Deep Broad Learning - Big Models for Big Data

Nayyar A. Zaidi  
Geoffrey I. Webb  
Mark J. Carman  
Francois Petitjean  
Faculty of Information Technology  
Monash University  
VIC 3800, Australia.

Editor: xxxxxxx xxxxxxx

Abstract

Deep learning has demonstrated the power of detailed modeling of complex high-order (multivariate) interactions in data. For some learning tasks there is power in learning models that are not only Deep but also Broad. By Broad, we mean models that incorporate evidence from large numbers of features. This is of especial value in applications where many different features and combinations of features all carry small amounts of information about the class. The most accurate models will integrate all that information. In this paper, we propose an algorithm for Deep Broad Learning called DBL. The proposed algorithm has a tunable parameter $n$, that specifies the depth of the model. It provides straightforward paths towards out-of-core learning for large data. We demonstrate that DBL learns models from large quantities of data with accuracy that is highly competitive with the state-of-the-art.

Keywords: Classification, Big Data, Deep Learning, Broad Learning, Discriminative-Generative Learning, Logistic Regression, Extended Logistic Regression

1. Introduction

The rapid growth in data quantity (Ganz and Reinsel, 2012) makes it increasingly difficult for machine learning to extract maximum value from current data stores. Most state-of-the-art learning algorithms were developed in the context of small datasets. However, the amount of information present in big data is typically much greater than that present in small quantities of data. As a result, big data can support the creation of very detailed models that encode complex higher-order multivariate distributions, whereas, for small data, very detailed models will tend to overfit and should be avoided (Brain and Webb, 2002; Martinez et al., 2015). We highlight this phenomenon in Figure 1. We know that the error of most classifiers decreases as they are provided with more data. This can be observed in Figure 1 where the variation in error-rate of two classifiers is plotted with increasing quantities of training data on the poker-hand dataset (Frank and Asuncion, 2010). One is a low-bias high-variance learner (KDB $k = 5$, taking into account quintic features, (Sahami, 1996)) and the other is a low-variance high-bias learner (naive Bayes, a linear classifier). For small quantities of data, the low-variance learner achieves the lowest error. However,
as the data quantity increases, the low-bias learner comes to achieve the lower error as it can better model higher-order distributions from which the data might be sampled.

The capacity to model different types of interactions among variables in the data is a major determinant of a learner’s bias. The greater the capacity of a learner to model differing distributions, the lower its bias will tend to be. However, many learners have limited capacity to model complex higher-order interactions.

Deep learning has demonstrated some remarkable successes through its capacity to create detailed (deep) models of complex multivariate interactions in structured data (e.g., data in computer vision, speech recognition, bioinformatics, etc.). Deep learning can be characterized in several different ways. But the underlying theme is that of learning higher-order interactions among features using a cascade of many layers. This process is known as ‘feature extraction’ and can be un-supervised as it leverages the structure within the data to create new features. Higher-order features are created from lower-order features creating a hierarchical structure. We conjecture that the deeper the model the higher the order of interactions that are captured in the data and the lower the bias that the model exhibits.

We argue that in many domains there is value in creating models that are broad as well as deep. For example, when using web browsing history, or social network likes, or when analyzing text, it is often the case that each feature provides only extremely small amounts of information about the target class. It is only by combining very large amounts of this micro-evidence that reliable classification is possible.

We call a model broad if it utilizes large numbers of variables. We call a model deep and broad if it captures many complex interactions each between numerous variables. For example, typical linear classifiers such as Logistic Regression (LR) and Naive Bayes (NB) are Broad Learners, in that they utilize all variables. However, these models are not deep, as

---

they do not directly model interactions between variables. In contrast, Logistic Regression\(^2\) with cubic features (LR\(^3\)) \cite{Langford2007} and Averaged 2-Dependence Estimators (A2DE) \cite{Webb2011}, \cite{Zaidi2012}, both of which consider all combinations of 3 variables, are both Deep and Broad. The parameters of the former are fit discriminatively through computationally intensive gradient descent-based search, while the parameters of the latter are fit generatively using computationally efficient maximum-likelihood estimation. This efficient estimation of A2DE parameters makes it computationally well-suited for big data. In contrast, we argue that LR\(^3\)’s discriminative parameterization can more closely fit the data than A2DE, making it lower bias and hence likely to have lower error when trained on large training sets. However, the computation required to optimize the parameters for LR\(^3\) becomes computationally intensive even on moderate dimensional data.

Recently, it has been shown that it is possible to form a hybrid generative-discriminate learner that exploits the strengths of both naive Bayes (NB) and Logistic Regression (LR) by creating a weighted variant of NB in which the weights are optimized using discriminative minimization of conditional log-likelihood \cite{Zaidi2013}. From one perspective, the resulting learner can be viewed as using weights to alleviate the attribute independence assumption of NB. From another perspective it can be seen to use the maximum likelihood parameterization of NB to pre-condition the discriminative search of LR. The result is a learner that learns models that are exactly equivalent to LR, but does so much more efficiently.

In this work, we show how to achieve the same result with LR\(^n\), creating a hybrid generative-discriminative learner named DBL\(^n\) for categorical data that learns equivalent deep broad models to those of LR\(^n\), but does so more efficiently. We further demonstrate that the resulting models have low bias and have very low error on large quantities of data. However, to create this hybrid learner we must first create an efficient generative counterpart to LR\(^n\).

In short, the contributions of this work are:

- developing an efficient generative counter-part to LR\(^n\), named Averaged n-Join Estimators (AnJE),
- developing DBL\(^n\), a hybrid of LR\(^n\) and AnJE,
- demonstrating that DBL\(^n\) has equivalent error to LR\(^n\), but is more efficient,
- demonstrating that DBL\(^n\) has low error on large data.

2. Notation

We seek to assign a value \(y \in \Omega_Y = \{y_1, \ldots, y_C\}\) of the class variable \(Y\), to a given example \(x = (x_1, \ldots, x_a)\), where the \(x_i\) are value assignments for the \(a\) attributes \(A = \{X_1, \ldots, X_a\}\). We define \(\binom{A}{n}\) as the set of all subsets of \(A\) of size \(n\), where each subset in the set is denoted \(\binom{A}{n}\) as the set of all subsets of \(A\) of size \(n\), where each subset in the set is denoted.

\(^2\) Logistic Regression taking into account all \(n\)-level features is denoted by LR\(^n\), e.g., LR\(^2\), LR\(^3\), LR\(^4\), etc. takes into account all quadratic, cubic, quartic, etc. features.
as \( \alpha \):
\[
\binom{A}{n} = \{ \alpha \subseteq A : |\alpha| = n \}.
\]

We use \( x_\alpha \) to denote the set of values taken by attributes in the subset \( \alpha \) for any data object \( x \).

