Modeling of EAD and LGD: Empirical Approaches and Technical Implementation

Yang, Bill Huajian and Tkachenko, Mykola

18 February 2012

MODELLING OF EAD AND LGD
Empirical Approaches and Technical Implementation*
(Pre-typeset version)
(Published in "Journal of Credit Risk", Vol. 8/No. 2, 2012)

Bill Huajian Yang and Mykola Tkachenko
Commercial Risk Methodology and Analytics
Bank of Montreal, Toronto, Canada

Abstract

The Basel Accords have created the need to develop and implement models for PD, LGD and EAD. Although PD is quite well researched, LGD and EAD still lag both in theoretical and practical aspects. This paper proposes some empirical approaches for EAD/LGD modelling and provides technical insights into their implementation. It is expected that modellers will be able to use the tools proposed in this paper.

Key words: Basel, EAD, LGD, WOE, Naïve Bayes, mixture density, neural network

1. Introduction

This paper proposes some practical approaches to modelling Loss Given Default (LGD) and Exposure at Default (EAD). These two measures are required by the BASEL Accords.

Probability of default (PD) modelling is supported by widely known methodologies used in Marketing, Account Management and Risk. LGD and EAD modelling are much less supported by best business practices in the modelling community. As a result, modelling methodologies for LGD and EAD are still in the developmental stages.

Several references ([1], [3]) give an overview of problems and restrictions encountered with LGD/EAD modelling. The focus of this paper is on practical techniques that will lead to feasible implementation and improvements in LGD/EAD modelling. These techniques can also be applied in other areas of predictive modelling, especially in fast-paced financial institutions.

This paper briefly surveys current modelling methodologies, then proposes some empirical approaches, and provides technical insights into their implementation. It is expected that modellers will be able to use these proposed tools for EAD/LGD modelling

---

*Bill Huajian Yang, Ph.D in mathematics, is a Senior Manager, and Mykola Tkachenko, Ph.D in physics, is Director of the Commercial Risk Methodology & Analytics team, Bank of Montreal, Toronto, Canada.

The views expressed in this article are not necessarily those of Bank of Montreal or any of its affiliates. Please direct any comments to bill.yang@bmo.com, phone 416-643-1922, and/or mykola.tkachenko@bmo.com, phone 416-927-5660.

as well as other predictive modelling. Performance comparison for all proposed tools is provided in Section 8 using our own software implementation. The authors would like to thank our colleagues Ping Wang, James Fung, and Jason Zhang for many valuable conversations, Clovis Sukam for his critical comments and proofreading of this article.

2. Overview of LGD and EAD Modelling Methodologies

2.1 LGD Modelling Approaches

Loss given default (LGD) is defined as:

$$LGD = 1 - Recovery\,Rate,$$

$$Recovery\,Rate = \frac{Amount\ \ Recovered}{Amount\ \ Outstanding\,at\,Default},$$

where amount recovered sums up all discounted cash flows received during the recovery process after default, less the total cost incurred.

There are major differences between PD and LGD modelling. While LGD is a continuous variable and usually follows a beta distribution, default events (PD) are binomial. LGD depends on the recovered amount, which may take several years after default to resolve, whereas PD describes the likelihood of a default event occurring within a specified period (usually 1 year). Information about events occurring after default has no effect on PD.

There is a lack of reliable historical data for LGD (and EAD). Interest in LGD (and EAD) data collection started in years 1996 - 2001 when specific mandatory BASEL requirements were imposed on financial institutions in order to become AIRB (advanced internal rating bands) compliant.

The non-normality of LGD (and EAD Factor) distribution calls for an explicit transformation so that the target variable follows a standard normal distribution. This will alow one to use a linear regression with a normally distributed target variable to get an LGD prediction as proposed in [3]. Although this approach allows one to build LGD models on a transformed target variable, the inverse transformation (reverting and predicting the actual target), usually exhibits large errors. This situation is due to the gradient of the inverse transformation. This gradient is usually higher in the tails, and hence, a small error with the transformed target may turn into a much bigger conversion error. This challenge suggests practical tip #1:

When linear regression is used, it is best to train the model by minimizing the error of the actual target, not the error of the transformed target.

