Journal of Machine Learning Research 01 (2017) 1-28 Submitted 01/17; Revised 00/00; Published 00/00

On the Effectiveness of Discretizing Quantitative Attributes in Linear Classifiers

Nayyar A. Zaidi NAYYAR.ZAIDI@MONASH.EDU
Yang Du YDU32@STUDENT.MONASH.EDU
Geoffrey I. Webb GEOFF.WEBB@MONASH.EDU
Faculty of Information Technology
Monash University
VIC 3800, Australia.

Editor: xxxxxx xxxxxx

arXiv:1701.07114v1 [cs.LG] 24 Jan 2017

Abstract

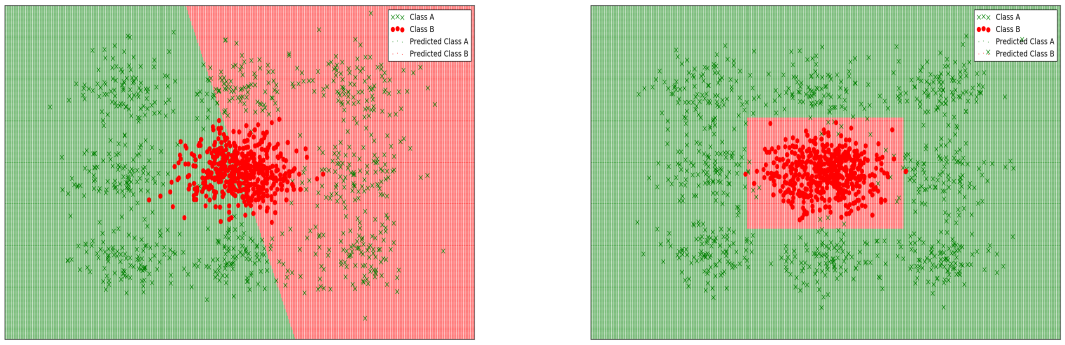
Learning algorithms that learn linear models often have high representation bias on real-world problems. In this paper, we show that this representation bias can be greatly reduced by discretization. Discretization is a common procedure in machine learning that is used to convert a quantitative attribute into a qualitative one. It is often motivated by the limitation of some learners to qualitative data. Discretization loses information, as fewer distinctions between instances are possible using discretized data relative to undiscretized data. In consequence, where discretization is not essential, it might appear desirable to avoid it. However, it has been shown that discretization often substantially reduces the error of the linear generative Bayesian classifier naive Bayes. This motivates a systematic study of the effectiveness of discretizing quantitative attributes for other linear classifiers. In this work, we study the effect of discretization on the performance of linear classifiers optimizing three distinct discriminative objective functions — logistic regression (optimizing negative log-likelihood), support vector classifiers (optimizing hinge loss) and a zero-hidden layer artificial neural network (optimizing mean-square-error). We show that discretization can greatly increase the accuracy of these linear discriminative learners by reducing their representation bias, especially on big datasets. We substantiate our claims with an empirical study on 42 benchmark datasets.

Keywords: discretization, classification, logistic regression, support vector classifier, big datasets, bias-variance analysis

1. Introduction

One of the many factors that affect the error of a learning system is its *representation bias* (van der Putten and van Someren, 2004), or, as it is also called, its *hypothesis language bias* (Mitchell, 1980). We define representation bias herein as the minimum loss of any model in the space of models available to the learner. It is clearly desirable in the general case to use a space of models that minimizes representation bias for a given problem. Learning algorithms that use linear models, such as logistic regression (LR) (Murphy, 2012) and support vector classifiers (SVC) (Chih-Jen, 2010), are very popular, possibly in part due to their lending themselves to convex optimization. In this paper we argue that learning algorithms that

©2017 Nayyar A. Zaidi, Yang. Du, Geoffrey I. Webb.



(a) Perceptron on original data

(b) Perceptron trained on discretized data. 3-bin equal frequency discretization is used.

Figure 1: Illustration of the effectiveness of a perceptron after EMD discretization on simple two-dimensional contrived data.

learn linear models often have high representation bias on real-world problems, and that often this bias can be reduced by discretization. We illustrate this in Figure 1 which shows on simple synthetic data how a linear classifier cannot create an accurate classifier on the numeric data but can when the data are discretized using simple univariate discretization.

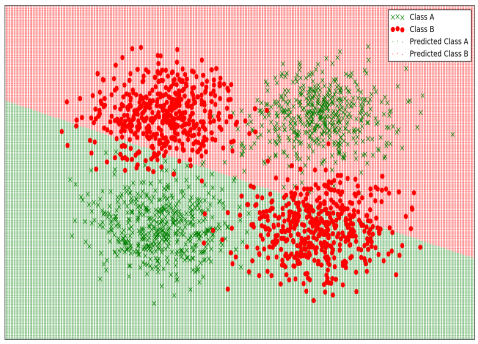
There are two important observations that motivate this work:

- A linear classifier with discretization is not linear with respect to the original data.
- Contrary to what might be thought given that discretization loses information, a linear classifier with discretization can reduce representation bias and may consequently reduce error.

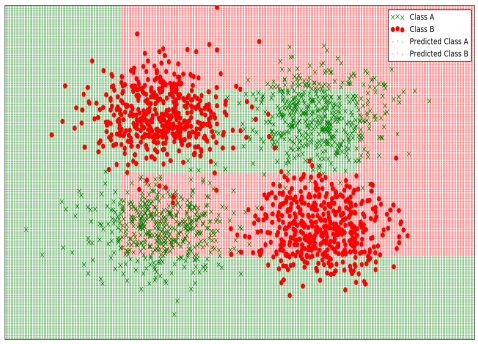
Note that we do not claim that discretization is the only useful feature transformation or a substitute for other approaches to creating non-linear classifiers, such as multi-layer perceptrons. The simple AND problem illustrated in Figure 1 is a case where where discretization is as effective as any other method for obtaining non-linear decision surfaces. However, there are problems where discretization will not be this effective. One example is the X-OR problem illustrated in Figure 2.

Note also that we do not claim that discretization always reduces representation bias. However, linear models make a strong implicit assumption, that the data are well modeled by a linear decision boundary. As illustrated in Figure 1 discretization can overcome this assumption. The startling result that we reveal is that doing so is often useful in practice. We show that discretization often reduces the bias of a linear classifier and that this reduction in bias frequently results in lower error when the data quantity is sufficiently large to minimize overfitting.

The rest of this paper is organized as follows. Some preliminary background and terminology is given in Section 2.1. We discuss discretization in general in Section 2.2. Linear



(a) Perceptron on original data



(b) Perceptron trained on discretized data. 4-bin equal frequency discretization is used.

Figure 2: Comparison of decision boundaries with and without discretization on a simple two-dimensional X-OR problem.

Notation	Description
I	Number of qualitative attributes
K	Number of quantitative attributes
n	Total number of attributes, $n = I + K$
N	Number of data points in $\mathcal{D}$
$\mathcal{D} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$	Data consisting of N objects
$\mathcal{L} = \{y^{(1)}, \ldots, y^{(N)}\}$	Labels of data points in $\mathcal{D}$
$\tilde{\mathrm{P}}(e)$	Actual probability of event (e)
$\mathrm{P}(e)$	Probability of event e
$\mathrm{P}(e \mid g)$	Conditional probability of event e given g
$\mathbf{x} = \langle x_0, x_1, \ldots, x_n \rangle$	An object (n-dimensional vector) and $\mathbf{x} \in \mathcal{D}$
Y	Random variable associated with class label
y	$y \in Y$. Class label for object. Same as x_0
C	$\lvert Y \rvert$, Number of classes
X_i	Random variable associated with qualitative attribute i
x_i	Actual value that X_i takes.
$\lvert X_i \rvert$	(Applicable only to qualitative attributes) Number of values of attribute X_i
β	LR parameter vector to be optimized
$\beta_{y,i}$	LR parameter associated with quantitative attribute i for class y
$\beta_{y,k,j}$	LR parameter associated with qualitative attribute k for class y taking value j
$\beta_{y,0}$	LR intercept term for class y.
λ	Regularization parameter

Table 1: List of symbols.

classifiers based on Conditional Log-Likelihood (CLL) Hinge Loss (HL) and Mean-square-error loss are discussed in Sections 2.3, 2.4 and 2.5 respectively. Optimization strategies for training these linear classifiers are discussed in Section 2.6. An overview of related work is given in Section 3. Experimental results are given in Section 4. We conclude in Section 5 with pointers to directions for future work.

2. Background

2.1 Terminology

In machine learning and data mining research, there exists variation in the terminology when it comes to characterizing the nature of an attribute (or feature). For example, 'continuous vs. discrete', 'numeric vs. categorical' and 'quantitative vs. qualitative'. We believe that the 'quantitative vs. qualitative' distinction is best suited for our study in this paper and hence, this is used throughout the paper.