LR for categorical data learns a weight for every attribute value per class. Therefore, for LR, we denote, \( \beta_y \) to be the weight associated with class \( y \), and \( \beta_{y,i,x_i} \) to be the weight associated with attribute \( i \) taking value \( x_i \) with class label \( y \). For LR\( ^n \), \( \beta_{y,\alpha,x_\alpha} \) specifies the weight associated with class \( y \) and attribute subset \( \alpha \) taking value \( x_\alpha \). The equivalent weights for DBL\( ^n \) are denoted by \( w_y \), \( w_{y,i,x_i} \) and \( w_{y,\alpha,x_\alpha} \).

The probability of attribute \( i \) taking value \( x_i \) given class \( y \) is denoted by \( P(x_i | y) \). Similarly, probability of attribute subset \( \alpha \), taking value \( x_\alpha \) is denoted by \( P(x_\alpha | y) \).

3. Using generative models to precondition discriminative learning

There is a direct equivalence between a weighted NB and LR \( ^{Zaidi et al., 2013, 2014} \). We write LR for categorical features as:
\[
P_{LR}(y | x) = \exp \left( \beta_y + \sum_{i=1}^{a} \beta_{y,i,x_i} - \log \sum_{c \in \Omega_Y} \exp \left( \beta_c + \sum_{j=1}^{a} \beta_{c,j,x_j} \right) \right) \tag{1}
\]
and NB as:
\[
P_{NB}(y | x) = \frac{P(y) \prod_{i=1}^{a} P(x_i | y)}{ \sum_{c \in \Omega_Y} P(c) \prod_{i=1}^{a} P(x_i | c) }.
\]

One can add the weights in NB to alleviate the attribute independence assumption, resulting in the WANBIA-C formulation, that can be written as:
\[
P_{W}(y | x) = \frac{P(y)^{w_y} \prod_{i=1}^{a} P(x_i | y)^{w_{y,i,x_i}}}{ \sum_{c \in \Omega_Y} P(c)^{w_c} \prod_{j=1}^{a} P(x_j | c)^{w_{c,j,x_j}} }
= \exp \left( w_y \log P(y) + \sum_{i=1}^{a} w_{y,i,x_i} \log P(x_i | y) - \log \sum_{c \in \Omega_Y} \exp \left( w_c \log P(c) + \sum_{j=1}^{a} w_{c,j,x_j} \log P(x_j | c) \right) \right) \tag{2}
\]

When conditional log likelihood (CLL) is maximized for LR and weighted NB using Equation 1 and 2 respectively, we get an equivalence such that \( \beta_c \propto w_c \log P(c) \) and \( \beta_{c,i,x_i} \propto w_{c,i,x_i} \log P(x_i | c) \). Thus, WANBIA-C and LR generate equivalent models. While it might seem less efficient to use WANBIA-C which has twice the number of parameters of LR, the probability estimates are learned very efficiently using maximum likelihood estimation, and provide useful information about the classification task that in practice serve to effectively precondition the search for the parameterization of weights to maximize conditional log likelihood.
4. Deep Broad Learner (DBL)

In order to create an efficient and effective low-bias learner, we want to perform the same trick that is used by WANBIA-C for LR with higher-order categorical features. We define \( \text{LR}^n \) as:

\[
P_{\text{LR}^n}(y|x) = \frac{\exp(\beta_y + \sum_{\alpha \in {^nA}} \beta_{y,\alpha, x_\alpha})}{\sum_{c \in \Omega_y} \exp(\beta_c + \sum_{\alpha^* \in {^nA}} \beta_{c, \alpha^*, x_{\alpha^*}})}.
\] (3)

We do not include lower-order terms. For example, if \( n = 2 \) we do not include terms for \( \beta_{y,i,x_i,} \) as well as for \( \beta_{y,i,x_i,j,x_j} \), because doing so does not increase the space of distinct distributions that can be modeled but does increase the number of parameters that must be optimized.

To precondition this model using generative learning, we need a generative model of the form

\[
P(y|x) = P(y) \prod_{\alpha \in {^nA}} P(x_\alpha|y)
\]

\[
= \exp\left( \log P(y) + \sum_{\alpha \in {^nA}} \log P(x_\alpha|y) - \log \sum_{c \in \Omega_y} \exp\left( \log P(c) + \sum_{\alpha^* \in {^nA}} \log P(x_{\alpha^*}|c) \right) \right).
\] (5)

The only existing generative model of this form is a log-linear model, which requires computationally expensive conditional log-likelihood optimization and consequently would not be efficient to employ. It is not possible to create a Bayesian network of this form as it would require that \( P(x_i, x_j) \) be independent of \( P(x_i, x_k) \). However, we can use a variant of the AnDE [Webb et al., 2011, 2005] approach of averaging many Bayesian networks. Unlike AnDE, we cannot use the arithmetic mean, as we require a product of terms in Equation 4 rather than a sum, so we must instead use a geometric mean.

4.1 Averaged n-Join Estimators (AnJE)

Let \( \mathcal{P} \) be a partition of the attributes \( A \). By assuming independence only between the sets of attributes \( A \in \mathcal{P} \) one obtains an \( n \)-joint estimator:

\[
P_{\text{AnJE}}(x|y) = \prod_{\alpha \in \mathcal{P}} P(x_\alpha|y).
\]

For example, if there are four attributes \( X_1, X_2, X_3 \) and \( X_4 \) that are partitioned into the sets \( \{X_1, X_2\} \) and \( \{X_3, X_4\} \) then by assuming conditional independence between the sets we obtain

\[
P_{\text{AnJE}}(x_1, x_2, x_3, x_4|y) = P(x_1, x_2|y)P(x_3, x_4|y).
\]

Let \( \Psi^A_n \) be the set of all partitions of \( A \) such that \( \forall \mathcal{P} \in \Psi^A_n \forall \alpha \in \mathcal{P} |\alpha| = n \). For convenience we assume that \( |A| \) is a multiple of \( n \). Let \( \Upsilon^A_N \) be a subset of \( \Psi^A_n \) that includes each set of \( n \) attributes once:

\[
\Upsilon^A_N \subseteq \Psi^A_n : \forall \alpha \in {^nA} |\{\mathcal{P} \in \Upsilon^A_N : \alpha \in \mathcal{P}\}| = 1.
\]
The AnJE model is the geometric mean of the set of \( n \)-joint estimators for the partitions \( Q \in \mathcal{Y}_N \). The AnJE estimate of conditional likelihood on a per-datum-basis can be written as:

\[
P_{\text{AnJE}}(y | x) \propto P(y) P_{\text{AnJE}}(x | y)
\]

\[
\propto P(y) \prod_{\alpha \in \binom{A}{n}} P(x_{\alpha} | y)^{(n-1)!/(a-1)!}.
\]

This is derived as follows. Each \( P \) is of size \( s = a/n \). There are \( \binom{a}{n} \) attribute-value \( n \)-tuples. Each must occur in exactly one partition, so the number of partitions must be

\[
p = \left( \frac{a}{n} \right) / s = \frac{(a-1)!}{(n-1)!(a-n)!}.
\]