Another problem associated with LGD models is the use of information on the collateral for the loans. One needs to appropriately integrate collateral values for different collateral

types so that the integrated collateral values can become input variables for the LGD model. The challenge with integrated collateral values is that many lenders and business people believe that there exists a linear relationship between collateral values and LGD. Since very often data suggest that LGD distribution over collateral values is non-linear, tensions can quickly build up between modellers and business people over the use of collaterals in LGD modelling. Other variables, like company size for example, can also become a bone of contention ([4]). This suggests practical tip #2:

Weight of Evidence (WOE) technique comes in useful when trying to incorporate business beliefs and judgements while maintaining the statistical strengths of the model.

2.2. EAD Modelling Approaches

EAD modelling approaches are illustrated with EAD-factor (Credit Conversion Factor) examples.

Denote

Bal_0 - the facility outstanding dollar amount at current time

Bal_1 - the facility outstanding dollar amount at default time

$Auth_0$ - the facility authorized dollar amount at current time

$Undrawn_0 = (Auth_0 - Bal_0)$ - the facility undrawn dollar amount at current time

Define EAD Factor (also called Credit Conversion Factor) as follows:

$$EAD\ Factor = \begin{cases} \dfrac{Max(Bal_1 - Bal_0, 0)}{Undrawn_0}, \ if\ \ undrawn_0 > 0 \\ 0, \ \ if\ undrawn_0 \le 0 \end{cases}$$

Thus EAD Factor is the proportion of $undrawn_0$ to drawn down at default time. The predicted exposure amount at default is then calculated as:

$$EAD = Bal_0 + EAD\ \ Factor \times Undrawn_0$$

The Max function applies in the definition, as Basel requires one to model the risk of further drawing down at time of default. Another practical option for the EAD Factor definition is to remove the Max function and model the EAD Factor between -1 and 1 (floored at -1 and capped at 1), then floor the prediction at 0. Modelling the EAD Factor this way captures the two-directional spending behaviour of drawing more and paying down as well. It is worthwhile to mention that modeling the EAD factor within a range between -1 and +1 and flooring the resulted EAD prediction at 0 could lead to underestimated EAD values.

Another option in selecting a target variable in Exposure at Default modelling is to model the facility utilization change, which is defined as:

$$Util\_Ch = \frac{Bal_1 - Bal_0}{Auth_0}$$

floored at 0 and capped at 1. Both the EAD Factor and Utilization Change model the outstanding dollar amount change with the first as a fraction of undrawn amount ($Undrawn_0$) and the latter as a fraction of current authorized limit ($Auth_0$).The EAD Factor is usually more difficult to model as the undrawn dollar amount for some facilities could be very small thus inflating the EAD Factor. Both types of models are good when converting back to predict the EAD dollar amount (or as fraction of current authorized limit $Auth_0$).

Possible other target variables for an EAD model include utilization ratio or the EAD dollar amount. Please refer to ([10]) for a review of possible target variables for EAD models.

We recommend modelling the EAD Factor or Utilization Change for the following reasons:

(a) Basel requires one to floor the estimated exposure at current outstanding, which means one needs only to focus on the change of exposure.
(b) Both are ratio variables, dimensionless (unlike dollars and cents which have a unit), thus will not be impacted by the magnitude of scale, and is within a narrow range between 0 and 1.

In following discussion, we choose to model the EAD Factor floored at 0 and capped at 1, rather than modelling the EAD dollar amount directly. As the outstanding amount at default (Bal_1) varies significantly from a very low dollar amount to an extremely high dollar amount, modelling the EAD dollar amount directly would be statistically difficult.

It turns out that even though an EAD Factor model such as the Logit model shown in Section 8 may have a low R squared (RSQ) .In the following example, the RSQ is only 0.27,therefore it can translate into a much higher RSQ (like 0.91 below) when converting to predict the EAD dollar amount using the above formula. This suggests practical tip #3:

Choose to model the EAD Factor or Utilization Change rather than the outstanding dollar amount at default.