Qualitative attributes are the attributes on which arithmetic operations can not be applied. The values of a qualitative attribute can be placed or categorized in distinct categories. Sometimes there exist a meaningful rank among these categories, resulting in distinction of ordinal and nominal among quantitative attributes. For example, *Student Grade:* {*HD, D, C, P, F*} and *Pool Depth:* {*Very Deep, Deep, Shallow*} are ordinal attributes, while *Marital Status:* {*Married, Never-married, Divorced, Widow, Widower*} and *Nationality:* {*Australian, American, British*} are nominal attributes.

Quantitative attributes, on the hand, are the attributes on which arithmetic operations can be applied. They can be both discrete and continuous. For example, *Number of Children* is a discrete-quantitative attribute (values determined by counting), whereas *Temperature* is a continuous-quantitative attribute (values determined by measuring).

A list of the various symbols used in this work is given in Table 1.

2.2 Discretization

Discretization is a common process in machine learning that is used to convert a quantitative into a qualitative attribute (Liu et al., 2002; Garcia et al., 2013). The need for discretization originates from the facts that some classifiers can only handle, and some others sometimes to operate better with qualitative attributes. The process involves finding cut-points within the range of the quantitative attribute and to group values into intervals based on these cut-points. This removes the ability to distinguish between data points falling in the same interval. Therefore, discretization entails information loss.

Discretization methods can be categorized into two categories: *Supervised* and *Unsupervised*. In the unsupervised case, class information is not used during cut-point determination process. Popular approaches are equal-frequency and equal-width discretization. Equal-width discretization (EWD) divides the quantitative attribute's range (maximum value x_i^{max} and minimum value x_i^{min}) into k equal-width intervals where k is provided by the user. Each interval will have a width of $w = \frac{x_i^{max} - x_i^{min}}{k}$. Equal-frequency discretization (EFD), on the other hand, divides the sorted values of a quantitative attribute such that each interval has approximately k number of data points. Each interval will contain N/k data points. It is also important that data points with identical value are placed in the same interval, therefore, in practice, each interval will have slightly different number of data points. Choosing EWD or EFD and the number of bins is problem specific and can have huge impact on the overall performance of any model. Of course, choosing a large k will result in less information loss, but can result in over-fitting on small datasets.

Supervised discretization methods, on the other hand, utilize the class information of the data point to better define the cut-points. For example, state-of-the-art discretization

technique Entropy-Minimization Discretization (EMD) sorts the quantitative attribute's values and then finds the cut-point such the information gain is maximized across the splits (Kohavi and Sahami, 1996). The technique is applied recursively on the successive splits and the minimum-description-length (MDL) criterion is used to determine when to stop splitting.

2.3 Linear Classifier - CLL

A Logistic Regression classifier optimizes the conditional log-likelihood (CLL) which is defined as:

$$\mathrm{CLL}(\beta) \quad = \quad \sum_{l=1}^{N} \log \mathrm{P}(y^{(l)} \,|\, \mathbf{x}^{(l)}), \tag{1}$$

where

$$\mathrm{P}(y^{(l)} \,|\, \mathbf{x}^{(l)}) \quad = \quad \frac{\exp(\beta_{y,0} + \beta_y^T \mathbf{x}^{(l)})}{\sum_{c=1}^{C} \exp(\beta_{y,0} + \beta_c^T \mathbf{x}^{(l)})}. \tag{2}$$

The term $\beta_{y,0} + \beta_y^T \mathbf{x}^{(l)}$ is expanded as: $\beta_{y,0} x_0^{(l)} + \beta_{y,1} x_1^{(l)} + \beta_{y,2} x_2^{(l)} + \cdots + \beta_{y,n} x_n^{(l)}$, where x_0 can be assumed to be 1 for all data points. Since the objective function as defined in Equation 1, is linear in x, it is a linear classifier.

Equation 2 leads to a multi-class softmax objective function. Since, a set of parameters are learned for each class, we have made this distinction explicit with subscript y in parameter notation, that is, $\beta_{y,j}$ denotes a parameter for class y and attribute j. Typically, an LR minimizes the negative of the CLL known as negative log-likelihood (NLL), which is defined as:

$$\mathrm{NLL}(\beta) \quad = \quad -\sum_{l=1}^{N} \left(\left(\beta_{y,0} + \beta_y^T \mathbf{x}^{(l)} \right) - \log \left(\sum_{c=1}^{C} \exp(\beta_{c,0} + \beta_c^T \mathbf{x}^{(l)}) \right) \right). \tag{3}$$

Note, in the following, for simplicity, we will drop the superscript (l) notation. It should be noted that many software libraries for multi-class LR are either based on implementing multi-class (softmax) objective function of Equation 3 or they optimize a more simpler binary objective function of the following form:

$$\mathrm{NLL}(\beta) \quad = \quad \sum_{l=1}^{N} \left(\log \left(1 + \exp(\beta_0 + \beta^T \mathbf{x}^{(l)}) \right) \right), \tag{4}$$

and solve a one-versus-all classification problem. Note that in the case of binary classifiers, there is only one set of parameters for the two classes as oppose to C set of parameter that needed to be optimized for the softmax case. At classification time, one needs to apply C different trained LR classifiers and choose one with the highest probability.

Nonetheless, optimizing a standard LR with NLL based either on Equation 3 or Equation 4 requires substantial input manipulation, i.e., appending 1 to all data points and then, converting qualitative attributes using one-hot-encoding. For example, a qualitative

attribute X_j taking values $\{a, b, c\}$, will be converted into three attributes X_j, X_{j+1}, X_{j+2}, each taking values either 0 or 1. An alternative to manipulating the input is to modify the model and optimize the following objective function instead:

$$\begin{aligned} \text{NLL}(\beta) &= -\sum_{l=1}^{N} \left(\left(\beta_{y,0} + \sum_{i=1}^{I} \beta_{y,i} x_i + \sum_{k=1}^{K} \sum_{j=1}^{|X_k|} \beta_{y,k,j} \mathbf{1}_{X_k = x_j} \right) \right. \\ &\quad \left. - \log \left(\sum_{c=1}^{C} \exp(\beta_{c,0} + \sum_{i=1}^{I} \beta_{c,i} x_i + \sum_{k=1}^{K} \sum_{j=1}^{|X_k|} \beta_{c,k,j} \mathbf{1}_{X_k = x_j}) \right) \right). \end{aligned} \tag{5}$$

Note that the models expressed in Equation 3 and 5 are exactly equivalent and will lead to the same results. The only difference is that the model in Equation 3 requires converting all qualitative attributes into quantitative ones using one-hot-encoding, whereas the model in Equation 5 does not. Equation 5 can be simplified even further – for datasets with only qualitative attributes, and including only terms that are not canceled out, we have:

$$\begin{aligned} \text{NLL}(\beta) &= -\sum_{l=1}^{N} \left(\beta_{y,0} + \sum_{k=1}^{K} \beta_{y,k,j} \mathbf{1}_{X_k = x_j, Y = y} \right. \\ &\quad \left. - \log \left(\sum_{c=1}^{C} \exp(\beta_{c,0} + \sum_{k=1}^{K} \beta_{c,k,j} \mathbf{1}_{X_k = x_j, Y = c}) \right) \right). \end{aligned} \tag{6}$$

Instead of converting qualitative attributes into quantitative ones and using the model of Equation 3, one can convert quantitative attributes into qualitative ones using discretization methods as discussed in Section 2.2 and use the model of Equation 6. It can be seen that with Equation 3, the number of parameters optimized are: $(C-1) + (C-1)n$. Whereas, with Equation 6, $(C-1)+(C-1)\sum_{i=1}^{n} |X_i|$ parameters are optimized. Since the two models are not equivalent, this will result in different training time, speed and rate of convergence and of course, classifications.

2.4 Linear Classifier – Hinge Loss

Hinge Loss (HL) is widely used as an alternative to CLL and has been the basis of Support Vector Machines. A classifier optimizing either a Hinge Loss objective function or its variant is a linear classifier and is known as the Support Vector Classifier (SVC). Here we define L2-Loss HL as:

$$\text{HL}(\beta) = \sum_{l=1}^{N} \max(0, 1 - y\beta^T \mathbf{x})^2. \tag{7}$$

An alternative is L1-Loss HL which is equal to: $\sum_{l=1}^{N} \max(0, 1 - y\beta^T \mathbf{x})$. In this work, we will focus only on the L2-Loss. In practice, a penalty term is also added for regularizing the objective function as:

$$\text{HL}(\beta) = \frac{1}{2} ||\beta^T \beta||^2 + \lambda \sum_{l=1}^{N} (\max(0, 1 - y\beta^T \mathbf{x}))^2, \tag{8}$$

where λ is the regularization parameter. We will discuss the gradient and Hessian of this objective function later in Section 2.6.