The geometric mean of all the AnJE models is thus

\[
P_{\text{AnJE}}(x | y) = \left[ \prod_{\alpha \in \binom{A}{n}} P(x_{\alpha} | y) \right]^{1/p}
\]

\[
= \prod_{\alpha \in \binom{A}{n}} P(x_{\alpha} | y)^{(n-1)!/(a-n)!/(a-1)!}.
\]

Using Equation (6) we can write the log of \( P(y | x) \) as:

\[
\log P_{\text{AnJE}}(y | x) \propto \log P(y) + \frac{(n-1)!/(a-n)!}{(a-1)!} \sum_{\alpha \in \binom{A}{n}} \log P(x_{\alpha} | y).
\]

4.2 DBL\(^n\)

It can be seen that AnJE is a simple model that places the weight defined in Equation (7) on all feature subsets in the ensemble. The main advantage of this weighting scheme is that it requires no optimization, making AnJE learning extremely efficient. All that is required for training is to calculate the counts from the data. However, the disadvantage AnJE is its inability to perform any form of discriminative learning. Our proposed algorithm, DBL\(^n\) uses AnJE to precondition LR\(^n\) by placing weights on all probabilities in Equation (4) and learning these weights by optimizing the conditional-likelihood\(^3\). One can re-write AnJE models with this parameterization as:

\[
P_{\text{DBL}}(y | x) = \exp \left( w_y \log P(y) + \sum_{\alpha \in \binom{A}{n}} w_{y,\alpha,x_{\alpha}} \log P(x_{\alpha} | y) - \log \sum_{c \in \Omega_y} \exp \left( w_c \log P(c) + \sum_{\alpha^* \in \binom{A}{n}} w_{c,\alpha^*,x_{\alpha^*}} \log P(x_{\alpha^*} | c) \right) \right).
\]

\(^3\) One can initialize these weights with weights in Equation (7) for faster convergence.
Note that we can compute the likelihood and class-prior probabilities using either MLE or MAP. Therefore, we can write Equation 10 as:
\[
\log P_{\text{DBL}}(y|x) = w_y \log \pi_y + \sum_{\alpha \in \mathcal{A}_n} w_{y,\alpha,x_\alpha} \log \theta_{x_\alpha|y} - \\
\log \sum_{c \in \Omega_Y} \exp \left( w_c \log \pi_c + \sum_{\alpha^* \in \mathcal{A}_n} w_{c,\alpha^*,x_{\alpha^*}} \log \theta_{x_{\alpha^*}|c} \right). 
\] (11)

Assuming a Dirichlet prior, a MAP estimate of \( P(y) \) is \( \pi_y \) which equals:
\[
\frac{#_y + m/|\mathcal{Y}|}{t + m},
\]
where \( #_y \) is the number of instances in the dataset with class \( y \) and \( t \) is the total number of instances, and \( m \) is the smoothing parameter. We will set \( m = 1 \) in this work. Similarly, a MAP estimate of \( P(x_\alpha|y) \) is \( \theta_{x_\alpha|y} \) which equals:
\[
\frac{#_{x_\alpha,y} + m/|x_\alpha|}{#_y + m},
\]
where \( #_{x_\alpha,y} \) is the number of instances in the dataset with class \( y \) and attribute values \( x_\alpha \).

DBL\( ^n \) computes weights by optimizing CLL. Therefore, one can compute the gradient of Equation 11 with respect to weights and rely on gradient descent based methods to find the optimal value of these weights. Since we do not want to be stuck in local minima, a natural question to ask is whether the resulting objective function is convex Boyd and Vandenberghe (2008). It turns out that the objective function of DBL\( ^n \) is indeed convex. Roos et al. (2005) proved that an objective function of the form \( \sum_{x \in \mathcal{D}} \log P_B(y|x) \), optimized by any conditional Bayesian network model is convex if and only if the structure \( G \) of the Bayesian network \( B \) is perfect, that is, all the nodes in \( G \) are moral nodes. DBL\( ^n \) is a geometric mean of several sub-models where each sub-model models \( \lfloor \frac{a}{n} \rfloor \) interactions each conditioned on the class attribute. Each sub-model has a structure that is perfect. Since, the product of two convex objective function leads to a convex function, one can see that DBL\( ^n \)'s optimization function will also lead to a convex objective function.

Let us first calculate the gradient of Equation 11 with respect to weights associated with \( \pi_y \). We can write:
\[
\frac{\partial \log P(y|x)}{\partial w_y} = 1_y \log \pi_y - \frac{\pi_y^{w_y} \log \pi_y \prod_{\alpha \in \mathcal{A}_n} \theta_{x_\alpha|y}^{w_{y,\alpha,x_\alpha}}}{\sum_{c \in \Omega_Y} \pi_c^{w_c} \prod_{\alpha^* \in \mathcal{A}_n} \theta_{x_{\alpha^*}|c}^{w_{c,\alpha^*,x_{\alpha^*}}}} = (1_y - P(y|x)) \log \pi_y, \tag{12}
\]
where \( 1_y \) denotes an indicator function that is 1 if derivative is taken with respect to class \( y \) and 0 otherwise. Computing the gradient with respect to weights associated with \( \theta_{x_\alpha|y} \) gives:
\[
\frac{\partial \log P(y|x)}{\partial w_{y,\alpha,x_\alpha}} = 1_y 1_\alpha \log \theta_{x_\alpha|y} - \frac{\pi_y^{w_y} \prod_{\alpha \in \mathcal{A}_n} \theta_{x_\alpha|y}^{w_{y,\alpha,x_\alpha}} 1_\alpha \log \theta_{x_\alpha|y}}{\sum_{c \in \Omega_Y} \pi_c^{w_c} \prod_{\alpha^* \in \mathcal{A}_n} \theta_{x_{\alpha^*}|c}^{w_{c,\alpha^*,x_{\alpha^*}}}} = (1_y - P(y|x)) 1_\alpha \log \theta_{x_\alpha|y}, \tag{13}
\]
where $1_\alpha$ and $1_y$ denotes an indicator function that is 1 if the derivative is taken with-
respect-to attribute set $\alpha$ (respectively, class $y$) and 0 otherwise.

### 4.3 Alternative Parameterization

Let us reparameterize DBL$^n$ such that:

$$
\beta_y = w_y \log \pi_y, \quad \text{and} \quad \beta_{y,\alpha,x} = w_{y,\alpha,x} \log \theta_{x\mid y}.
$$

(14)

Now, we can re-write Equation 11 as:

$$
\log P_{LR}(y\mid x) = \beta_y + \sum_{\alpha \in (A^n)} \beta_{y,\alpha,x} - \log \sum_{c \in \Omega_y} \exp \left( \beta_c + \sum_{\alpha^* \in (A^n)} \beta_{c,\alpha^*,x\mid y} \right).
$$

(15)

It can be seen that this leads to Equation 3. We call this parameterization LR$^n$.

