bar{\sigma}) \;=\; E[\,\lambda \,|\, \bar{\mu}, \bar{\sigma}\,] \;=\; \frac{\sum_{i=1}^k \frac{\mu_i}{\sigma_i^2}}{\frac{1}{\tau^2} \;+\; \sum_{i=1}^k \frac{1}{\sigma_i^2}}$$

Since τ^2 is unknown, we estimate τ^2 using empirical Bayes approach. As shown in (Web, A), the resulting empirical Bayes estimator, called the "James-Stein estimator", is

$$\hat{\lambda}_{\text{JS}} \quad = \quad \left(1 - \frac{\sum_{i=1}^k \frac{1}{\sigma_i^2}}{(\sum_{i=1}^k \frac{\mu_i}{\sigma_i^2})^2}\right)^+ \hat{\lambda}_{\text{MLE}} \qquad (4)$$

where $(a)^+ = \max\{0, a\}$.[3]

3.2. Belief Networks, in General

In general, a (Bayesian) belief net is a model of a joint distribution, whose nodes represent random variables and whose directed arcs describe the dependencies among the nodes. For example, in Figure 1(a), E_1 depends directly on C, and is independent of E_2 given C; and in Figure 1(b), E_4 depends directly on C and E_2, etc.; see (Pearl, 1988). Each node D also includes a "CPtable" $\Theta = \{\theta_{d|\mathbf{f}}\}$ that quantifies these dependencies, with one "row" for each assignment $\mathbf{f}$ to D's parents, $\mathbf{F}_D$, which gives the conditional distribution for D in that context. For example, assuming C and each E_i is binary, then Figure 1(b)'s E_3's CPtable would have the following $2^2 = 4$ rows[4]

C	E_2	$E_3 = +$	$E_3 = -$
$+$	$+$	$\theta_{+e_3\|+c,+e_2}$	$\theta_{-e_3\|+c,+e_2}$
$+$	$-$	$\theta_{+e_3\|+c,-e_2}$	$\theta_{-e_3\|+c,-e_2}$
$-$	$+$	$\theta_{+e_3\|-c,+e_2}$	$\theta_{-e_3\|-c,+e_2}$
$-$	$-$	$\theta_{+e_3\|-c,-e_2}$	$\theta_{-e_3\|-c,-e_2}$

(5)

(There would be 2 rows for E_1, corresponding to $C = +$ and $C = -$, and only 1 row for C, which has no parents.)

We take a Bayesian stance, where we identify each CPtable row with a Dirichlet-distributed random variable; here perhaps $\theta_{E_1|+c} = \langle \theta_{+e_1|+c}, \theta_{-e_1|+c} \rangle$ is drawn from the $Dir(3, 7)$ Dirichlet distribution; *i.e.*, $\theta_{E_1|+c} \sim Dir(3, 7)$; see (Heckerman, 1998).

We often then use the expected value of each parameter; *i.e.*, $\bar{\theta}_{E_1|+c} = E\left[\theta_{E_1|+c}\right] = \langle \frac{3}{3+7}, \frac{7}{3+7} \rangle$. In general, if $\theta_{D|\mathbf{F}=\mathbf{f}} \sim Dir(\alpha_1, \ldots, \alpha_{|D|})$, then $\bar{\theta}_{D=i|\mathbf{F}=\mathbf{f}} = \frac{\alpha_i}{n_{D|\mathbf{F}=\mathbf{f}}}$ where

$$n_{D|\mathbf{F}=\mathbf{f}} \quad = \quad \sum_j \alpha_j \qquad (6)$$

is the "effective sample size" of this row. In fact, Cooper & Herskovitz (1992) prove that belief net computations using these posterior means always produce the mean value for any inference. That is, for any fixed query $P(\,c\,|\,\mathbf{e}\,)$ and belief net structure, let $P(\,c\,|\,\mathbf{e}, \Theta\,) = P_\Theta(c\,|\,\mathbf{e})$ be the response associated with the Θ parameters. Then

$$E[\,P(\,c\,|\,\mathbf{e}, \Theta\,)\,] \;=\; P(\,c\,|\,\mathbf{e}, E[\,\Theta\,]) \;=\; P_{\bar{\Theta}}(c\,|\,\mathbf{e}) \quad (7)$$