2.5 Linear Classifier – Mean-Square-Error

Another linear classifier is based on optimizing the Mean-Square-Error (MSE) objective function and is defined as:

$$\text{MSE}(\beta) \quad = \quad \frac{1}{2}\sum_{l=1}^{N}\sum_{c=1}^{C}(\tilde{\text{P}}(c\,|\,\mathbf{x}) - \text{P}(c\,|\,\mathbf{x}))^2, \tag{9}$$

where $\text{P}(c\,|\,\mathbf{x})$ is given in Equation 2 and $\tilde{\text{P}}(c\,|\,\mathbf{x})$ is the actual probability of class c given data instance $\mathbf{x}$. This will be a vector of size C with all zeros except at the location of the label of $\mathbf{x}$, where it will be 1 (assuming there are no duplicate data points in the dataset). The objective function of Equation 9 is similar to that optimized by artificial neural-networks (ANN). However, in ANN, $\text{P}(c\,|\,\mathbf{x})$ is defined in terms of multiple layers. We can interpret Equation 9 as the objective function of a zero-layer ANN.

2.6 Optimization

There is no closed form solution to optimizing the negative log-likelihood, hinge loss and mean-square-error objective function, and, therefore, one has to resort to iterative minimization procedures such as gradient descent or quasi-Newton. An iterative optimization procedure generates a sequence $\{\beta^k\}_{k=1}^{\infty}$ converging to an optimal solution. At every iteration k, the following update is made: $\beta^{k+1} = \beta^k + \mathbf{s}^k$, where $\mathbf{s}^k$ is the search direction vector. The following equation plays the pivotal role as it holds the key to obtain $\mathbf{s}^k$ by solving a system of linear equations:

$$\nabla^2 f(\beta^k)\mathbf{s}^k = -\nabla f(\beta^k), \tag{10}$$

where f is the objective function that we are optimizing. There are two very important issues that must be addressed when solving for search direction vector using Equation 10 (Nocedal and Wright, 2006). First, it can be infeasible to explicitly compute and store the Hessian, especially on high-dimensional data. Second, the solution obtained using Equation 10, does not guarantee convergence. There are three main strategies for addressing the first issue:

- Consider $\nabla^2 f(\beta^k)$ to be an identity matrix – in this case, $\mathbf{s}^k = -\nabla f(\beta^k)$. This leads to a family of algorithms known as first-order methods such as Gradient Descent, Coordinate Descent, etc.

- Do not compute $\nabla^2 f(\beta^k)$ directly, but approximate it from the information present in $\nabla f(\beta^k)$ instead. This property is useful for large scale problems where we cannot store the Hessian matrix. This leads to approximate second-order methods known as quasi-Newton algorithms, for example, L-BFGS which, is considered to be the most efficient algorithm (de-facto standard) for training LR.

- Third, use standard 'direct algorithms' for solving a system of linear equations such as Gaussian elimination to solve for $\mathbf{s}^k$. Or, use any one of the iterative algorithm

such as conjugate gradient, etc. For large datasets, generally iterative methods are preferable over direct methods, as the former requires computing the whole Hessian matrix. The optimization method now has two layers of iterations. An outer layer of iteration to update β^k, and an inner layer of iterations to find Newton direction $\mathbf{s}^k$. In practice, one can only use an approximate Newton direction in early stages of the outer iterations. This method is known as 'Truncated Newton method' (Nash, 2000).

It should be noted that these methods differ in terms of the speed-of-convergence, cost-per-iteration, iterations-to-convergence, etc. For example, Coordinate Descent updates one component of β at every iteration, so the cost-per-iteration is very low, but iterations-to-convergence will be very high. On the other hand, Newton methods, will have high cost-per-iteration, but very low number of iterations-to-convergence. The three methods described above are all affected by the scaling of the axis. Therefore, scaling quantitative attributes or converting quantitative into qualitative attributes will effect the speed and the quality of the convergence.

The second issue can be addressed by adjusting the length of the Newton direction. For that, two techniques are mostly used – line search and trust region. Line search methods are standard in optimization research. We can modify Equation 10 as $\nabla^2 f(\beta^k)\mathbf{s}^k = -\eta^k \nabla f(\beta^k)$, where η^k is known as the step-size. Standard line searches obtain an optimal step-size as a solution to the following sub-optimization problem: $\eta^k = \text{argmin}_\eta f(\beta^k + \eta \mathbf{s}^k)$. Trust-region methods, unlike line search, are relatively new in optimization research. Trust-region methods first find a region around the current solution – in this region, a quadratic (or linear) model is used to approximate the objective function. The step size is determined based on the goodness of fit of the approximate model. If a significant decrease in the objective function is achieved with a forward step, the approximated model is a good representative of the original objective function and vice-versa. The size of the (trust) region is specified as a spherical area of size Δ_k. The convergence of the algorithm is guaranteed by controlling the size of the region which (in each iteration) is proportional to the reduction in the value of objective function in the previous iteration.

In the following we will define the gradient and Hessian of the three objective functions conditional log-likelihood, hinge loss and mean square error. Note, we only define the gradient and the Hessian for qualitative attributes here. For softmax CLL, $\nabla f(\beta^k)$ and $\nabla^2 f(\beta^k)$ can be written as:

$$\frac{\partial \text{NLL}(\beta)}{\partial \beta_{y',i}} = \sum_{l=1}^{N} (\mathbf{1}_{y=y'} - \text{P}(y'|\mathbf{x}))x_i,$$

$$\frac{\partial^2 \text{NLL}(\beta)}{\partial \beta_{y',i} \partial \beta_{y'',j}} = -\sum_{l=1}^{N} (\mathbf{1}_{y'=y''} - \text{P}(y'|\mathbf{x}))\text{P}(y''|\mathbf{x})x_i x_j.$$

For Hinge-loss, the (sub-) gradients can be written as:

$$\frac{\partial \text{HL}(\beta)}{\partial \beta_i} = 2\sum_{l=1}^{N'} (y\beta^T \mathbf{x} - 1)yx_i,$$

where N', are the instances for which $y\beta^T\mathbf{x} < 1$ is true. Similarly the Hessian can be written as:

$$\frac{\partial^2 \mathrm{HL}(\beta)}{\partial\beta_i\partial\beta_j} = 2\sum_{l=1}^{N'} x_i x_j$$

For MSE, one can write the gradients as:

$$\frac{\partial \mathrm{MSE}(\beta)}{\partial\beta_{j,x_j,k}} = \sum_{l=1}^{N}\sum_{c=1}^{C}(\mathbf{1}_{y=c} - \mathrm{P}(c|\mathbf{x}))(\mathbf{1}_{k=c} - \mathrm{P}(k|\mathbf{x}))x_i,$$

and the Hessian can be written as:

$$\begin{aligned}\frac{\partial^2 \mathrm{MSE}(\beta)}{\partial\beta_{j,x_j,k}\partial\beta_{j',x_{j'},k'}} = -\sum_{l=1}^{N}[\ &(-1)(\mathbf{1}_{k'=c} - \mathrm{P}(k'|\mathbf{x}))\mathrm{P}(c|\mathbf{x})\mathbf{1}_{j=j'} + \\ &(\mathbf{1}_{y=c} - \mathrm{P}(c|\mathbf{x}))(-1)(\mathbf{1}_{k'=k} - \mathrm{P}(k'|\mathbf{x}))P(k|\mathbf{x})\mathbf{1}_{j=j'}\mathrm{P}(c|\mathbf{x}) + \\ &(\mathbf{1}_{y=c} - \mathrm{P}(c|\mathbf{x}))(\mathbf{1}_{k=c} - \mathrm{P}(k|\mathbf{x}))(\mathbf{1}_{k'=c} - \mathrm{P}(k'|\mathbf{x}))\mathrm{P}(c|\mathbf{x})\mathbf{1}_{j=j'}]\ x_i x_j\end{aligned}$$

3. Related Work

Discretization is often motivated by a need to adapt data for a model that cannot handle quantitative attributes. In Statistics and many of its related and applied branches (such as epidemiology, medical research and consumer marketing), it goes by names of 'dichotomization' and 'categorization' (where the two techniques differ as the former splits the measurement scale into two while the later can have more than two categories) – and has been examined in many studies (Irwin and McClelland, 2003; MacCallum et al., 2002; Greenland, 1995). However, in most of these studies a majority opinion is against the use of dichotomization – and for categorization, it is advised to be used with caution. The main reason cited for this is that dichotomization and categorization lead to information loss since the variability among the members of the group is subsumed. For example, Altman and Royston (2006) write:

> ... Firstly, much information is lost, so the statistical power to detect a relation between variable and patient outcome is reduced ... and considerable variability may be subsumed within each group. Individuals close to but on opposite sides of cut-point are characterized as being very different rather than very similar ...

In practical machine learning, the common practice is to discretize an attribute only if necessary (i.e., if a model expects categorical attributes). An exception is for Bayesian classifiers, where it is common practice to discretize numeric attributes (Yang and Webb, 2009).

The ambivalence towards discretization is understandable. Obviously, the quality (and sometime quantity) of data is the key to training accurate models and hence getting good results. In many cases, the data are the result of costly and time-consuming efforts (for example in breast cancer research where there are several stake-holders involved just to obtain a few attributes of the data). Losing some of the data (or more precisely, losing some

distinction among the instances) due to discretization should be undesirable. However, a number of motivations for discretization have been put forward:

- Discretization can lead to simplification of statistical analysis. For example, if a quantitative attribute is split on the median, then one can compare the two groups based on t, χ^2 or some other test to estimate the difference between the two groups. This may ease interpretation and presentation of results (Altman and Royston, 2006).