By the definition of EAD Factor, one needs to divide the sample into two segments before modelling; one with $undrawn_0 > 0$, the other with $undrawn_0 \le 0$. We need to model the EAD Factor for those with $undrawn_0 > 0$ only. This suggests practical tip #4:

Model the EAD Factor only for those facilities with $undrawn_0 > 0$**.**

Different financial institutions, especially wholesale lending portfolios, face some common problems with EAD modelling. As an example, since the EAD Factor distribution usually exhibits higher concentration around 0 and 1 after flooring at 0 and capping at 1, a linear regression model will predict only a narrow range around the target variable average. Some other problems include, having a small model development sample size, and small pool of candidates for covariate selection, especially for companies that are not publicly traded, like wholesale portfolios.

Macroeconomic factors, even if not explicitly included in Basel models, inevitably influence the risk profile of the obligors and facilities associated with the obligor. Inclusion of macroeconomic factors in the model however, can significantly shift the model prediction from what is required by the Basel Accord. Namely, prediction could violate the Basel Accord expectation that EAD/LGD cannot be less than the long run default weighted average of the EAD/LGD risk factors. With macroeconomic variables in the model, model prediction closely follows current conditions of the obligor regardless of the aggregated profiling parameters of a complete economic cycle. This model falls under the category of Point in Time models (PIT) and is considered not sufficiently conservative by many regulators. A Through the Cycle (TTC) model that is supposed to produce predictions throughout the economic cycle, is considered to be the most conservative risk assessment model. It should include neither direct macroeconomic variables nor any variable somewhat correlated with the economic state of being. Obviously, the latter is impossible to control, but such a model could have a significant advantage from the point of view of some regulators in a sense that capital, calculated based on a TTC model, shows essential stability over time. The downside of TTC models is that they cause complete economic cyclical insensitivity, and can negatively influence overall economic development due to stable but low credit lending activity.

It turns out that macroeconomic effects impact the EAD Factor in the following way: when business is booming for more than one year, or when there is a downturn for more than one year, the EAD Factor starts to climb during the 2nd year. For this reason, we prefer not to include any macro variable in the EAD Factor model. This suggests practical tip #5:

Choose to model the EAD Factor or Utilization Change model with no macro variables.

3. Proposed Parameter Estimation Approaches

3.1. Variable Transformation. Leaving selection of the target variable to the preference of the practitioner, the first proposed innovation for LGD/EAD modelling is the technique of Weight of Evidence (WOE) transformation for independent variables. It applies to both types of variables; numeric and character. As will be shown later, such variable transformation allows one to tackle problems with optimum selection of variables, issues with outliers, as well as problems with imputation of missing values.

WOE methodology is quite well known for Risk models, but surprisingly, it is not widely used for Basel specific models. As previously mentioned in 2.1, the WOE approach could accommodate business judgement as well.

3.2 Variable Selection. With all independent variables WOE transformed, a Naïve Bayesian (see Section 5) model can be employed for variable selection. Each time, it selects one variable that gives the highest lift to the Naïve Bayesian model. Naïve Bayesian selection methodology is better than using the SAS stepwise selection procedure for logistic or linear regression, which also includes variables with negative coefficients. With WOE transformed variables, a negative coefficient in general indicates noise effect, and will most likely not be accepted by business partners.

3.3 Model Structure. We consider the following models, which are either trained by maximizing likelihood or minimizing the least square error:

1. High/Low models by using SAS logistic procedure
2. Logit model trained using SAS logistic procedure with events/trials syntax
3. Logit model trained by minimizing the least square error
4. Naïve Bayesian model with no parameter training
5. Mixture model by minimizing least square error
6. Single layer neural network by minimizing least square error

As both LGD and EAD Factor distributions usually exhibit high concentration around 0 and 1, the mixture model or single layer neural network demonstrate significant improvement over the logit model that uses only raw variables (with no WOE transformation). Even a simple model like the Naïve Bayesian model sees a decent improvement (see Section 8).

3.4 Boost Model. Boost modelling is discussed in Section 7.

4. Nearest-Neighbour Method and WOE Transformation

Due to the significant importance of WOE (weight of evidence) variable transformation as a vital component in further analysis, we now review the theory behind the WOE technique.