In general, we will be considering posterior Dirichlet distributions for each CPtable row, each based on

[2] (1) In general, to produce probability estimates we would then have to normalize these values: $P(\,c\,|\,\mathbf{e}\,) = q(c, \mathbf{e}) / \sum_{c'} q(c', \mathbf{e})$. Note however for binary classifiers, we will see that $\sigma_i^2(+, \mathbf{e}) = \sigma_i^2(-, \mathbf{e})$, which implies that $\sum_{c'} q(c', \mathbf{e}) = 1$.
(2) Recall this assumes the estimates are independent. There are obvious extensions to deal with correlated estimates; see (Taniguchi & Tresp, 1997).

[3] In our experiments, reported in Section 4, we never needed to use this "$\max\{0, \ldots\}$" condition.

[4] Here $+a$ is a shorthand for $A = +$ and $-a$ for $A = -$.

a prior $Dir(\alpha_1, \ldots, \alpha_{|D|})$ (which we will assume has $\alpha_i = 1$) and an observed sample. As we are dealing with complete samples S, this has an easy form (as the Dirichlet distribution is the conjugate prior for the multinomial distribution): $Dir(1 + \#_S(D = 1, \mathbf{F}_D = \mathbf{f}), \ \ldots, \ 1 + \#_S(D = |D|, \mathbf{F}_D = \mathbf{f}))$ where $\#_S(D = i, \mathbf{F}_D = \mathbf{f})$ is the number of instances in S that have both $D = i$ and $\mathbf{F}_D = \mathbf{f}$. For example, if there are 100 instances in S that have $E_1 = +$ and $C = +$, and 200 instances that have $E_1 = -$ and $C = +$, then $\theta_{E_1|+c} \sim Dir(1 + 100, \ 1 + 200)$.

As we are using the belief net B for a classification task, the associated Bayesian classifier H_B will return the class label $H_B(\mathbf{e}) = \text{argmax}_c\{P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,)\}$. We will measure error as the simple empirical 0/1 loss over an independent (*e.g.*, hold-out) dataset $S = \{\langle c_i, \mathbf{e}_i\rangle\}_i$

$$\widehat{\text{err}}^{(S)}(\, B\,) \quad = \quad \frac{|\{H_B(\mathbf{e}_i) \neq c_i \ | \ \langle \mathbf{e}_i, c_i\rangle \in S\}|}{|S|}$$

3.2.1. Computing the Variance of a Belief Net Response, in General

For a given structure and query, the response $q(\Theta) = P(\, c \,|\, \mathbf{e}, \Theta\,)$ is a function of the parameters Θ. As these parameters are random variables, so is this response $q(\Theta)$. Eqn 7 provides its expected value. Assuming the structure is correct, its variance is asymptotically (Van Allen et al., 2001):

$$\begin{aligned}
&\widehat{\sigma}^2\,[\,P(\, c \,|\, \mathbf{e}\,)\,] \quad = \quad \sum_{\theta_{D|\mathbf{f}} \in \Theta} \frac{1}{1 + n_{D|\mathbf{f}}} \bar{v}_{c|\mathbf{e}}(D|\mathbf{f}) \\
&\bar{v}_{c|\mathbf{e}}(D|\mathbf{f}) \quad = \\
&\sum_{d \in D} \frac{1}{\bar{\theta}_{d|\mathbf{f}}} \left[\, P_{\bar{\Theta}}(\, d, \mathbf{f}, c \,|\, \mathbf{e}\,) - P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,)\ P_{\bar{\Theta}}(\, d, \mathbf{f} \,|\, \mathbf{e}\,)\ \right]^2 \\
&- \left[\, P_{\bar{\Theta}}(\, \mathbf{f}, c \,|\, \mathbf{e}\,) - P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,) P_{\bar{\Theta}}(\, \mathbf{f} \,|\, \mathbf{e}\,)\ \right]^2 \qquad (8)
\end{aligned}$$

where $P_{\bar{\Theta}}(\cdot\,|\,\cdot)$ refers to the response computed using the posterior mean $\bar{\Theta}$. Note there is one $\bar{v}_{c|\mathbf{e}}(D|\mathbf{f})$ term for each CPtable row $\theta_{D|\mathbf{F}=\mathbf{f}}$. We also use its associated "effective sample size" $n_{D|\mathbf{F}=\mathbf{f}}$ (Eqn 6); given our uniform prior and complete data assumptions, $n_{D|\mathbf{F}=\mathbf{f}} = |D| + \#_S(\mathbf{F}_D = \mathbf{f})$. Note this $\widehat{\sigma}^2\,[\,P(\, c \,|\, \mathbf{e}\,)\,]$ quantity depends only on the expected values of the parameters and the effective sample size of each row.