Like DBL$^n$, LR$^n$ also leads to a convex optimization problem, and, therefore, its param-
eters can also be optimized by simple gradient decent based algorithms. Let us compute
the gradient of objective function in Equation 15 with-respect-to $\beta_y$. In this case, we can
write:

$$
\frac{\partial \log P(y\mid x)}{\partial \beta_y} = (1 - P(y\mid x)).
$$

(16)

Similarly, computing gradient with-respect-to $\beta_{\alpha\mid c}$, we can write:

$$
\frac{\partial \log P(y\mid x)}{\partial \beta_{y,\alpha,x}} = (1 - P(y\mid x))1_\alpha.
$$

(17)

### 4.4 Comparative analysis of DBL$^n$ and LR$^n$

It can be seen that the two models are actually equivalent and each is a re-parameterization
of the other. However, there are subtle distinctions between the two. The most important
distinction is the utilization of MAP or MLE probabilities in DBL$^n$. Therefore, DBL$^n$ is a
two step learning algorithm:

- Step 1 is the optimization of log-likelihood of the data ($\log P(y, x)$) to obtain the
  estimates of the prior and likelihood probabilities. One can view this step as of
generative learning.

- Step 2 is the introduction of weights on these probabilities and learning of these
  weights by maximizing CLL ($P(y\mid x)$) objective function. This step can be interpreted
  as discriminative learning.

DBL$^n$ employs generative-discriminative learning as opposed to only discriminative learning
by LR$^n$.

One can expect a similar bias-variance profile and a very similar classification perfor-
mance as both models will converge to a similar point in the optimization space, the only
difference in the final parameterization being due to recursive descent being terminated
before absolute optimization. However, the rate of convergence of the two models can be
very different. [Zaidi et al. (2014)] show that for NB, such DBL\textsuperscript{n} style parameterization with generative-discriminative learning can greatly speed-up convergence relative to only discriminative training. Note, discriminative training with NB as the graphical model is vanilla LR. We expect to see the same trend in the convergence performance of DBL\textsuperscript{n} and LR\textsuperscript{n}.

Another distinction between the two models becomes explicit if a regularization penalty is added to the objective function. One can see that in case of DBL\textsuperscript{n}, optimizing parameters towards 1 will effectively pull parameters back towards the generative training estimates. For smaller datasets, one can expect to obtain better performance by using a large regularization parameter and pulling estimates back towards 1. However, one cannot do this for LR\textsuperscript{n}. Therefore, DBL\textsuperscript{n} models can very elegantly combine generative discriminative parameters.

An analysis of the gradient of DBL\textsuperscript{n} in Equation 12 and 13 and that of LR\textsuperscript{n} in Equation 16 and 17 also reveals an interesting comparison. We can write DBL\textsuperscript{n}’s gradients in terms of LR\textsuperscript{n}’s gradient as follows:

\[
\frac{\partial \log P(y | x)}{\partial w_y} = \frac{\partial \log P(y | x)}{\partial \beta_y \log \pi_y},
\]

\[
\frac{\partial \log P(y | x)}{\partial w_{y,a,x_a}} = \frac{\partial \log P(y | x)}{\partial \beta_{y,a,x_a} \log \theta_{x_a | y}}.
\]

It can be seen that DBL\textsuperscript{n} has the effect of re-scaling LR\textsuperscript{n}’s gradient by the log of the conditional probabilities. We conjecture that such re-scaling has the effect of pre-conditioning the parameter space and, therefore, will lead to faster convergence.

5. Related Work

**Averaged n-Dependent Estimators (AnDE)** is the inspiration for AnJE. An AnDE model is the arithmetic mean of all Bayesian Network Classifiers in each of which all attributes depend on the class and the same \(n\) attributes. A simple depiction of A1DE in graphical form is shown in Figure 2. There are \(\binom{a}{n}\) possible combination of attributes that can be used as parents, producing \(\binom{a}{n}\) sub-models which are combined by averaging.

AnDE and AnJE both use simple generative learning, merely the counting the relevant **sufficient statistics** from the data. Second, both have only one tweaking parameter: \(n\) – that controls the bias-variance trade-off. Higher values of \(n\) leads to low bias and high variance and vice-versa.

It is important not to confuse the equivalence (in terms of the level of interactions they model) of AnJE and AnDE models. That is, the following holds:

\[
f(A2JE) = f(A1DE),
\]

\[
f(A3JE) = f(A2DE),
\]

\[
\vdots = \vdots
\]

\[
f(AnJE) = f(A(n-1)DE),
\]

where \(f(.)\) is a function that returns the number of interactions that the algorithm models. Thus, an AnJE model uses the same core statistics as an A(n-1)DE model. At training
time, AnJE and A(n-1)DE must learn the same information from the data. However, at classification time, each of these statistics is accessed once by AnJE and \( n \) times by A(n-1)DE, making AnJE more efficient. However, as we will show, it turns out that AnJE’s use of the geometric mean results in a more biased estimator than than the arithmetic mean used by AnDE. As a result, in practice, an AnJE model is less accurate than the equivalent AnDE model.

However, due to the use of arithmetic mean by AnDE, its weighted version would be much more difficult to optimize than AnJE, as transformed to log space it does not admit to a simple linear model.

A work relevant to DBL\(^n\) is that of Greiner et al. (2004); Greiner and Zhou (2002). The proposed technique in these papers named ELR has a number of similar traits with DBL\(^n\). For example, the parameters associated with a Bayesian network classifier (naive Bayes and TAN) are learned by optimizing the CLL. Both ELR and DBL\(^n\) can be viewed as feature engineering frameworks. An ELR (let us say with TAN structure) model is a subset of DBL\(^2\) models. The comparison of DBL\(^n\) with ELR is not the goal of this work. But in our preliminary results, DBL\(^n\) produce models of much lower bias that ELR (TAN). Modelling higher-order interactions is also an issue with ELR. One could learn a Bayesian network structure and create features based on that and then use ELR. But several restrictions needs to be imposed on the structure, that is, it has to fulfill the property of perfectness, to make sure that it leads to a convex optimization problem. With DBL\(^n\), as we discussed in Section 4.2, there are no restrictions. Need less to say, ELR is neither broad nor deep. Some related ideas to ELR are also explored in Pernkopf and Bilmes (2005); Pernkopf and Wohlmayr (2009); Su et al. (2008).

6. Experiments

In this section, we compare and analyze the performance of our proposed algorithms and related methods on 77 natural domains from the UCI repository of machine learning (Frank and Asuncion, 2010). The experiments are conducted on the datasets described in Table 1.
There are a total of 77 datasets, 40 datasets with less than 1000 instances, 21 datasets with instances between 1000 and 10000, and 16 datasets with more than 10000 instances. There are 8 datasets with over 100000 instances. These datasets are shown in bold font in Table 1.

Each algorithm is tested on each dataset using 5 rounds of 2-fold cross validation.

We compare four different metrics, i.e., 0-1 Loss, RMSE, Bias and Variance.

We report Win-Draw-Loss (W-D-L) results when comparing the 0-1 Loss, RMSE, bias and variance of two models. A two-tail binomial sign test is used to determine the significance of the results. Results are considered significant if \( p \leq 0.05 \).