The concept of weight of evidence (WOE) is close to the k nearest-neighbour algorithm in machine learning. Its importance can further be demonstrated by the Naïve Bayesian model as described in Section 5.

Given a sample $D = \{(x, y)\}$, where y is the dependent variable, and $x = (x_1, x_2, ..., x_m)$ is a vector of independent variables, denote $E(y \mid x)$ the regression function, that is, the expected value of y given x.

For a fixed $k > 1$, the k-nearest neighbour method estimates $E(y \mid x)$ by taking the average of k values of $y : y_1, y_2, ..., y_k$, corresponding to k nearest points relative to the current point x (not including the current point x):

$$y \sim (y_1 + y_2 + ... + y_k)/k$$

The resulting algorithm is called the k-NN algorithm in machine learning, which is a type of instance-based learning ([5] pp.14-16, pp.165-168, [13]).

A special case when vector x consists of only a single variable leads to the following concepts of WOE transformation:

4.1 WOE Transformation

Consider the linear regression model:

$$y \sim a + bx,$$

where x is a vector of independent variables, a is the intercept, and b is the vector of parameters.

The WOE transformation for a numerical variable z consists of the following steps:

(a) Partition the variable z into intervals.
(b) Calculate the average of y for each interval.
(c) Derive a new variable $w(z)$ by assigning the average from (b) when the value of the variable z falls in that interval.

We call the derived variable $w(z)$ the WOE transformation for the variable z. The idea with WOE transformation is that it gets the model right on the individual variable level first, regardless of what we do subsequently.

For a class variable, the WOE transformation can be implemented by first calculating the average of y for each class level, then grouping based on business and statistical considerations.

Step (b) in general requires adaption when the chosen model structure is not linear. Below are two practical examples.

4.2 Two Practical Examples

A. Binary dependent variable ($y = 0,1$)

Assume that we choose the logit model:

$$\log(p/(1-p)) \sim b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_m x_m$$

where p and $1-p$ are the conditional probabilities given by:

$$p = P(y=1 \mid x),$$
$$1-p = P(y=0 \mid x) \ .$$

By Bayes theorem, we have:

$$\frac{p}{1-p} = \frac{P(x \mid y=1)P(y=1)}{P(x \mid y=0)P(y=0)},$$

$$\log\frac{p}{1-p} = \log\frac{P(x \mid y=1)}{P(x \mid y=0)} + \log\frac{P(y=1)}{P(y=0)}.$$

The second term $\log\dfrac{P(y=1)}{P(y=0)}$ is just the population log odd, which does not depend on x. While the first term (when x reduces to a single variable) $\log\dfrac{P(x \mid y=1)}{P(x \mid y=0)}$ suggests the following adaption to (b) of Section 4.1:

(b1) Divide the sample into Bad ($y=1$) and Good ($y=0$) segments. Calculate the percentage distribution of Bad over the partitioned intervals (denoted by bdist), and the percentage distribution of Good over the partitioned intervals (denoted by gdist). Compute the WOE value for an interval as:

$$WOE = \log\frac{bdist}{gdist} \ .$$

B. Continuous Dependent Variable ($0 < y < 1$)

Again, assume that we choose the logit model:

$$\log(y/(1-y)) \sim b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_m x_m \ ,$$

or that the logit model is a model component, as with the mixture model or single layer neural network discussed in Section 6.

Notice that y can be estimated by the regression function $E(y \mid x)$, which is estimated by avg(y), the average of y values for a given x. When x reduces to a single variable, this suggests the following adaption to (b) of Section 4.1:

(b2) Calculate the average of y values over a partitioned interval and denote it by avg(y). Compute the WOE value for that interval as:

WOE = log (avg(y)/(1-avg(y))

4.3. Technical Implementation

An automation of the WOE transformation should encompass the following:

(i) Controlling the number of intervals and the size of an interval.
(ii) Choosing the interval partition that maximizes the variable predictive power.
(iii) Isolating special values of the variable.

Such application should also provide other functionalities including: printing the SAS code for the WOE transformation, allowing interval breaking and regrouping, calculating the predictive power for the transformed variable, and outputting the bucketing for reporting and documentation.