To compute each $\bar{v}_{c|\mathbf{e}}(D|\mathbf{f})$ component, we sum over the values of the variable $d \in D$. To help explain the various terms here, consider Figure 1(b) where all variables are binary, and suppose the query is $P(+c|\ +e_1, -e_2, -e_3, +e_4)$. We would then sum over the 15 CPtable rows $\{\langle D, \mathbf{F}_D = \mathbf{f}\rangle\}$, each $\bar{v}_{+c|+e_1,-e_2,-e_3,+e_4}(D|\mathbf{F}_D = \mathbf{f})$ component corresponding to some node $D \in \{C, E_1, \ldots, E_4\}$, conditioned on some assignment $\mathbf{F}_D = \mathbf{f}$ to its parents $\mathbf{F}_D$. One such row, from Eqn 5, is $\theta_{E_3|+c,+e_2}$ (here "$\mathbf{F}_{E_3} = \mathbf{f}$" is "$+c, -e_2$"), requiring us to compute a value for $\bar{v}_{+c+|+e_1,-e_2,-e_3,+e_4}(E_3| + c, -e_2)$.

While the proof underlying Eqn 8 is based an asymptotically-Gaussian assumption, we are not suggesting using that the actual form of the posterior distribution is Gaussian, especially as this response is in $[0, 1]$, while a Gaussian has infinite support. (Web, B) explores the idea of instead fitting this mean and approximate variance to a Beta distribution, and shows this posterior distribution is fairly accurate. Also, while Eqn 8 is only guaranteed to be (asymptotically) correct when the structure is correct, we will nevertheless apply it below in situations where we know the structure is wrong, and observe that it still works effectively.

3.3. Naïve Bayes (NB)

The "naïve Bayes" structure assumes the attributes (a.k.a. evidence nodes) are independent, given the class (Duda et al., 2002); see Figure 1(a). Here,

$$E\,[\,P_{\Theta}(\, c \,|\, \mathbf{e}\,)\,] = P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,) \ =$$

$$\frac{P_{\bar{\Theta}}(\, c\,)\ \prod_i P_{\bar{\Theta}}(\, e_i \,|\, c\,)}{P_{\bar{\Theta}}(\, \mathbf{e}\,)} \quad = \quad \frac{\bar{\theta}_c\ \prod_i \bar{\theta}_{e_i|c}}{\sum_{c'} \bar{\theta}_{c'}\ \prod_i \bar{\theta}_{e_i|c'}}$$

To compute the variance of a naïve Bayes response, we need to compute $\bar{v}_{c|\mathbf{e}}(D|\mathbf{f})$ for each CPtable row. We will use the quantity $R(c; \mathbf{e}) = \left[P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,) \ - \ P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,)^2\right]^2$. (Given that we have already computed the expected response $P_{\bar{\Theta}}(\, c \,|\, \mathbf{e}\,)$, this $R(c; \mathbf{e})$ quantity is trivial to compute.) In a slight change of notation, we will now use $+c$ to refer to the value mentioned in query, and $-c$ to refer to any other possible value of C. Similarly, $+e$ will refer to the value of E appearing in the evidence part of the query — *i.e.*, "$E_2 = +e_2$" $\in \mathbf{E}{=}\mathbf{e}$.