- If there is error in the measurement scale, discretization can improve the performance of the model by reducing the contamination (Flegal and Keyl, 1991; Reade-Christopher and Kupper, 1991; Fung and Howe, 1984; Shentu and Xie, 2010).

- In many domains there exist pre-defined (or standard) thresholds to convert a quantitative to a qualitative scale. In these cases, a discretized attribute might better represent the task at hand as it will be more interpretable or have distinct significance. For example, in medical research doctors might better interpret blood-pressure as high and low rather than on a numeric scale.

- A discretized attribute might be better utilized than the quantitative attribute by the learning system. For example, consider a classifier that relies on estimation of conditional probabilities such as $\mathrm{P}(x_i \mid y)$. If X_i is quantitative, x_i can take infinite many values and if the number of training samples are small, reliable estimation of $\mathrm{P}(x_i \mid y)$ from the data is not possible. A common approach is to impose a parametric model to estimate the value of $\mathrm{P}(x_i \mid y)$ based on this model in which case the accuracy will depend on the appropriateness of the parametric model selected. Discretization can obviate this problem. By converting a quantitative attribute X_i into a qualitative one X_i^*, the probabilities will take the form of $\mathrm{P}(x_i^* \mid y)$ which may be reliably estimated from the data as there will be many x_i values falling into the same interval (Yang and Webb, 2009).

- The final reason for discretization has to do with overcoming a model's assumptions. It might be the case that discretization help avoid some strong assumption that the learner makes about the data. If those assumptions are correct, discretization will have a negative impact, but if those assumptions are false, discretization may lead to better results (Altman et al., 1994). It is this final motivation that we examine herein.

The effect of discretization on various classification algorithms such as naive Bayes, Support Vector Machines and Random Forest is discussed in Lustgarten et al. (2008). On many biomedical datasets, it is shown that discretization can greatly improve the performance of the learning algorithm. The role of discretization as feature selection technique is also explored. On various contrived datasets, Maleki et al. (2009) studied the effect of discretization on the precision and recall of various classification methods.

The effectiveness of discretization for naive Bayes classifier is relatively well studied (Hsu et al., 2000; Dougherty et al., 1995; Yang and Webb, 2009). Dougherty et al. (1995) conducted an empirical study of naive Bayes with four well-known discretization methods and found that all the discretization methods result in significantly reducing error relative to a naive Bayes that assumes a Gaussian distribution for the continuous variables. Hsu

et al. (2000) attributes this to the *perfect aggregation* property of Dirichlet distributions. In naive Bayes settings, a discretized continuous distribution is assumed to have a categorical distribution with Dirichlet priors. The perfect aggregation property of Dirichlet implies that we can learn the class-conditional probability of the discretized interval with arbitrary accuracy. It is also shown that there exists a *partition independence assumption*, by virtue of that, Dirichlet parameters corresponding to a certain interval depend only on the area below the curve of the probability distribution function, but is independent of the shape of the curve in that interval.

4. Experiments

In this section, we compare the performance of linear classifier with discretized linear classifier on various datasets from the UCI repository (Lichman, 2017).

We denote linear classifier optimizing the conditional log-likelihood as LR, a linear classifier optimizing the Hinge loss as SVC (support vector classifier) and a linear classifier optimizing the mean-square-error as ANN^0 (artificial neural network with zero hidden layers) – their discrete counterparts are denoted as LR(d), SVC(d) and ANN^0(d) respectively.

In the remainder of this paper, when discussing results, we will collectively refer to LR, SVC and ANN^0 as linear classifiers and denote them by LC. We will collectively refer to LR(d), SVC(d) and ANN^0(d) as discretized linear classifiers and denote them by LC^d.

The details of datasets used in this work are given in Appendix B. For discretized linear classifiers, different supervised and unsupervised discretization techniques were considered. Since, this is not a comparative study on the relative efficacies of various discretization techniques for linear classifiers, we only report results with supervised entropy-based discretization of Fayyad and Irani (1992), which we found gives better results than other discretization methods such as equal-frequency, equal-width, etc.

Each algorithm is tested on each dataset using either 5 or 10 rounds of 2-fold cross validation.

During the presentation of results, we split our datasets into two categories – *Big* and *Little*. The *Big* category comprises of datasets with more than $100,000$ instances and the *Little* category comprises of the remaining datasets with $< 100,000$ instances.

We compare four different metrics: 0-1 Loss, RMSE, Bias and Variance. We also compare training-time, testing time, and rate of convergence. As discussed in Section 1, the reason for performing bias-variance estimation is that it provides insights into how the learning algorithm might be expected to perform with varying amounts 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). There are a number of different bias-variance decomposition definitions. In this research, we use the bias and variance definitions of Kohavi and Wolpert (1996) together with the repeated cross-validation bias-variance estimation method proposed by Webb (2000).

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 $p \leq 0.05$ and shown in bold.

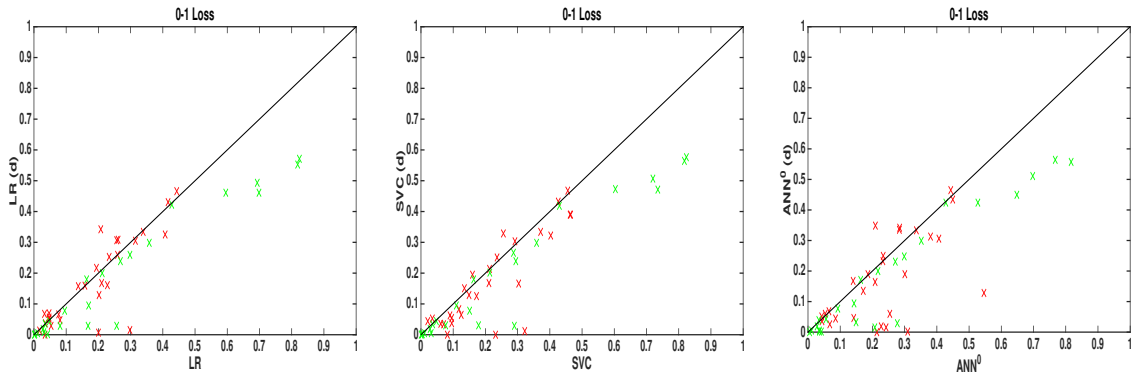


Figure 3: Comparative scatter of `0-1 Loss` results for linear and discretized-linear classifiers. Linear classifiers are on the X-axis whereas discretized-linear classifiers are on the Y-axis. For points below the diagonal line, discretized-linear classifiers win. Results on *Big* datasets are shown in green, whereas results on *Little* datasets are shown in red.

For hinge-loss, a dataset with more than two classes was transformed into a binary dataset. Data points belonging to the majority class were assigned to class A and the remaining data points were assigned to class B.

Missing values of the quantitative attribute were replaced with the mean of the attribute values whereas missing values of the qualitative attribute were treated as a distinct attribute value.

Quantitative attributes were also normalized between 0 and 1, as this is often recommended for gradient-based optimization methods.

Three optimization methods – gradient descent, quasi-Newton, Trust-region based Newton method (TRON) were used. We found TRON to be converging relatively faster than the other methods. Therefore, in the following, we report results with TRON optimization only. However, it is worth mentioning that a similar pattern of results was seen between LC and LC$^{\text{d}}$ for the other optimization methods.

4.1 Comparison of the Accuracy of LC$^{\text{d}}$ and LC

In this section, we compare the accuracy of LC$^{\text{d}}$ and LC in terms of their 0-1 Loss and RMSE on 52 datasets. Results are shown in Figures 3 and 4. It can be seen that the three LC$^{\text{d}}$ classifiers result in much better accuracy than their corresponding LC. In the scatter plots, results on *Big* datasets are shown in green dots, whereas results on *Little* datasets are shown in red dots. It can be seen that, on almost all *Big* datasets, LC$^{\text{d}}$ leads to higher accuracy (most green-dots are below the diagonal line). It can also be seen that some of the differences are substantial – this shows the effectiveness of discretization on LR, SVC and ANN^0 especially for big datasets.