5. Naïve Bayesian Models

A Naïve Bayesian model calculates the conditional probability under some assumptions of independence. We show in this section how the WOE technique contributes interestingly to the implementation of Naïve Bayesian models.

Suppose y is a binary dependent variable. As before, let $x = (x_1, x_2, ..., x_m)$ denote a list of independent variables.

Under the Naïve Bayes assumptions, variables $x_1, x_2, ..., x_m$ are independent conditional on $y = 1$ and $y = 0$, respectively.

5.1 Naïve Bayesian Models

Under the Naïve Bayes assumptions, we have by Bayes theorem:

$$\frac{P(y=1 \mid x)}{P(y=0 \mid x)} = \frac{P(x \mid y=1)P(y=1)}{P(x \mid y=0)P(y=0)}$$

$$= \frac{P(x_1 \mid y=1)P(x_2 \mid y=1)...P(x_m \mid y=1)P(y=1)}{P(x_1 \mid y=0)P(x_2 \mid y=0)...P(x_m \mid y=0)P(y=0)} \quad .$$

Taking the natural logarithm, we have

$$\log(p/(1-p)) = \log\frac{P(y=1)}{P(y=0)} + \log\frac{P(x_1 \mid y=1)}{P(x_1 \mid y=0)} + \log\frac{P(x_2 \mid y=1)}{P(x_2 \mid y=0)} + ... + \log\frac{P(x_m \mid y=1)}{P(x_m \mid y=0)}$$
$$= w_0 + w(x_1) + w(x_2) + ... + w(x_m),$$

where

$$w_0 = \log\frac{P(y=1)}{P(y=0)},$$
$$w(x_i) = \log\frac{P(x_i \mid y=1)}{P(x_i \mid y=0)}.$$

The constant w_0 is just the population log odd as seen before. But $w(x_i)$ is something interesting, as it can be estimated by the WOE transformation described in adaption (b1) of Section 4.2.

This means that the Naïve Bayesian model can be interpreted simply as a model with log odd score being given by summing up the WOE transformed variables and the population log odd:

$$\log odd\ score = \log(p/(1-p)) \quad (1)$$
$$= w_0 + w(x_1) + w(x_2) + ... + w(x_m) \quad (2)$$
$$p = \frac{1}{1+\exp(-\log odd\ score)} \quad (3)$$

The Naïve Bayesian model is a particularly simple model in structure, but is very robust in performance for ranking order, and no training of the parameters is required.

Although the above discussion is specific to the case when the dependent variable *y* is binary, it can be extended to the case when y is continuous with $0 < y < 1$. The only difference is that we need to calculate WOE by adapting (b2) of Section 4.2 and change $\log(p/(1-p))$ in (1) to $\log(y/(1-y))$.

5.2 Technical Implementation

To implement the Naïve Bayesian model for either a binary dependent variable y, or a continuous dependent variable 0<y<1, such as the EAD Factor and LGD, we take the following steps:

(a) Apply the WOE transformation appropriately (either adapt (b1) or (b2) of Section 4.2) to independent variables
(b) Use stepwise selection of variables that add incremental performance lift
(c) Output the model in the forms of (1)-(3)

Although the Naïve Bayesian model rank orders well, one may run into a problem when the magnitude of the predicted value matters. This is because Naïve Bayes assumptions are generally not strictly satisfied ([8], [14]). Under this circumstance, a segmentation step can be applied:

> (d) create score bands for the built Naïve Bayesian model and map a score band to the average of y values over that band

Another solution is to apply the boost methodologies as described in Section 7 to correct and adjust the prediction bias.

6. Modelling a Continuous Dependent Variable

In this section, we assume that y is a continuous dependent variable with values $0 < y < 1$ (with EAD Factor or LGD, one can always cap or floor or scale the dependent variable to the 0 - 1 range).

We propose in this section a few methodologies in addition to the Naïve Bayesian model discussed previously. We will compare their performance in Section 8.