$$\bar{v}_{+c|\mathbf{e}}(C|\{\}) \quad = \quad \sum_{c'} \frac{R(c'; \mathbf{e})}{\bar{\theta}_{c'|\{\}}} \ = \ \sum_{c'} \frac{R(c'; \mathbf{e})}{\bar{\theta}_{c'}} \qquad (9)$$

$$\bar{v}_{+c|\mathbf{e}}(E| + c) \quad = \quad R(+c; \mathbf{e}) \left[\frac{1}{\bar{\theta}_{+e|+c}} - 1\right] \qquad (10)$$

$$\begin{aligned}
\bar{v}_{+c|\mathbf{e}}(E| - c) \quad &= \ [P_{\bar{\Theta}}(\, +c \,|\, \mathbf{e}\,)\, P_{\bar{\Theta}}(\, -c \,|\, \mathbf{e}\,)]^2 \ \times \\
&\left[\frac{1}{\bar{\theta}_{+e|+c}} - 1\right] \qquad (11)
\end{aligned}$$

(When C is binary, then $P_{\bar{\Theta}}(\, -c \,|\, \mathbf{e}\,) = 1 - P_{\bar{\Theta}}(\, +c \,|\, \mathbf{e}\,)$, which means Eqn 10 and Eqn 11 become the same.)

3.4. Tree-Augmented Naïve Bayes (TAN)

A naïve Bayes structure is unable to express any dependencies between the attributes. To begin to address this limitation, a "tree-augmented naïve bayes" structure, TAN, augments the NB structure by allowing some edges among attributes. In particular, a TAN structure includes a link from the classification node down to each attribute and, if we ignore those class-to-attribute links, the remaining links, connecting attributes to each other, form a tree; see Figure 1(b). (Hence this representation allows each attribute to have at most one "attribute parent".) Friedman *et al.* (1997) provide an efficient "straight-line" algorithm for learning such TAN structures given complete data, and prove that the resulting structure maximizes the likelihood of the data, over all possible TAN structures.

To compute the variance of a TAN response, we have to consider CPtable rows from three types of nodes: the root node C, the evidence node with no evidence parents (*e.g.*, E_1 in Figure 1(b)), and all other evidence nodes with one evidence parent (*e.g.*, E_2, E_3 and E_4 in Figure 1(b)). We can use Eqn 9 for C and Eqns 10 and 11 for E_1. We therefore need only deal with the evidence nodes with one evidence parent. To simplify the notation, call this node E, with its evidence parent F and class parent C. We need to consider 4 classes of CPtable rows, depending on whether the value of F matches the evidence part of the query or not (written "$+f$" vs "$-f$") and whether the value of C matches the query c or not ("$+c$" vs "$-c$").

$$\bar{v}_{+c|\mathbf{e}}(E|+f,+c) = R(c;\mathbf{e})\left[\frac{1}{\bar{\theta}_{e|+f,+c}}-1\right] \quad (12)$$

$$\bar{v}_{+c|\mathbf{e}}(E|+f,-c) = [P_{\bar{\Theta}}(+c\,|\,\mathbf{e})\,P_{\bar{\Theta}}(-c\,|\,\mathbf{e})]^2 \times \left[\frac{1}{\bar{\theta}_{e|+f,-c}}-1\right] \quad (13)$$

$$\bar{v}_{+c|\mathbf{e}}(E|-f,+c) = \bar{v}_{+c|\mathbf{e}}(E|-f,-c) = 0$$

(Eqn 12 and 13 are isomorphic to Eqn 10 and 11.)

4. Experiments

This section empirically investigates how to compute the best MUV, which depends on $\langle 1a\rangle$ the type of base classifiers, $\langle 1b\rangle$ the number of such classifiers, $\langle 2\rangle$ how we generate the training instance for each, and $\langle 3\rangle$ exactly how we combine their results.

Let MUV(kNB, $\langle$ boot/disj, bal/skew $\rangle$, mle/js) refer to the MUV built from k naïve Bayes classifiers, combined using maximum likelihood (mle) vs a James-Stein estimator (js), from a set of datasamples produced by standard bootstrap methods vs divided into k disjoint partitions, where the division was balanced wrt the class, vs skewed. (Note that these arguments do not apply to MUV(1NB, $\langle\cdot,\cdot\rangle$, $\cdot$).) We similarly define the $2^3 = 8$ classes for each k: MUV(kTAN, $\langle$ boot/disj, bal/skew $\rangle$, mle/js). In general, the term MUV($\langle 1\rangle, \langle 2\rangle, \langle 3\rangle$) refers to the mixture-using-variance model where $\langle 1\rangle$ refers to the type of base classifier, and perhaps the number of such classifiers (*e.g.*, 2-NB or 3-TAN); $\langle 2\rangle$ is how to generate the different samples used to produce the different classifiers, and $\langle 3\rangle$ refers to the estimator used (either mle or js).