A comparison of the win-draw-loss between the two models is given in Table 2. It can be seen that on big datasets, LC$^{\text{d}}$ wins on all except on 2 datasets – very promising result. This proves our hypothesis that on big datasets, discretization leads to low-bias non-linear classifier resulting in far superior results than a linear classifier with no discretization. On

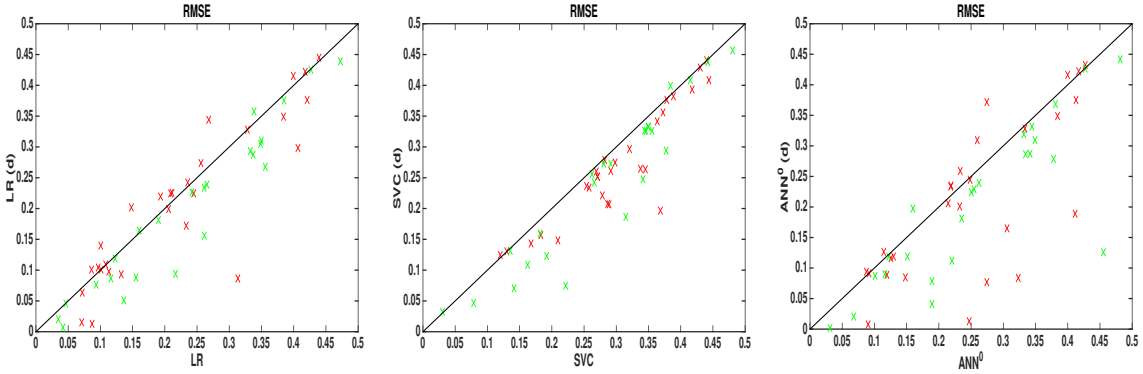


Figure 4: Comparative scatter of `RMSE` results for linear and discretized-linear classifiers. Linear classifiers are on the X-axis whereas discretized-linear classifiers are on the Y-axis. For points below the diagonal line, discretized-linear classifiers win. Results on *Big* datasets are shown in green, whereas results on *Little* datasets are shown in red.

	LR(d) vs. LR		**SVC(d) vs. SVC**		**ANN^0(d) vs. ANN^0**	
	W-D-L	p	W-D-L	p	W-D-L	p
All Datasets						
0-1 Loss	35/1/16	**0.011**	39/1/12	**<0.001**	41/1/10	**<0.001**
RMSE	34/1/7	**0.016**	39/1/12	**<0.001**	47/1/4	**<0.001**
Big Datasets						
0-1 Loss	22/0/2	**<0.001**	22/0/2	**<0.001**	23/0/1	**<0.001**
RMSE	22/0/2	**<0.001**	22/0/2	**<0.001**	22/0/2	**<0.001**
Small Datasets						
0-1 Loss	13/1/14	1.000	17/1/10	0.247	18/1/9	0.087
RMSE	12/1/15	0.701	17/1/10	0.247	25/1/2	**<0.001**

Table 2: Win-Draw-Loss comparison of `0-1 Loss` and `RMSE` of LR(d) vs. LR, SVC(d) vs. SVC and ANN^0(d) vs. ANN^0. Significant results are shown in bold.

small datasets, discretization is significantly effective for SVC and non-significantly effective for ANN^0. Note that on small datasets, LR(d) and LR leads to similar performances with 13 wins and 14 losses for 0-1 Loss and 12 wins and 15 losses for RMSE. However, one should take into account that the scale of LR(d) wins is much higher than that of LR. This can be seen from the spread of red-dots in the left-most plots of Figures 3 and 4.

4.2 Comparison of the Bias and Variance

Figures 5 and 6 present scatter plots of the bias and variance of LC and LC$^{\text{d}}$ classifiers. It can be seen that the three LC$^{\text{d}}$ classifiers lead to low-bias and high-variance models. Note that we present a bias-variance analysis on only *Little* datasets. This is because the software we have for obtaining bias-variance estimates is single threaded and could not benefit from

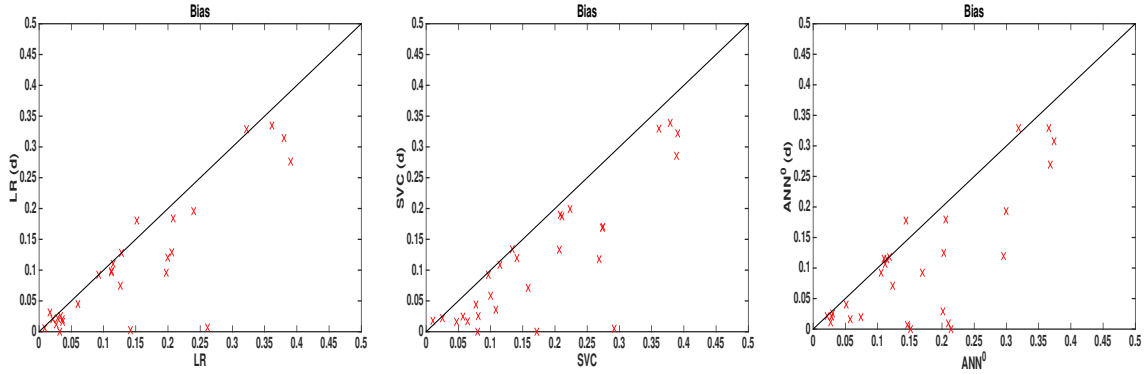


Figure 5: Comparative scatter of `Bias` results for LC and LC$^{\mathrm{d}}$ classifiers. LC are on the X-axis whereas LC$^{\mathrm{d}}$ are on the Y-axis. For points below the diagonal line, LC$^{\mathrm{d}}$ win.

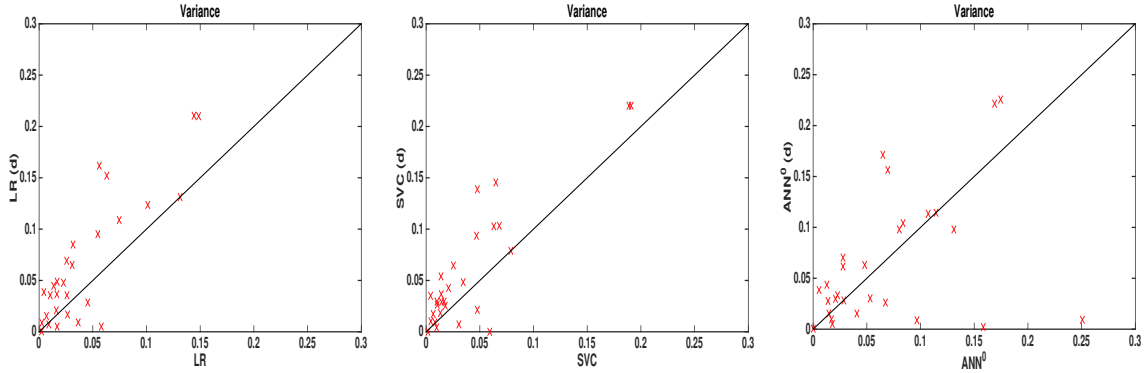


Figure 6: Comparative scatter of `Variance` results for LC and LC$^{\mathrm{d}}$ classifiers. LC are on the X-axis whereas LC$^{\mathrm{d}}$ classifiers are on the Y-axis. For points below the diagonal line, LC$^{\mathrm{d}}$ classifiers win.

the high-performance environment in which most of our experiments were run. As a result it was not feasible to run these experiments on the larger datasets. Nonetheless, results confirm our hypothesis that LC$^{\mathrm{d}}$ classifiers tend to have lower bias than LC.

4.3 Comparison of the Convergence Curves of LC$^{\mathrm{d}}$ and LC

As training of both LC$^{\mathrm{d}}$ and LC classifiers are based on iterative optimization algorithms, they produce a sequence of values as part of their training, i.e., of their objective function which (should ideally) decrease with successive iterations until convergence. A technique that leads to the global minimum faster (steeper curve) and in fewer iterations (shorter curve) is desirable. Note that LC and LC$^{\mathrm{d}}$ have different models (and parameterizations) and, therefore, the optimization space for the two problems is also very different. In the following, let us compare the convergences of LC and LC$^{\mathrm{d}}$ on some sample datasets. A similar trend was observed on all datasets, here we report results on nine representative datasets only.