6.1 High and Low Binary Modelling

With this methodology, we first define two binary variables:

$$H = \begin{cases} 1 \ if \ y > a, \\ 0 \ else \end{cases}$$

$$L = \begin{cases} 1 \ if \ y > b, \\ 0 \ else \end{cases}$$

where constants a and b can be chosen appropriately depending on the distribution of the dependent variable y. For example, we may choose $a = 0.75$ and $b = 0.05$. In general, the sizes of segments $H = 1$ and $L = 1$ should not be too small to pick up the high and low trends of y values respectively.

We then build a probability model p_1 to predict $H = 1$, and a probability model p_2 to predict $L = 1$. Reverse the probability p_2 by taking $1 - p_2$, and define an enhanced composite score as:

$$p = (p_1 + p_2)/2$$

Typically, the composite score p exhibits stronger ranking power than either p_1 or p_2 individually. Keep in mind that as long as a model rank orders well, bias in

magnitude can be corrected by the segmentation methodology as mentioned in Section 5.2 (d).

6.2. Logit Model

With this model, we assume:

$$\log(y/(1-y)) \sim b_0 + b_1 w(x_1) + b_2 w(x_2) + ... + b_m w(x_m) \qquad (4)$$

where $w(x_i)$ is the WOE transformation described in Section 4.2 for a continuous dependent variable y at a given x_i.

Parameters $b_0, b_1, ..., b_m$ can be obtained by either

(a) maximizing the negative log likelihood function, or
(b) minimizing the least square error of the model prediction.

For EAD Factor and LGD modelling, the least square minimization (b) is generally preferred. In this case, we label this least square logit model as LS Logit. We will discuss the LS Logit models in Section 6.3.

Training the logit model by maximum likelihood is implemented in SAS logistic procedure. One can use the events/trials syntax for SAS logistic procedure to get the parameter estimates (notice that on the left side of the equal sign in the model statement, it is " $y/1$ " instead of " y "):

```
proc logistic data=data;
      model   y/1 = w(x1) w(x2) ... w(xm) ;
run;
```

Alternatively, one can use the regular SAS logistic procedure for a binary dependent variable to train a model predicting y, following the steps below:

(a) augment the original sample $D = \{(x, y)\}$ in such a way: for each record in D duplicate the same record 100 times (or 1000 times if higher precision is required). For example, if f_0 stands for a record for a facility in the sample D, then inside the augmented dataset there are *100* duplicate records for f_0.

(b) Define a binary dependent variable H for the augmented sample as follows: suppose for record f_0 in D the y value rounds up to $k/100$, where $0 \le k \le 100$ is an integer. Among these 100 duplicate records for f_0 in the

augmented sample, assign k of them to have $H = 1$, and the rest $100 - k$ to have $H = 0$.

The frequency of event $H = 1$ for the given facility f_0 is k (out of 100). We thus have transformed the target variable y into the probability of the event $H = 1$ for the given facility f_0 with the augmented sample. One can then apply the WOE transformation for a binary target (as in Section 4.2) to independent variables, and train the probability model predicting the probability of $H = 1$ using the regular SAS binary logistic regression.

This methodology is essentially the same as using SAS logistic events/trials syntax if one scales up y by 100 (that is, replace y by $y \times 100$) and change "y/1" to "y/100", as in the following:

```
proc logistic data=data;
  model y/100 = w(x1)  w(x2)  ...  w(xm);
run;
```

Although both methodologies result in essentially the same parameter estimates (except possibly the intercept, due to the fact that the binary WOE transformation differs from the continuous version of WOE transformation by a constant), scaling up y by 100 usually decreases the p value for the significance of a variable to be included in the model.

6.3 Least Square Logit (LS Logit) Models

This is a model where model (4) is trained by minimizing the total least square error:

$$\sum (y - p)^2$$

where

$$p = \frac{1}{1 + \exp(-\log odd\ score)},$$

$$\begin{aligned} \log odd\ score &= \log(p/(1-p)) \\ &= b_0 + b_1 w(x_1) + b_2 w(x_2) + ... + b_m w(x_m) \end{aligned}$$

Here $w(x_i)$ is the WOE transformation for variable x_i as in Section 6.2.