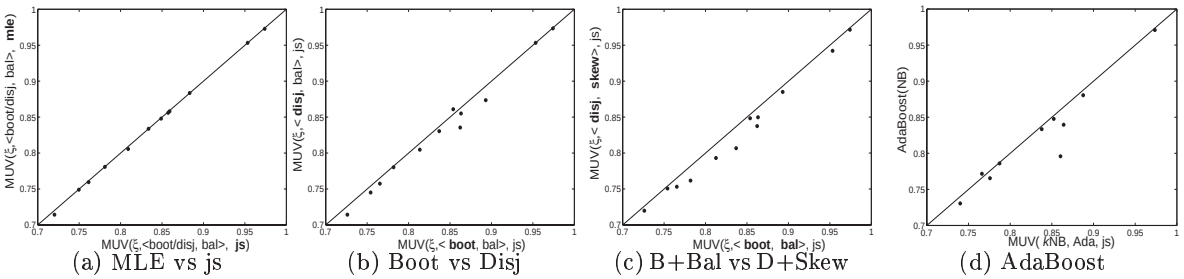
We used the following 12 UCI datasets (UCI, 2006) for our experiments, listing the name followed by its number of classes, number of instances and number of attributes: AUSTRALIAN(2, 690, 14), BREAST(2, 286, 9), CLEVE(2, 303, 13), CRX(2, 690, 15), DIABETES(2, 768, 8), GERMAN(2, 1000, 20), GLASS2(2, 163, 9), HEART(2, 270, 13), IRIS(3, 150, 4), PIMA(2 768, 8), VOTE(2, 435, 15),[5] and WAVEFORM(3, 300, 21); see (Web, A) for details.

4.1. How to combine Base Classifiers $\langle 3\rangle$: MLE vs James-Stein Estimator

For each trial (each dataset, and each cross-validation iteration), we learned a single naïve Bayes (1NB) from the training data S. We then produced k bootstrap datasets $\{S_i\}$ from S, and trained a NB from each. Here, we considered two combination rules: MLE (Eqn 1, producing Eqn 3) vs JS (Eqn 4). Hence we are comparing MUV(kNB, $\langle$boot,bal$\rangle$, **mle**) with MUV(kNB, $\langle$boot,bal$\rangle$, **js**). We similarly produced and compared MUV(kTAN, $\langle$boot,bal$\rangle$, **mle**) with MUV(kTAN, $\langle$boot,bal$\rangle$, **js**). We then evaluated each these systems on the remaining fold.

Over these tests, we found that the "js" variants were typically better than the corresponding "mle" ones. For example, MUV(3NB, $\langle$boot,bal$\rangle$, **js**) dominates MUV(3NB, $\langle$boot,bal$\rangle$, **mle**) in 5 out of 12 sets and is never worse. A 1-sided t-test, over all 360 = 12[datasets] × 3 [values of k] × 2 [structures] × 5 [folds] situations shows that the James-Stein estimator is better than MLE with $p < 2.3\text{E-}5$. We found similar results in the "disjoint" case (next section) where MUV(kNB, $\langle$disj,bal$\rangle$, **js**) dominated MUV(kNB, $\langle$disj,bal$\rangle$, **mle**); and MUV(kTAN, $\langle$disj,bal$\rangle$, **js**) dominated MUV(kTAN, $\langle$disj,bal$\rangle$, **mle**) with $p < 0.0099$.

[5] As suggested in (Holte, 1993), we remove the "Physician-free-freeze" variable from the VOTE data, as it alone leads to 99% accuracy.