The datasets in Table 1 are divided into two categories. We call the following datasets Big – KDDCup, Poker-hand, USCensus1990, Coverttype, MITFaceSetB, MITFaceSetA, Census-income, Localization. All remaining datasets are denoted as Little in the results. Due to their size, experiments for most of the Big datasets had to be performed in a heterogeneous environment (grid computing) for which CPU wall-clock times are not commensurable. In consequence, when comparing classification and training time, the following 9 datasets constitutes Big category – Localization, Connect-4, Shuttle, Adult, Letter-recog, Magic, Nursery, Sign, Pendigits.

Table 1: Details of Datasets

<table>
<thead>
<tr>
<th>Domain</th>
<th>Case Att Class</th>
<th>Domain</th>
<th>Case Att Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kddcup</td>
<td>5209000 41 40 Vowel</td>
<td>990 14 11</td>
<td></td>
</tr>
<tr>
<td>Poker-hand</td>
<td>1175067 10 10 Tic-Tac-ToeEndgame</td>
<td>958 10 2</td>
<td></td>
</tr>
<tr>
<td>MITFaceSetC</td>
<td>839000 361 2 Annealing</td>
<td>898 39 6</td>
<td></td>
</tr>
<tr>
<td>Coverttype</td>
<td>581012 55 7 Vehicle</td>
<td>846 19 4</td>
<td></td>
</tr>
<tr>
<td>MITFaceSetB</td>
<td>489400 361 2 PimaIndiansDiabetes</td>
<td>768 9 2</td>
<td></td>
</tr>
<tr>
<td>MITFaceSetA</td>
<td>474000 361 2 BreastCancer(Wisconsin)</td>
<td>699 10 2</td>
<td></td>
</tr>
<tr>
<td>Census-Income(KDD)</td>
<td>299285 40 2 CreditScreening</td>
<td>690 16 2</td>
<td></td>
</tr>
<tr>
<td>Localization</td>
<td>164860 7 3 BalanceScale</td>
<td>625 5 3</td>
<td></td>
</tr>
<tr>
<td>Connect-4Opening</td>
<td>67557 43 3 Syncon</td>
<td>600 61 6</td>
<td></td>
</tr>
<tr>
<td>Statlog(Shuttle)</td>
<td>58000 10 7 Chess</td>
<td>551 40 2</td>
<td></td>
</tr>
<tr>
<td>Adult</td>
<td>48842 15 2 Cylinder</td>
<td>540 40 2</td>
<td></td>
</tr>
<tr>
<td>LetterRecognition</td>
<td>20000 17 26 Musk1</td>
<td>476 167 2</td>
<td></td>
</tr>
<tr>
<td>MAGICGammaTelescope</td>
<td>19020 11 2 HouseVotes84</td>
<td>435 17 2</td>
<td></td>
</tr>
<tr>
<td>Nursery</td>
<td>12860 9 5 HorseColic</td>
<td>368 22 2</td>
<td></td>
</tr>
<tr>
<td>Sign</td>
<td>12546 9 3 Dermatology</td>
<td>366 35 6</td>
<td></td>
</tr>
<tr>
<td>PenDigits</td>
<td>10992 17 10 Ionosphere</td>
<td>351 35 2</td>
<td></td>
</tr>
<tr>
<td>Thyroid</td>
<td>9169 30 20 LiverDisorders(Bupa)</td>
<td>345 7 2</td>
<td></td>
</tr>
<tr>
<td>Pioneer</td>
<td>9150 37 57 PrimaryTumor</td>
<td>339 18 22</td>
<td></td>
</tr>
<tr>
<td>Mushrooms</td>
<td>8124 23 2 Haberman'sSurvival</td>
<td>306 4 2</td>
<td></td>
</tr>
<tr>
<td>Musk2</td>
<td>6598 167 2 HeartDisease(Cleveland)</td>
<td>303 14 2</td>
<td></td>
</tr>
<tr>
<td>Satellite</td>
<td>6435 37 6 Hungarian</td>
<td>294 14 2</td>
<td></td>
</tr>
<tr>
<td>OpticalDigits</td>
<td>5620 49 10 Audiology</td>
<td>226 70 24</td>
<td></td>
</tr>
<tr>
<td>PageBlocksClassification</td>
<td>5473 11 5 New-Thyroid</td>
<td>215 6 3</td>
<td></td>
</tr>
<tr>
<td>Wall-following</td>
<td>5456 25 4 GlassIdentification</td>
<td>214 10 3</td>
<td></td>
</tr>
<tr>
<td>Nettalk(Phoneme)</td>
<td>5438 8 52 SonarClassification</td>
<td>208 61 2</td>
<td></td>
</tr>
<tr>
<td>Waveform-5000</td>
<td>5600 41 3 AutoImports</td>
<td>205 26 7</td>
<td></td>
</tr>
<tr>
<td>Spambase</td>
<td>4601 58 2 WineRecognition</td>
<td>178 14 3</td>
<td></td>
</tr>
<tr>
<td>Abalone</td>
<td>4177 9 3 Hepatitis</td>
<td>155 20 2</td>
<td></td>
</tr>
<tr>
<td>Hypothyroid(Garavan)</td>
<td>3772 30 4 TeachingAssistantEvaluation</td>
<td>151 6 3</td>
<td></td>
</tr>
<tr>
<td>Sick-euthyroid</td>
<td>3772 30 2 IrisClassification</td>
<td>150 5 3</td>
<td></td>
</tr>
<tr>
<td>King-rook-vs-king-pawn</td>
<td>3196 37 2 Lymphography</td>
<td>148 19 4</td>
<td></td>
</tr>
<tr>
<td>Splice-junctionGeneSequences</td>
<td>3190 62 3 Echocardiogram</td>
<td>131 7 2</td>
<td></td>
</tr>
<tr>
<td>Segment</td>
<td>2310 29 7 PromoterGeneSequences</td>
<td>106 58 2</td>
<td></td>
</tr>
<tr>
<td>CarEvaluation</td>
<td>1728 8 4 Zoo</td>
<td>101 17 7</td>
<td></td>
</tr>
<tr>
<td>Volcanoes</td>
<td>1520 4 4 PostoperativePatient</td>
<td>90 9 3</td>
<td></td>
</tr>
<tr>
<td>Yeast</td>
<td>1484 9 10 LaborNegotiations</td>
<td>57 17 2</td>
<td></td>
</tr>
<tr>
<td>ContraceptiveMethodChoice</td>
<td>1473 10 3 LungCancer</td>
<td>32 57 3</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>1000 21 2 Contact-lenses</td>
<td>24 5 3</td>
<td></td>
</tr>
<tr>
<td>LED</td>
<td>1000 8 10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. Exception is MITFaceSetA, MITFaceSetB and Kddcup where results are reported with 2 rounds of 2-fold cross validation.