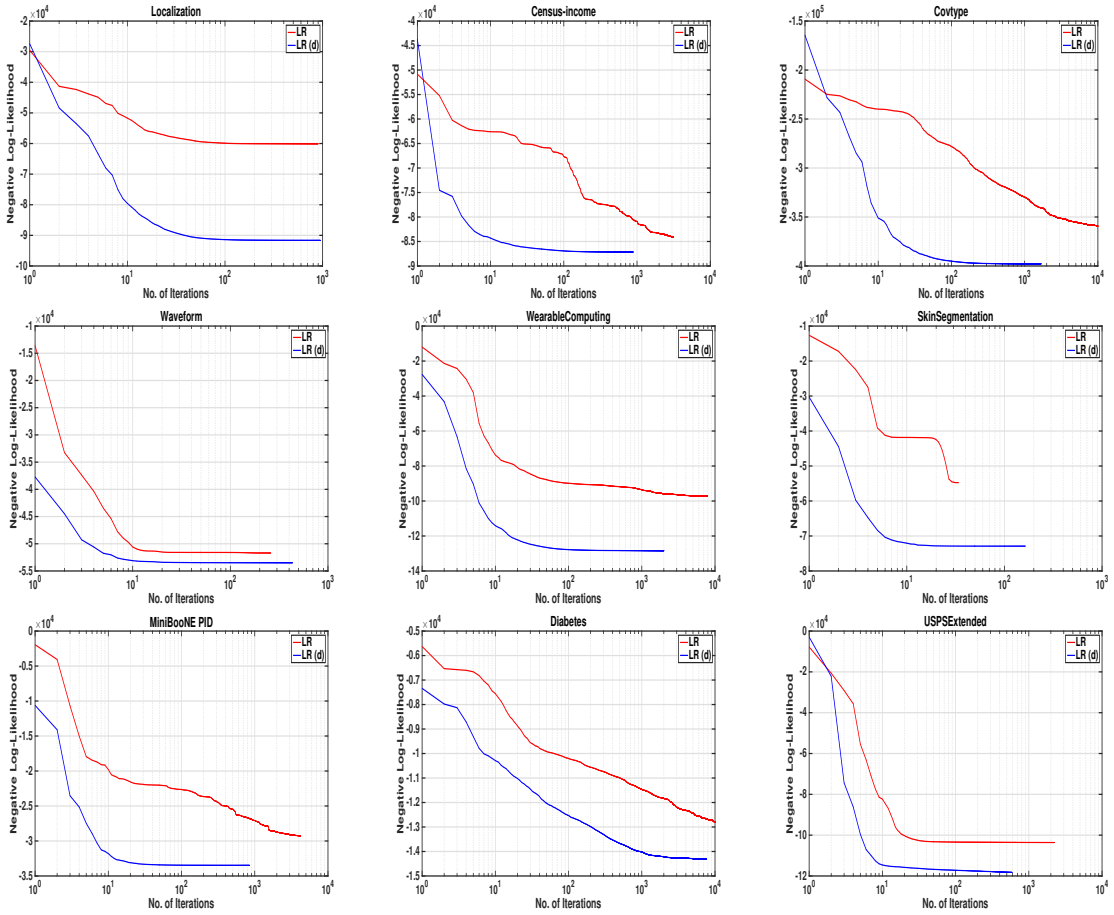


Figure 7: Comparison of the rate of convergence of LR and LR(d) on nine sample datasets. The X-axis is on log scale.

A comparison of the variation in NLL objective function for LR and LR(d) is shown in Figure 7. It can be seen that LR(d) has steeper curve – that is, it asymptotes to its global minimum much quickly. It is also important to see that LR(d) leads to much lower NLL. Better accuracy of LR(d) is the result of this much lower NLL.

Figures 8 shows the variation in HL for SVC and SVC(d) whereas, Figure 9 shows the variation in MSE for ANN^0(d) and ANN^0. A similar trend to NLL can be seen, that is SVC(d) and ANN^0(d) leading to a better value of the objective function while converging more rapidly.

4.4 Comparison of the Learning and Classification Time LC^d and LC

In this section, we compare the training and classification time of LC^d and LC. It can be seen from Figure 10 that LR(d) and ANN^0(d) are slightly faster than LR and ANN^0 respectively (majority of points below the diagonal line), whereas SVC(d) and SVC have similar training-time profiles. We already have seen the superior classification performance of LC^d classifiers.

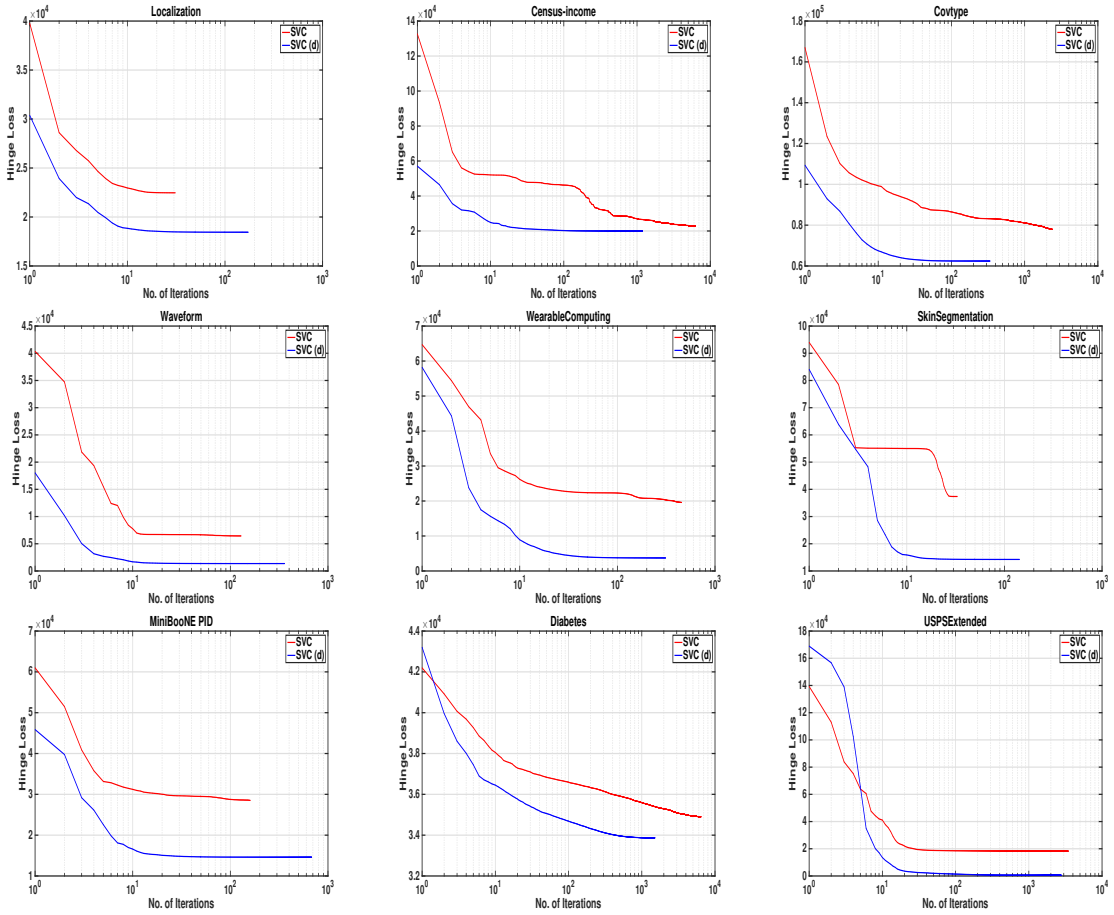


Figure 8: Comparison of the rate of convergence of SVC and SVC(d) on nine sample datasets. The X-axis is on the log scale.

These training-time results are extremely encouraging as they suggest that LC^d can result in much better classification accuracy without compromising computational performance. We have reported the results only on four big datasets. This is because, the results were obtained by running the jobs on a local-desktop computer (i.e., a controlled set-up), rather than the heterogeneous cluster-computing environment in which most of the experimentation was performed. The scatter plots of classification time results for LC and LC^d are presented in Figure 11. It can be seen that LC has slightly faster classification time than LC^d. However, in most cases the difference in the magnitude is small.

5. Conclusion and Future Works

In this paper, we study the role of discretization for linear classifiers in machine learning. Current practice is primarily to apply discretization only when the learner requires qualitative data. Overall, there exists some aversion to discretization as it loses information. We argue that discretization – despite losing information, can help model non-linear rela-