(a) MLE vs js (b) Boot vs Disj (c) B+Bal vs D+Skew (d) AdaBoost

Figure 2. (a) MUV($\xi, \langle(\text{boot/disj}), \text{bal}\rangle, \underline{\text{js}}$) vs MUV($\xi, \langle(\text{boot/disj}), \text{bal}\rangle, \underline{\text{mle}}$); (b) MUV($\xi, \langle\underline{\text{boot}}, \text{bal}\rangle, \text{js}$) vs MUV($\xi, \langle\underline{\text{disj}}, \text{bal}\rangle, \text{js}$); (c) MUV($\xi, \langle\underline{\text{boot}}, \underline{\text{bal}}\rangle, \text{js}$) vs MUV($\xi, \langle\underline{\text{disj}}, \underline{\text{skew}}\rangle, \text{js}$); (d) MUV(kNB, Ada, js) vs AdaBoost(NB). Each $\langle X, Y\rangle$ point corresponds to one dataset. In (a,b,c), averaged over all $\xi \in \{2, 3, 4\} \times \{\text{NB, TAN}\}$.

Figure 2(a) plots the average for each database, over both NB and TAN cases.

4.2. How to generate Samples: $\langle 2a\rangle$ Bootstrap vs Disjoint

For each trial, we need to generate different training sets for the k different classifiers. One approach is to divide the data into k disjoint subsets (each of size $\approx |S|/k$); another involves generating k bootstrap samples — each formed by drawing $|S|$ instances with replacement. We found that bootstrap worked significantly better with $p < 1.41\text{E-}10$. Figure 2(b) shows that the MUV-ensemble (over both kNBs and kTANs, for $k = 2, 3, 4$) based on bootstrapping MUV(kNB, $\langle$**boot**,bal$\rangle$, js) outperforms the one based on disjoint subsets MUV(kNB, $\langle$**disj**,bal$\rangle$, js).

4.3. How to generate Samples: $\langle 2b\rangle$ Balanced vs Skewed

So far, we always used "balanced" subsets of the data — where each split (whether bootstrap or disjoint) has roughly the population average over the class distribution. (So if 60% of the instances in S are +, then each subset will have about 60% positive instances.) In our case, it might make sense to instead consider learning one NB based on the positive instances, and another only on the negatives. Here, we assume the +NB will know more about +instances, and so have more confidence in its vote than will −NB. (This is motivated by the way multi-TAN systems (Friedman et al., 1997) are constructed.) We therefore compare MUV($\xi, \langle\text{boot}, \textbf{bal}\rangle, \text{js}$), systems with MUV($\xi, \langle\text{disj}, \textbf{skew}\rangle, \text{js}$) where the latter splits the data based on the class. (Note that the k here corresponds to the number of classes.)

However, our empirical results show that MUV($\xi, \langle\text{disj}, \textbf{skew}\rangle, \text{js}$) is a bad idea, as it almost always performs worse than MUV($\xi, \langle\text{boot}, \textbf{bal}\rangle, \text{js}$), often with significant margins; see Figure 2(c). Balanced is better than Skewed with $p < 3.59\text{E-}09$.

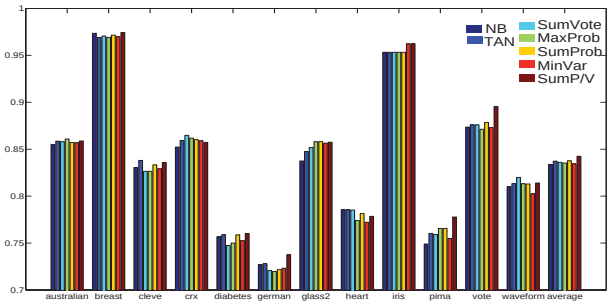
This empirical evidence supports the claim that one should use bootstrapping with balanced split, combined using James-Stein shrinkage — MUV($-, \langle\text{boot}, \text{bal}\rangle, \text{js}$) — over the other options.

4.4. How to generate Samples: $\langle 2\rangle$ AdaBoost

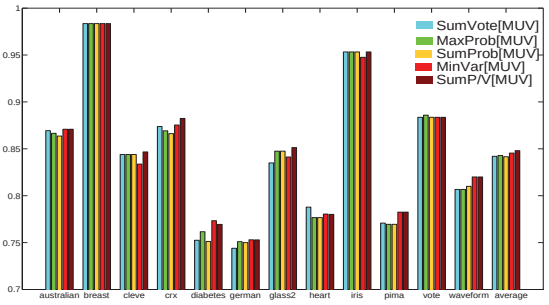
There are, of course, other very effective ways to generate different samples for different base classifiers, for use in an ensemble. One well-known approach is Boosting (Freund & Schapire, 1997), which sequentially re-weights the instances, to produce a series of different training sets. Many projects have shown that that boosting in general, and AdaBoost in particular, works very effectively.