5. As discussed in Section 1, the reason for performing bias/variance estimation is that it provides insights into how the learning algorithm will perform with varying amount of data. We expect low variance algorithms to have relatively low error for small data and low bias algorithms to have relatively low error for large data [Brain and Webb 2002].
Table 2: Win-Draw-Loss: DBL\textsuperscript{2} vs. A2JE and DBL\textsuperscript{3} vs A3JE. \( p \) is two-tail binomial sign test. Results are significant if \( p \leq 0.05 \).

When comparing average results across \textit{Little} and \textit{Big} datasets, we normalize the results with respect to DBL\textsuperscript{2} and present a geometric mean.

Numeric attributes are discretized by using the Minimum Description Length (MDL) discretization method \cite{fayyad1992minimum}. A missing value is treated as a separate attribute value and taken into account exactly like other values.

We employed L-BFGS quasi-Newton methods \cite{zhu1997algorithm} for solving the optimization.

We used a Random Forest that is an ensemble of 100 decision trees \cite{breiman2001random}.

Both DBL\textsuperscript{n} and LR\textsuperscript{n} are L\textsuperscript{2} regularized. The regularization constant \( C \) is not tuned and is set to \( 10^{-2} \) for all experiments.

The detailed 0-1 Loss and RMSE results on \textit{Big} datasets are also given in Appendix A.

6.1 DBL\textsuperscript{n} vs. AnJE

A W-D-L comparison of the 0-1 Loss, RMSE, bias and variance of DBL\textsuperscript{n} and AnJE on \textit{Little} datasets is shown in Table 2. We compare DBL\textsuperscript{2} with A2JE and DBL\textsuperscript{3} with A3JE only. It can be seen that DBL\textsuperscript{n} has significantly lower bias but significantly higher variance. The 0-1 Loss and RMSE results are not in favour of any algorithm. However, on \textit{Big} datasets, DBL\textsuperscript{n} wins on 7 out of 8 datasets in terms of both RMSE and 0-1 Loss. The results are not significant since \( p \) value of 0.070 is greater than our set threshold of 0.05. One can infer that DBL\textsuperscript{n} successfully reduces the bias of AnJE, at the expense of increasing its variance.

Normalized 0-1 Loss and RMSE results for both models are shown in Figure 3. It can be seen that DBL\textsuperscript{n} has a lower averaged 0-1 Loss and RMSE than AnJE. This difference is substantial when comparing on \textit{Big} datasets. The training and classification time of AnJE is, however, substantially lower than DBL\textsuperscript{n} as can be seen from Figure 4. This is to be expected as DBL\textsuperscript{n} adds discriminative training to AnJE and uses twice the number of parameters at classification time.

\footnote{The original L-BFGS implementation of \cite{byrd1995limited} from \url{http://users.eecs.northwestern.edu/~nocedal/lbfgsb.html} is used.}
6.2 DBL$^n$ vs. AnDE

A W-D-L comparison for 0-1 Loss, RMSE, bias and variance results of the two DBL$^n$ models relative to the corresponding AnDE models are presented in Table 3. We compare DBL$^2$ with A1DE and DBL$^3$ with A2DE only. It can be seen that DBL$^n$ has significantly lower bias and significantly higher variance variance than AnDE models. Recently, AnDE models have been proposed as a fast and effective Bayesian classifiers when learning from large quantities of data (Zaidi and Webb, 2012). These bias-variance results make DBL$^n$ a suitable alternative to AnDE when dealing with big data. The 0-1 Loss and RMSE results (with exception of RMSE comparison of DBL$^3$ vs. A2DE) are similar.

<table>
<thead>
<tr>
<th></th>
<th>DBL$^2$ vs. A1DE</th>
<th></th>
<th>DBL$^3$ vs. A2DE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W-D-L</td>
<td>p</td>
<td>W-D-L</td>
</tr>
<tr>
<td>Bias</td>
<td>65/3/7</td>
<td>&lt;0.001</td>
<td>53/5/1Y</td>
</tr>
<tr>
<td>Variance</td>
<td>21/5/49</td>
<td>0.001</td>
<td>26/5/44</td>
</tr>
<tr>
<td>0-1 Loss</td>
<td>42/4/29</td>
<td>0.1539</td>
<td>39/3/33</td>
</tr>
<tr>
<td>RMSE</td>
<td>30/1/44</td>
<td>0.130</td>
<td>22/1/52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Big Datasets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-1 Loss</td>
<td>8/0/0</td>
<td>0.001</td>
<td>7/0/1</td>
</tr>
<tr>
<td>RMSE</td>
<td>7/0/1</td>
<td>0.073</td>
<td>6/0/2</td>
</tr>
</tbody>
</table>

Table 3: Win-Draw-Loss: DBL$^2$ vs. A1DE and DBL$^3$ vs A2DE. $p$ is two-tail binomial sign test. Results are significant if $p \leq 0.05$. 
Figure 5: Geometric mean of 0-1 Loss (Left) and RMSE (Right) performance of DBL$^2$, A1DE, DBL$^3$ and A2DE for Little and Big datasets.

Figure 6: Geometric mean of Training Time (Left), Classification Time (Right) of DBL$^2$, A1DE, DBL$^3$ and A2DE for All and Big datasets.

Normalized 0-1 Loss and RMSE are shown in Figure 5. It can be seen that the DBL$^n$ models have lower 0-1 Loss and RMSE than the corresponding AnDE models.

A comparison of the training time of DBL$^n$ and AnDE is given in Figure 6. As expected, due to its additional discriminative learning, DBL$^n$ requires substantially more training time than AnDE. However, AnDE does not share such a consistent advantage with respect to classification time, the relativities depending on the dimensionality of the data. For high-dimensional data the large number of permutations of attributes that AnDE must consider results in greater computation.

6.3 DBL$^n$ vs. LR$^n$

In this section, we will compare the two DBL$^n$ models with their equivalent LR$^n$ models. As discussed before, we expect to see similar bias-variance profile and a similar classification performance as the two models are re-parameterization of each other.

We compare the two parameterizations in terms of the scatter of their 0-1 Loss and RMSE values on Little datasets in Figure 7, 9 respectively, and on Big datasets in Figure 8, 10 respectively. It can be seen that the two parameterizations (with an exception of one dataset, that is: wall-following) have a similar spread of 0-1 Loss and RMSE values for both $n = 2$ and $n = 3$.

The comparative scatter of the number of iterations each parameterization takes to converge is shown in Figure 11 and 12 for Little and Big datasets respectively. It can be
Figure 7: Comparative scatter of 0-1 Loss of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Little datasets.

Figure 8: Comparative scatter of 0-1 Loss of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Big datasets.

Figure 9: Comparative scatter of RMSE of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Little datasets.
seen that the number of iterations for DBL$^n$ are far fewer than LR$^n$. With a similar spread of 0-1 Loss and RMSE values, it is very encouraging to see that DBL$^n$ converges in far fewer iterations. The number of iterations to converge plays a major part in determining an algorithm’s training time. The training time of the two parameterizations is shown in Figure 13 and 14 for Little and Big datasets, respectively. It can be seen that DBL$^n$ models are much faster than equivalent LR$^n$ models.