Usually, for generalized linear models, including the logit model, it is assumed that the error term should follow a distribution belonging to exponential family ([9]), which is not normal in the case of logit model. As maximizing likelihood is not necessarily equivalent to minimizing the total least square error (unless the error term is normal), general logit model differs in general from the least square logit model. We do not assume that the

error term for least logit model follows a normal distribution, as in the case of Single Index model described in [2].

Technical Implementation

Training the LS Logit model can be implemented following the Iteratively Re-Weighted Least Square algorithm as described in ([5] p349). This algorithm applies to the training of models of the form:

$$y \sim h(b_0 + b_1 w(x_1) + b_2 w(x_2) + ... + b_m w(x_m)),$$

where h is a given differentiable function.

We mention here that with SAS logistic procedure, the maximum likelihood estimation of parameters is also implemented using the Iteratively Re-Weighted Least Square algorithm ([12] or [6]). However, the weight assigned for each iteration differs from what is assigned here for an LS Logit model. This is because their objective functions for optimization are different.

6.4 Mixture Models and Single Layer Neural Networks

When capped at 1 and floored at 0, the distribution of the EAD Factor or LGD exhibits heavy concentration around 0 and 1, as mentioned previously. Recall the mixture model for cluster analysis for unsupervised learning ([15]), where we model probability density by assembling the component densities from individual clusters (density $f_i(x, b_i)$ for *i-th* cluster) as in the following:

$$f(x,b) \sim p_1 f_1(x, b_1) + p_2 f_2(x, b_2) + ... + p_m f_m(x, b_m).$$

Here, $b = \{b_1, b_2, ..., b_m\}$ are parameters to be determined, and p_i is the prior probability of falling into the *i-th* cluster. This suggests an adapted mixture model (for our supervised learning) of two components: one component that models the low y value (y is the LGD or EAD Factor in our case) cluster and another that models the high y value cluster.

Adding another component to address the median y values usually improves the prediction accuracy. This middle component serves as a correction component between the high and low y value clusters.

This suggests a 3-component mixture model.

6.4.1 3-Component Mixture Models and Single Layer Neural Networks

A 3-component mixture model is defined as:

$$\begin{aligned} y \sim\ & p_1 h(b_{10} + b_{11} w(x_1) + b_{12} w(x_2) + \ldots + b_{1m} w(x_m)) \\ & + p_2 h(b_{20} + b_{21} w(x_1) + b_{22} w(x_2) + \ldots + b_{2m} w(x_m)) \\ & + p_3 h(b_{30} + b_{31} w(x_1) + b_{32} w(x_2) + \ldots + b_{3m} w(x_m)) \end{aligned} \qquad (5)$$

where $w(x_i)$ is the WOE transformation for variable x_i as in Section 6.2, and

$$h(z) = \frac{1}{1 + \exp(-z)}.$$

Here parameters p_1, p_2, p_3 are nonnegative numbers just like the prior probabilities for the mixture models for the cluster analysis, satisfying

$$p_1 + p_2 + p_3 = 1. \qquad (6)$$

When the constraint (6) is removed, the model is called a 3-component single layer neural network. In either case, the parameters can be trained by minimizing the total least square error of the prediction.

Compared to the Logit or LS Logit model, a mixture model or single layer neural network usually demonstrates strong predictive power and high accuracy even with just a minimal increase of the number of parameters.

6.4.2 Technical Implementation

The above mixture models and single layer neural networks can be implemented and trained by using the conjugate gradient algorithm and the Fletcher-Reeves method as described in ([7] pp121-133, [16] pp424-429, [5] p355): first, find a decreasing trend for the total least square error, then perform a line search along this direction to ensure that a sufficient decrease has been found with the total least square error.

Just as with the cluster analysis, the EM (Expectation Maximization)-algorithm ([15], lecture 16) applies, where for the M-step the above minimization process applies for training parameters $\{b_{ij}\}$, and for the E-step the above minimization process applies for training parameters $\{p_i\}$. Both steps iterate until convergence.

With the WOE technique, one can knock out a variable during the training process when the corresponding coefficient becomes negative. As mentioned earlier, negative coefficients usually indicate noise effect and are rarely accepted by business partners.