After AdaBoost has produced its set of classifiers, it then combines them in an instance-independent fashion — *i.e.*, it learns a single scalar weight for each classifier, which it uses when combining their responses, for any instance. We therefore investigated whether we could improve this performance by using our mixture-using-variance approach to combine their responses.

In particular, we used NB as the AdaBoost base classifiers. This produced a set of base NB classifiers. We then formed the AdaBoost(NB) classifier by combining these base classifiers in the usual AdaBoost manner, and also formed the MUV(kNB, Ada, js) ensemble by combining them using the MUV approach, here using James-Stein estimator. The results appear in Figure 2(d). We see that MUV(kNB, Ada, js) outperforms AdaBoost(NB) at the $p < 0.04$ level, even though we are using the base classifiers that AdaBoost generated! We also note that MUV's performance in non-binary data sets (*e.g.*, waveform) does not degrade as much as AdaBoost(NB)'s.



(a) Different combinations of multiple base classifiers



(b) Different combinations of multiple MUV classifiers.

Figure 3. Average accuracies (over 12 datasets) of different combinations of different classifiers

4.5. Other Combination Rules

The above empirical evidence shows that our MUV approach works effectively. Of course, this might simply be because *any* combination would work well. We therefore compared our approach with other combinators.

For each query, each binary base classifier p_i produces 3 quantities: $\mu_i(+)$, $\mu_i(-)$, and σ_i^2. (Recall from Footnote 2 that $\sigma_i^2(+) = \sigma_i^2(-)$ for binary classifiers.) For each i, let $V_i(+) = 1$ and $V_i(-) = 0$ if $\mu_i(+) > \mu_i(-)$, and otherwise let $V_i(+) = 0$ and $V_i(-) = 1$. Below are five ways to combine k such classifiers, for a given query $\mathbf{e}$.

1. **SumVote**: return $\text{argmax}_c\{\sum_i V_i(c)\}$
2. **MaxProb**: return $\text{argmax}_c\{\max_i \mu_i(c)\}$
3. **SumProb**: return $\text{argmax}_c\{\sum_i \mu_i(c)\}$
4. **MinVar**: return $\text{argmin}_c\{\min_i \sigma_i^2(c)\}$
5. **SumP/V**: return $\text{argmax}_c\{\sum_i \mu_i(c)/\sigma_i^2(c)\}$

Note only the final two involve variance; **SumP/V** corresponds[6] to our MUV; and given our bootstrap sampling framework, **SumProb** corresponds to Bagging (Breiman, 1996). Figure 3(a) summarizes the average performance of these 5 combinators and shows that **SumP/V** (aka MUV) outperforms the other combination rules, with the following p values:

1NB	1TAN	SumVote	MaxPr	SumPr	MinV
0.003	0.016	0.015	0.003	0.011	0.001

4.6. Which Base Classifiers ⟨1⟩: MUV[MUV]?

The previous sections investigated MUVs built from between 1 and 4 base classifiers, each either a NB or a TAN structure. Unfortunately, it is not clear which of these work best, as different MUVs work best for different databases.

Why not learn *many* MUVs, and then somehow combine their responses for each query? This "mixture of MUV-mixtures" is practical, as the algorithms to learn each of these MUVs is straight-line code and so is very efficient, as are the algorithms to compute, and then combine, their responses.

Unfortunately, it is not clear how to combine these ensembles. We therefore considered the same combination rules presented in Section 4.5; but rather than use base classifiers, we instead combined nine MUVs: 1NB, 2NB, 3NB, 4NB, 1TAN, 2TAN, 3TAN, 4TAN, and AdaBoost(NB) (using "js" throughout, and "boot,bal" for the first 8).[7] Figure 3(b) presents the results, over those 5 options, for each dataset. We see that "SumP/V[Muv]" (that is, "MUV[MUV]") is the best; it is more accurate than the second best model selection approach, MinVar[Muv] (which also uses variance), with $p < 0.039$. Overall this result is consistent with the other outcomes (Figure 3(a)), as it shows that the variance-based combination rule continue to produce the best classifiers.