A comparison of rate of convergence of Negative-Log-Likelihood (NLL) of DBL$^2$ and LR$^2$ parameterization on some sample datasets is shown in Figure 15. It can be seen that, DBL$^2$ has a steeper curve, asymptoting to its global minimum much faster. For example, on almost all datasets, one can see that DBL$^2$ follows a steeper, hence more desirable, path toward convergence. This is extremely advantageous when learning from very few iterations (for example, when learning using Stochastic Gradient Descent based optimization) and, therefore, is a desirable property for scalable learning. A similar trend can be seen in Figure 16 for DBL$^3$ and LR$^3$.

Finally, let us present some comparison results about the speed of convergence of DBL$^n$ vs. LR$^n$ as we increase $n$. In Figure 17 we compare the convergence for $n = 1, n = 2$
Figure 12: Comparative scatter of iterations of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Big datasets.

Figure 13: Comparative scatter of Training time of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Little datasets.

Figure 14: Comparative scatter of Training time of DBL$^2$ and LR$^2$ (Left) and DBL$^3$ and LR$^3$ (Right) for Big datasets.
and \( n = 3 \) on the sample **Localization** dataset. It can be seen that the improvement that DBL\( n \) provides over LR\( n \) gets better as we go to deeper structures, i.e., as \( n \) becomes larger. The similar behaviour was observed for several datasets and, although studying rates
Figure 16: Comparison of rate of convergence of DBL$^3$ and LR$^3$ on several datasets. The X-axis (No. of iterations) is on log scale.

of convergence is a complicated matter and is outside the scope of this work, we anticipate this phenomenon to be an interesting venue of investigation for future work.
Figure 17: Compared rates of convergence of DBL\(^n\) vs. LR\(^n\), for \(n = 1, 2, 3\) on the sample Localization dataset. Y-axis is the negative log-likelihood.

<table>
<thead>
<tr>
<th></th>
<th>DBL(^2) vs. RF</th>
<th>DBL(^3) vs. RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>W-D-L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Little</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bias</td>
<td>51/3/21</td>
<td>52/2/21</td>
</tr>
<tr>
<td></td>
<td>(&lt;0.001)</td>
<td>(&lt;0.001)</td>
</tr>
<tr>
<td>Variance</td>
<td>33/3/39</td>
<td>28/5/42</td>
</tr>
<tr>
<td></td>
<td>0.556</td>
<td>(0.119)</td>
</tr>
<tr>
<td>0-1 Loss</td>
<td>40/3/32</td>
<td>37/3/35</td>
</tr>
<tr>
<td></td>
<td>0.409</td>
<td>0.906</td>
</tr>
<tr>
<td>RMSE</td>
<td>26/1/48</td>
<td>27/1/47</td>
</tr>
<tr>
<td></td>
<td>0.014</td>
<td>0.026</td>
</tr>
<tr>
<td>Big</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0-1 Loss</td>
<td>5/0/3</td>
<td>6/0/2</td>
</tr>
<tr>
<td></td>
<td>0.726</td>
<td>0.289</td>
</tr>
<tr>
<td>RMSE</td>
<td>5/0/3</td>
<td>5/0/3</td>
</tr>
<tr>
<td></td>
<td>0.726</td>
<td>0.726</td>
</tr>
</tbody>
</table>

Table 4: Win-Draw-Loss: DBL\(^2\) vs. RF and DBL\(^3\) vs RF. \(p\) is two-tail binomial sign test. Results are significant if \(p \leq 0.05\).

6.4 DBL\(^n\) vs. Random Forest

The two DBL\(^n\) models are compared in terms of W-D-L of 0-1 Loss, RMSE, bias and variance with Random Forest in Table 4. On Little datasets, it can be seen that DBL\(^n\) has significantly lower bias than RF. The variance of DBL\(^3\) is significantly higher than RF, whereas, difference in the variance is not significant for DBL\(^2\) and RF. 0-1 Loss results of DBL\(^n\) and RF are similar. However, RF has better RMSE results than DBL\(^n\) on Little datasets. On Big datasets, DBL\(^n\) wins on majority of datasets in terms of 0-1 Loss and RMSE.

The averaged 0-1 Loss and RMSE results are given in Figure 18. It can be seen that DBL\(^2\), DBL\(^3\) and RF have similar 0-1 Loss and RMSE across Little datasets. However, on Big datasets, the lower bias of DBL\(^n\) results in much lower error than RF in terms of both 0-1 Loss and RMSE. These averaged results also corroborate with the W-D-L results in Table 4 showing DBL\(^n\) to be a less biased model than RF.

The comparison of training and classification time of DBL\(^n\) and RF is given in Figure 19. It can be seen that DBL\(^n\) models are worst than RF in terms of the training time but better in terms of classification time.

7. Conclusion and Future Work

We have presented an algorithm for deep broad learning. DBL consists of parameters that are learned using both generative and discriminative training. To obtain the generative parameterization for DB, we first developed AnJE, a generative counter-part of higher-order logistic regression. We showed that DBL\(^n\) and LR\(^n\) learn equivalent models, but that
DBL\textsuperscript{n} is able to exploit the information gained generatively to effectively precondition the optimization process. DBL\textsuperscript{n} converges in fewer iterations, leading to its global minimum much more rapidly, resulting in faster training time. We also compared DBL\textsuperscript{n} with the equivalent AnJE and AnDE models and showed that DBL\textsuperscript{n} has lower bias than both AnJE and AnDE models. We compared DBL\textsuperscript{n} with state of the art classifier Random Forest and showed that DBL\textsuperscript{n} models are indeed lower biased than RF and on bigger datasets DBL\textsuperscript{n} often obtains lower 0-1 loss than RF.

There are a number of exciting new directions for future work.

• We have showed that DBL\textsuperscript{n} is a low bias classifier with minimal tuning parameters and has the ability to handle multiple classes. The obvious extension is to make it out-of-core. We argue that DBL\textsuperscript{n} is greatly suited for stochastic gradient descent based methods as it can converge to global minimum very quickly.

• It may be desirable to utilize a hierarchical DBL, such that \( hDBL^n = \{DBL^1 \ldots DBL^n\} \), incorporating all the parameters up till \( n \). This may be useful for smoothing the parameters. For example, if a certain interaction does not occur in the training data, at classification time one can resort to lower values of \( n \).

• In this work, we have constrained the values of \( n \) to two and three. Scaling-up DBL\textsuperscript{n} to higher values of \( n \) is greatly desirable. One can exploit the fact that many inter-
actions at higher values of $n$ will not occur in the data and hence can develop sparse implementations of DBL$^n$ models.

- Exploring other objective functions such as Mean-Squared-Error or Hinge Loss can result in improving the performance and has been left as a future work.

- The preliminary version of DBL that we have developed is restricted to categorical data and hence requires that numeric data be discretized. While our results show that this is often highly competitive with random forest using local cut-points, on some datasets it is not. In consequence, there is much scope for investigation of deep broad techniques for numeric data.