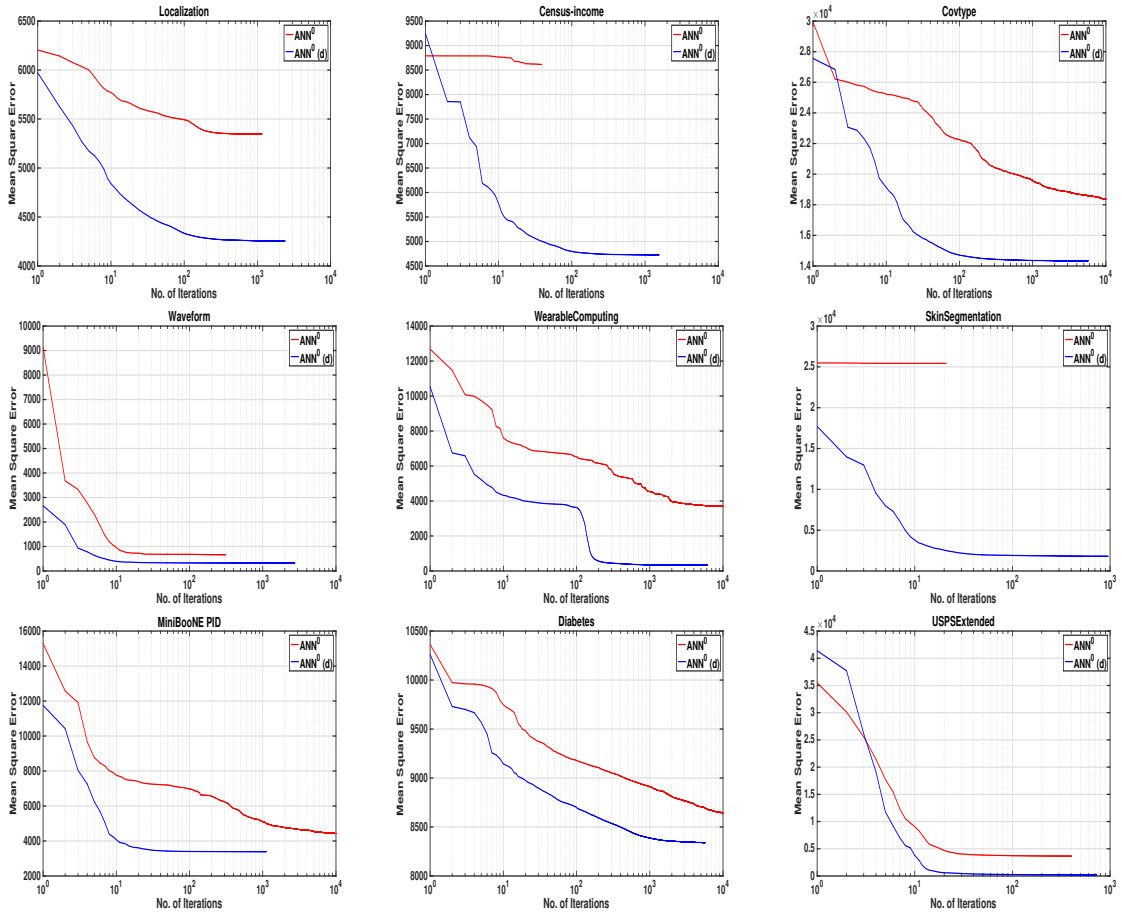


Figure 9: Comparison of rate of convergence of ANN^0 and ANN^0(d) on nine sample datasets. The X-axis is on the log scale.

tionships in the data and, therefore, can help reduce the bias of a learner that uses linear models. A linear classifier trained on discretized data is not linear any more, which has the potential to help in modeling non-linear decision boundaries which might otherwise require the use of kernels and multi-layer networks.

We show that discretization can greatly reduce the error of logistic regression and other discriminative linear classifiers optimizing Hinge Loss and Mean-square-error especially on large datasets. We compare the performance of linear classifiers trained with both qualitative and quantitative attributes (denoted as LC) with LR trained with qualitative attributes only (denoted as LC^d), where quantitative attributes were discretized first. Our empirical analysis on 52 datasets showed that LC^d led to a low-bias model and, therefore, it resulted in significantly better 0-1 Loss and RMSE performance on large datasets. Quite surprisingly, it also reduced training time and had more desirable convergence, converging more rapidly to models that better fit the data. These substantial benefits come at a cost of a minor increase in classification time.

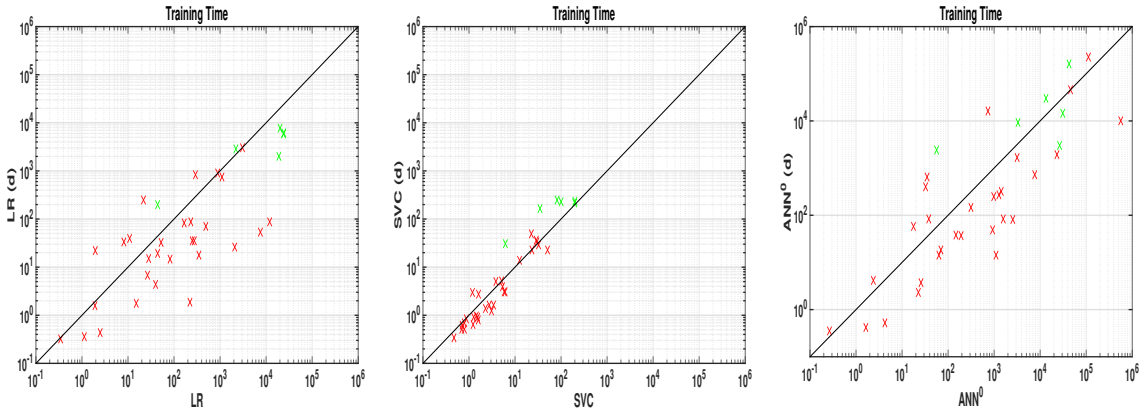


Figure 10: Comparative scatter of `Training-time` results for LC and LC$^{\mathrm{d}}$ classifiers. LC are on the X-axis whereas LC$^{\mathrm{d}}$ are on the Y-axis. For points below the diagonal line, LC$^{\mathrm{d}}$ is faster. Results on *Big* datasets are shown in green, whereas results on *Little* datasets are shown in red.

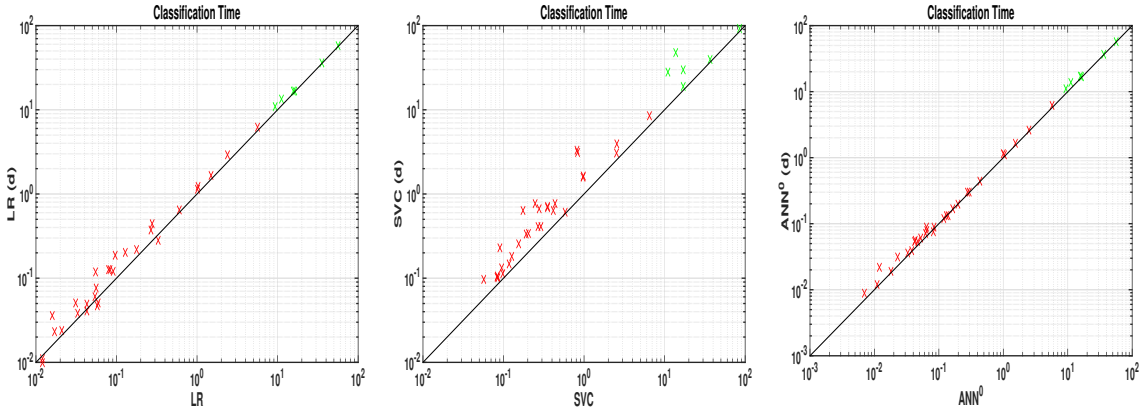


Figure 11: Comparative scatter of `Classification-time` results for LC and LC$^{\mathrm{d}}$ classifiers. LC are on the X-axis whereas LC$^{\mathrm{d}}$ are on the Y-axis. For points below the diagonal line, LC$^{\mathrm{d}}$ is faster. Results on *Big* datasets are shown in green, whereas results on *Little* datasets are shown in red.

Given the surprising gains from discretization, it is tempting to include both the original quantitative and derived discretized features in the data. Doing so avoids losing any information due to discretization. We undertook some preliminary experiments with this approach. They suggested that while it led to slight lower bias, they did not produce any improvement (in terms of error or convergence) over using only discretized-quantitative features. Further investigation of this research direction has been left as a future work.

With faster training, better convergence and low-bias we believe that discretization is worth consideration in any context where linear classifiers are learned from quantitative data.

6. Code

The details of the software library `fastLC` is given in Appendix A. The library along with running instructions can be downloaded from Github: `https://github.com/nayyarzaidi/fastLC.git`.

7. Acknowledgments

This research has been supported by the Australian Research Council (ARC) under grant DP140100087, and by the Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under contract FA2386-15-1-4007.

Appendix A. `fastLC` – Linear Classifiers Library

The library can handle both quantitative and qualitative attributes. There is no need to do a one-hot-encoding for qualitative attributes, as the LR model built can actually handle the data types.

One can execute the code in the library by issuing the following command for LR:
`>> java -cp /fastLC.jar fastLC.BVDcrossvalx -t /dataset.arff -i 2 -x 2 -W LR.LRClassifier -- -V -O "Tron"`. For SVC, use the following command:
`>> java -cp /fastLC.jar fastLC.BVDcrossvalx -t /dataset.arff -i 2 -x 2 -W SVC.SVCClassifier -- -V -O "Tron"`. For ANN^0, use the following command:
`>>java -cp /fastLC.jar fastLC.BVDcrossvalx -t /dataset.arff -i 2 -x 2 -W ANN.ANNClassifier -- -V -O "Tron"`.

Note, `-i 2 -x 2` flags specify two rounds of two-fold cross-validation. `-V` is the verbosity flag, whereas, `-O` specifies the solver. One can choose from the following list of solvers: $\{\text{GD}, \text{QN}, \text{CG}, \text{Tron}, \text{SGD}\}$, that is – gradient descent, conjugate gradient, truncated Newton and stochastic gradient descent.

For SVC, the dataset has to be binary (i.e, the number of classes are only two). For non-binary dataset use the following command:
`>> java -cp /fastLC.jar fastLC.BVDcrossvalx -t /dataset.arff -i 2 -x 2 -W onevsAllSVCclassifier -- -V -O "Tron"`.

For computing results on discretized data, either pre discretize the dataset, or use the `-D` flag to convert quantitative attributes into qualitative one by the learner.

Appendix B. Details of Datasets

Domain	Case	NomAtt	NumAtt	Class	MissVal	Description
HIGGS	11000000	0	28	2	N	This dataset is generated using Monte Carlo simulation, related to separating particle-producing collisions from a background source, including 21 kinematic properties and 7 high level attributes.