5. Conclusions

Future Work: (1) We are attempting to compute the variance of a MUV-classifier formed using the "js" combination Eqn 4. This is important for the MUV[MUV] model; see Footnote 7. (2) Is there a way to connect this analysis to the Bayesian approach, perhaps by interpreting the normalized version of the term $1/\sigma_m^2(c, e)$ in terms of the posterior probability of that classifier? (3) What is the asymptotic behaviour of this framework? Can we, for example, quantify its generalization performance as we increase the number of base classifiers? (It appears fairly robust to overfitting, at least over the number of classifiers we considered.) What should happen, as we increase the sample size, and hence the variance become smaller for each

[6] We actually modify the sum using James-Stein shrinkage, corresponding to Eqn 4. Note this is the only combinator where this operation is possible.

[7] Even though we use the "JS" variant for each MUV classifier, we use the variance for the associated MLE estimate Eqn 2.

classifier?

Contributions: This paper proposes a novel ensemble method MUV for constructing a mixture of Bayesian classifiers, which combines their responses using a query-specific measure that is based on the variances of these classifier responses. For each query, a MUV will first compute the mean and variance of the response of each base probabilistic classifier, then apply a statistically sound combination process (based on maximum likelihood estimation, possibly augmented with James-Stein shrinkage) to produce a single response. In essence, this allows each classifier to independently state its confidence in its assessment, which our combination process uses in making its assessment.

We applied this MUV technique to learn classifiers for a wide range of real-world datasets, using simple belief nets (NB and TAN structures) as base classifiers. We explored several ways of producing different classifiers and for combining their response; our results show that it is best to use *James-Stein shrinkage* and use *balanced bootstrap samples* to produce different classifiers. MUV's performance dominates two well-known ensemble methods: Boosting and Bagging. Moreover, our method is not as sensitive to non-binary classes as AdaBoost. Finally, we considered combining these MUVs, and found that using the mixture-using-variance approach on these combinations, MUV[MUV], produces yet better accuracy. In addition, we note that this system is extremely efficient, as both learning and testing involve only straight-line code.

For these reasons, we advocate using this MUV[MUV] approach for classification problems in general. See (Web, A) for additional details, and data.

Acknowledgements

We all gratefully acknowledge the advice from Ajit Singh and Peter Hooper, as well as the four anonymous reviewers, and financial support from iCORE, NSERC and the Alberta Ingenuity Centre for Machine Learning.

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning, 36.*

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24.*

Cooper, G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning, 9.*

Duda, R., Hart, P., & Stork, D. (2002). *Pattern classification, 2nd ed.* Wiley.

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55.*

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29.*

Hastie, T., Tibshirani, R., & Friedman, J. (2002). *The elements of statistical learning.* Springer.

Heckerman, D. (1998). A tutorial on learning with Bayesian networks. *Learning in Graphical Models.*

Holte, R. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 3.*

Inza, I., Larranaga, P., Lozano, J., & Pena, J. (2005). *Special issue of* Machine Learning Journal*: Probabilistic graphical models for classification*, vol. 59.

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation.*

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *NIPS* (pp. 231–238).

Lehmann, E., & Casella, G. (2003). *Theory of point estimation.* Springer.

Mitchell, T. (1997). *Machine learning.* McGraw-Hill.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research, 11.*

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann.

Perrone, M., & Cooper, L. (1993). When networks disagree: ensemble method for neural networks. *Artificial Neural Networks for Speech and Vision.*

Sollich, P., & Krogh, A. (1996). Learning with ensembles: How overfitting can be useful. *NIPS.*

Taniguchi, M., & Tresp, V. (1997). Averaging regularized estimators. *Neural Computation, 9.*

UCI (2006). http://www.ics.uci.edu/~mlearn/.

Van Allen, T., Greiner, R., & Hooper, P. (2001). Bayesian error-bars for belief net inference. *UAI.*

Web (A). http://www.cs.ualberta.ca/~greiner/MUV.

Web (B). http://www.cs.ualberta.ca/~greiner/BNvar.