- DBL presents a credible path towards deep broad learning for big data. We have demonstrated very competitive error on big data and expect future refinements to deliver even more efficient and effective outcomes.

8. Code and Datasets

Code with running instructions can be downloaded from https://www.dropbox.com/sh/iw33mgcku9m2quc/AABXwYewVtm0mVE6KoyMPEVPa?dl=0.

9. Acknowledgments

This research has been supported by the Australian Research Council (ARC) under grants DP140100087, DP120100553, DP140100087 and Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under contracts FA2386-12-1-4030, FA2386-15-1-4017 and FA2386-15-1-4007.

Appendix A. Detailed Results

In this appendix, we compare the 0-1 Loss and RMSE results of DBL$^n$, AnDE and RF. The goal here is to assess the performance of each model on Big datasets. Therefore, results on 8 big datasets are reported only in Table 5 and 6 for 0-1 Loss and RMSE respectively. We also compare results with AnJE. Note A1JE is naive Bayes. Also DBL$^1$ results are also compared. Note, DBL$^1$ is WANBIA-C (Zaidi et al., 2013).

The best results are shown in bold font.

References


Table 5: 0-1 Loss values of A1JE, A2JE, A3JE, A1DE, A2DE, DBL¹, DBL², DBL³, RF and RF (numeric) on Big datasets.

<table>
<thead>
<tr>
<th></th>
<th>Localization</th>
<th>Poker-hand</th>
<th>Census-income</th>
<th>Covtype</th>
<th>Kddcup</th>
<th>MITFa-ceSetA</th>
<th>MITFa-ceSetB</th>
<th>MITFa-ceSetC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1JE</td>
<td>0.4938</td>
<td>0.4988</td>
<td>0.2354</td>
<td>0.3143</td>
<td>0.0091</td>
<td>0.0116</td>
<td>0.0268</td>
<td>0.0729</td>
</tr>
<tr>
<td>A2JE</td>
<td>0.3653</td>
<td>0.0763</td>
<td>0.2031</td>
<td>0.2546</td>
<td>0.0061</td>
<td>0.0106</td>
<td>0.0239</td>
<td>0.0630</td>
</tr>
<tr>
<td>A3JE</td>
<td><strong>0.2813</strong></td>
<td>0.0763</td>
<td>0.1674</td>
<td>0.1665</td>
<td>0.0053</td>
<td>0.0096</td>
<td>0.0215</td>
<td>0.0550</td>
</tr>
<tr>
<td>A1DE</td>
<td>0.3584</td>
<td>0.4640</td>
<td>0.0986</td>
<td>0.2387</td>
<td>0.0025</td>
<td>0.0124</td>
<td>0.0322</td>
<td>0.0417</td>
</tr>
<tr>
<td>A2DE</td>
<td>0.2844</td>
<td>0.1348</td>
<td>0.0682</td>
<td>0.1552</td>
<td>0.0023</td>
<td>0.0105</td>
<td>0.0325</td>
<td>0.0339</td>
</tr>
<tr>
<td>DBL¹</td>
<td>0.4586</td>
<td>0.4988</td>
<td><strong>0.0433</strong></td>
<td>0.2576</td>
<td>0.0017</td>
<td>0.0012</td>
<td>0.0047</td>
<td>0.0244</td>
</tr>
<tr>
<td>DBL²</td>
<td>0.3236</td>
<td><strong>0.0021</strong></td>
<td>0.0686</td>
<td>0.1381</td>
<td>0.0014</td>
<td>0.0002</td>
<td>0.0007</td>
<td>0.0007</td>
</tr>
<tr>
<td>DBL³</td>
<td>0.2974</td>
<td>0.0056</td>
<td>0.0557</td>
<td>0.0797</td>
<td>0.0013</td>
<td><strong>0.0001</strong></td>
<td><strong>0.0005</strong></td>
<td><strong>0.0005</strong></td>
</tr>
<tr>
<td>RF</td>
<td>0.2976</td>
<td>0.0687</td>
<td>0.0494</td>
<td>0.0860</td>
<td>0.0015</td>
<td>0.0012</td>
<td>0.0022</td>
<td>0.0013</td>
</tr>
</tbody>
</table>

Table 6: RMSE values of A1JE, A2JE, A3JE, A1DE, A2DE, DBL¹, DBL², DBL³ and RF on Big datasets.

<table>
<thead>
<tr>
<th></th>
<th>Localization</th>
<th>Poker-hand</th>
<th>Census-income</th>
<th>Covtype</th>
<th>Kddcup</th>
<th>MITFa-ceSetA</th>
<th>MITFa-ceSetB</th>
<th>MITFa-ceSetC</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1JE</td>
<td>0.2386</td>
<td>0.2382</td>
<td>0.4599</td>
<td>0.2511</td>
<td>0.0204</td>
<td>0.1053</td>
<td>0.1607</td>
<td>0.2643</td>
</tr>
<tr>
<td>A2JE</td>
<td>0.2115</td>
<td>0.1924</td>
<td>0.4231</td>
<td>0.2256</td>
<td>0.0170</td>
<td>0.1006</td>
<td>0.1516</td>
<td>0.2455</td>
</tr>
<tr>
<td>A3JE</td>
<td>0.1972</td>
<td>0.1721</td>
<td>0.3612</td>
<td>0.1857</td>
<td>0.0160</td>
<td>0.0954</td>
<td>0.1436</td>
<td>0.2293</td>
</tr>
<tr>
<td>A1DE</td>
<td>0.2090</td>
<td><strong>0.1890</strong></td>
<td>0.2217</td>
<td>0.2174</td>
<td>0.0103</td>
<td>0.1079</td>
<td>0.1746</td>
<td>0.1989</td>
</tr>
<tr>
<td>A2DE</td>
<td>0.2300</td>
<td>0.2382</td>
<td><strong>0.1807</strong></td>
<td>0.2254</td>
<td>0.0072</td>
<td>0.0347</td>
<td>0.0692</td>
<td>0.1360</td>
</tr>
<tr>
<td>DBL¹</td>
<td>0.2179</td>
<td>0.1970</td>
<td>0.250</td>
<td>0.1802</td>
<td>0.0068</td>
<td>0.0123</td>
<td>0.0248</td>
<td>0.0257</td>
</tr>
<tr>
<td>DBL²</td>
<td>0.2723</td>
<td><strong>0.0323</strong></td>
<td>0.2332</td>
<td>0.1494</td>
<td><strong>0.0065</strong></td>
<td><strong>0.0105</strong></td>
<td><strong>0.0198</strong></td>
<td><strong>0.0241</strong></td>
</tr>
<tr>
<td>RF</td>
<td>0.1939</td>
<td>0.1479</td>
<td>0.1928</td>
<td><strong>0.1336</strong></td>
<td>0.0072</td>
<td>0.0296</td>
<td>0.0484</td>
<td>0.0651</td>
</tr>
</tbody>
</table>