Continued on next page

Table 3 – Continued from previous page

Domain	Case	NomAtt	NumAtt	Class	MissVal	Description
HEPMASS	10500000	0	27	2	N	This dataset is generated using Monte Carlo simulation, representing information about separating particle-producing collisions from a background source, including 22 low level attributes and 5 high level attributes.
kddcup	5209460	7	34	40	N	This dataset used for the Third International Knowledge Discovery and Data Mining Tools Competition
SUSY	5000000	0	18	2	N	This dataset is generated using Monte Carlo simulation, representing information about separating particle-producing collisions from a background source, including 8 low level attributes and 10 high level attributes.
Watch_accelerometer	3540962	0	3	7	N	This dataset contains the readings of Accelerometer in smart watches. Two time related attributes are removed from the original dataset.
Watch_gyroscope	3205431	0	3	7	N	This dataset contains the readings of Gyroscope in smart watches. Two time related attributes are removed from the original dataset.
Phone_gyroscope(40%)	2786526	0	3	7	N	The dataset contains the readings of Gyroscope in smartphones. Two time related attributes are removed from the original dataset.
Phone_accelerometer(40%)	2612495	0	3	7	N	The dataset contains the readings of Accelerometer in smartphones. Two time related attributes are removed from the original dataset.
satellites(25%)	2176290	0	138	24	Y	This dataset describes satellite image time series. The classes are colours representing various land covers.
PAMAP2(25%)	962626	1	53	19	N	The dataset is generated by nine subjects wearing three colibri wireless activity sensors and one heart rate monitor. Timestamp attributes are removed from the original dataset.
MITFaceSetC	839330	0	361	2	N	Each face in MITFaceSetB is rotated between ???20° and 20°, in increments of 2°.
covertype	581012	44	10	7	N	This dataset describes information about using cartograpic variables to identify forest cover type.
MITFaceSetB	489410	0	361	2	N	This dataset is generated by making the original training face blurred and then adding to MITFaceSetA.
MITFaceSetA	474101	0	361	2	N	This dataset is composed of 477366 new non-faces and the original MIT face dataset.
USPESExtended	341462	0	675	2	N	The original training set has 1,005 zeros and 1,194 ones, while the test set has 359 zeros and 264 ones. To better study the scaling behavior, we extend this data set by first converting the resolution from 16??16 to 26??26, and then generate new images by shifting the original ones in all directions for up to five pixels.
census-income(KDD)	299285	32	9	2	N	This data set contains weighted census data extracted from the 1994 and 1995 Current Population Surveys conducted by the U.S. Census Bureau. The data contains 41 demographic and employment related variables
SkinSegmentation	245057	0	3	2	N	The skin dataset is collected by randomly sampling B,G,R values from face images of various age groups, race groups, and genders obtained from FERET database and PAL database.

Continued on next page

Table 3 – Continued from previous page

Domain	Case	NomAtt	NumAtt	Class	MissVal	Description
WearableComputing	165632	2	15	5	N	This dataset contains 5 different activities, gathered from 4 subjects wearing accelerometers mounted on their waist, left thigh, right arm, and right ankle.
localization	164860	2	3	11	N	People used for recording of the data were wearing four tags (ankle left, ankle right, belt and chest). Each instance is a localization data for one of the tags. The tag can be identified by one of the attributes.
TwitterAbsoluteSigma500	140607	0	76	2	N	The objective of this dataset is to determine whether or not these time-windows are followed by buzz events. In this dataset, each instance covers seven days of observation for a specific topic. Considering the couple day following this initial observation; If there is at least 500 additional active discussions by day then, the predicted attribute Buzz is True.
MiniBooNE_PID	130065	0	50	2	N	This dataset is used for classifying signal and background event based on 50 particle ID variables.
TVNewsChannelCommercial	129685	0	4124	2	N	This dataset is for automatic identification of commercial blocks in news videos finds a lot of applications in the domain of television broadcast analysis and monitoring.
Diabetes	101766	52	9	3	Y	The dataset represents 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. It includes over 50 features representing patient and hospital outcomes.
waveform	100000	0	20	3	N	UCI
shuttle	58000	0	9	7	N	The dataset describes shuttle status log.
adult	48842	8	6	2	N	The dataset is for predicting if a person can make over 50K a year.
letter-recog	20000	0	16	24	N	The dataset is for predicting 26 capital English alphabet letters.
magic	19020	0	10	2	N	The dataset is generated by Monte Carlo simulation, for signal and background event classification using images from a Cherenkov gamma telescope.
sign	12546	0	8	3	N	UCI
pendigits	10992	0	16	10	N	A pen-based handwritten digits dataset, collected from UCI.
pioneer	9150	7	29	57	N	UCI
satellite	6435	0	35	6	N	The dataset has records that describe different sub-areas of scenes converted from the satellite image.
optdigits	5620	0	64	10	N	This dataset is for predicting handwritten digits.
page-blocks	5473	0	10	5	N	UCI
wall-following	5456	0	24	4	N	UCI
phoneme	5438	0	7	50	N	UCI
waveform-5000	5000	0	40	3	N	The dataset is collected from UCI
spambase	4601	0	57	2	N	UCI
abalone	4177	0	8	3	N	UCI
segment	2310	0	19	7	N	UCI
mfeat-mor	2000	2	3	10	N	UCI
volcanoes	1520	0	3	4	N	UCI
yeast	1484	1	7	10	N	UCI
vowel	990	3	10	11	N	UCI

Continued on next page

Table 3 – Continued from previous page

Domain	Case	NomAtt	NumAtt	Class	MissVal	Description
vowel-context	990	1	10	11	N	UCI
vehicle	946	0	18	4	N	UCI
anneal	798	32	6	3	N	UCI
pid	768	0	8	2	N	UCI
syncon	600	0	60	6	N	UCI
musk1	476	0	166	2	N	UCI
new-thyroid	215	0	5	3	N	UCI
wine	178	0	13	3	N	UCI

References

D. Altman, B. Lausen, W. Sauerbrei, and M. Schumacher. Dangers of using "optimal" cutpoints in the evaluation of prognostic factors. *Journal of the National Cancer Institute*, 86:829–835, 1994.

D. G. Altman and P. Royston. The cost of dichotomising continuous variables. *BMJ*, 332, 2006.

Damien Brain and Geoffrey I. Webb. The need for low bias algorithms in classification learning from large data sets. In *PKDD*, pages 62–73, 2002.

L. Chih-Jen. Liblinear library, 2010. URL `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *ICML*, 1995.

U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8(1):87–102, 1992.

C. Flegal and P. Keyl, P. Nieto. Differential misclassification arising from nondifferential errors in exposure measurement. *American Journal of Epidemiology*, 34:1233–1244, 1991.

K. Y. Fung and G. R. Howe. Methodological issues in case-control studies. III: The effect of joint misclassification of risk factors and confounding factors upon estimation and power. *International Journal of Epidemiology*, 13:1366–370, 1984.

S. Garcia, J. Luengo, J. Saez, V. Lopez, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013.

S. Greenland. Dose-response and trend analysis in epidemiology: Alternatives to categorical analysis. *Epidemiology*, 6:356–365, 1995.

C. N. Hsu, H. J Huang, and T. T. Wong. Why discretization works for naive bayesian classifiers. In *International Conference on Machine Learning*, pages 309–406, 2000.

C. N. Hsu, H. J Huang, and T. T. Wong. Implications of the dirichlet assumption for discretization of continuous variables in naive bayesian classifiers. *Machine Learning*, 53(3):235–263, 2003.

J. R. Irwin and G. H. McClelland. Negative consequences of dichotomizing continuous predictor variables. *Journal of Marketing Research*, 40 (3):366–371, 2003.

R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *AAAI*, 1996.

R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *ICML*, pages 275–283, 1996.

M. Lichman. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Machine Learning*, 6(4):393–423, 2002.

J. L. Lustgarten, V. Gopalakrishnan, H. Grover, and S. Visweswaran. Improving classification performance with discretization on biomedical datasets. In *AMIA Symposium Proceedings*, 2008.

R. C. MacCallum, S. Zhang, K. J. Preacher, and D. D. Rucker. On the practice of dichotomization of quantitative variables. *Psychological Methods*, 7 (1):10–40, 2002.

R. E. Maleki, S. M. Iranmanesh, and B. M. Bidgoli. An experimental investigation of the effect of discrete attributes on the precision of classification methods. In *Information and Communication Technologies*, 2009.

T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, Department of Computer Science, 1980.

K Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

S. G. Nash. A survey of truncated newton methods. *Journal of Computational and Applied Mathematics*, 124:45–59, 2000.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

S. Reade-Christopher and L. Kupper. Effect of exposure misclassification on regression analysis. *Biometrics*, 47:535–548, 1991.

Y. Shentu and M. Xie. A note on dichotomization of continuous response variable in the presence of contamination and model specification. *Statistics in Medicine*, 29:2200–2214, 2010.

P. van der Putten and M. van Someren. A bias-variance analysis of a real world learning problem: The coil challenge 2000. *Machine Learning*, 57(1):177–195, 2004.

G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.

Y. Yang and G. I. Webb. Discretization for naive-Bayes learning: managing discretization bias and variance. *Machine Learning*, 74:39–74, 2009.