7. Boost Modelling

A situation can arise where we are not satisfied with the model built, probably because of its bias in prediction for some score bands. In addition to the segmentation methodology mentioned in Section 5.2 (d), we can use the following boost strategies to improve the accuracy:

(a) boosting the model by a decision tree
(b) boosting the model by a scalar model
(c) boosting the model by a linear regression

With methodology (a), we simply choose the model prediction error as the new dependent variable and train a decision tree using all available variables (no WOE transformation is required), including the built model score. In general, we get a decent improvement in accuracy.

We now focus on methodologies (b) and (c).

7.1. Boosting by a Scalar Model

Let *ptrend* denote the value of y predicted by the model built (base model). A scalar model can be trained by scaling an exponential factor to the base model:

$$y \sim \exp(a_0 + a_1 z_1 + a_2 z_2 + ... + a_m z_k) \times ptrend \ ,$$

where $z_1, z_2, ..., z_k$ are either indicator variables denoting the score bands that require adjustment for the prediction error, or other available variables that give lift to this scalar model. The resulting boost model needs to be capped at 1 (because 0<y<1).

Scalar boost models can be implemented and trained similarly by using the conjugate gradient algorithm and the Fletcher-Reeves method as described in ([7] pp121-133, [16] pp424-429) through the minimization of the total least square error.

7.2. Boosting by a Linear Regression

In this section, we discuss how linear regression can be used to improve the base model prediction.

First, we transform all independent variables, including the built base model score *ptrend* , into WOE format following (a)-(c) in Section 4.1. At this point, if we regress y on these WOE transformed variables, we usually end up with just one variable in the model; the base model score *ptrend* . This is because the base model has already absorbed all the possible available information.

Therefore, more diligence is required as suggested in (a) and (b):

(a) generate new predictive variables
(b) segment the sample into high and low segments using *ptrend* .

For example, we can divide the sample into high/low segments using indicators H and L, where:

$$H = \begin{cases} 1 \; if \;\; ptrend > c, \\ 0 \; else \end{cases}$$
$$L = 1 - H$$

where number *c* can be chosen to be the mean or mode of variable y.

We can then fit a linear sub-model to each segment, individually forcing the WOE transformed *ptrend* variable to be included in the new models (as the base score). Let p_1 and p_2 be the sub-model scores for H and L segments respectively, both capped at 1. Then the final model can be given by:

$$p = H \times p_1 + L \times p_2 \;\; .$$

To generate new predictive variables, we consider variables of the form:

$$ptrend \times w(z), \tag{7}$$

where $w(z)$ is the WOE transformed variable for variable z following (a)-(c) in Section 4.1. Usually, these types of variables exhibit higher predictive power.

Applying the Bayes theorem, we can show that a boost variable of the form (7) usually carries more information than $w(z)$ or *ptrend* individually, provided that variable z had not been included in the base model.

Let B be a binary variable. Assume independence between x and z, and independence between $x \mid B$ and $z \mid B$.

Then we have:

$$P(x, z) = P(x)P(z),$$
$$P(x, z \mid B) = P(x \mid B)P(z \mid B)$$

Thus,

$$P(B \mid x, z) = \frac{P(B, x, z)}{P(x, z)}$$

$$= \frac{P(B)\,P(x, z \mid B)}{P(x, z)}$$

$$= \frac{P(B)P(x \mid B)P(z \mid B)}{P(x, z)}$$

$$= \frac{P(x)P(z)P(B \mid x)P(B \mid z)}{P(x, z)P(B)}$$

$$= \frac{P(B \mid x)P(B \mid z)}{P(B)}$$

This means $P(B \mid x) * P(B \mid z)$ differs from $P(B \mid x, z)$ only by a constant $1/P(B)$. Because $P(B \mid x, z)$ carries more information than $P(B \mid x)$ *or* $P(B \mid z)$ individually, we conclude that $ptrend \times w(z)$, which is an estimate for $P(B \mid x) * P(B \mid z)$, is more likely to be more predictive than *ptrend* and $w(z)$ individually.

8. Model Performance Comparison

In this section we present the model performance results. The following models were trained using either SAS procedures or our own software implementation:

Logit Raw - Logistic model by SAS using variables with no WOE transformation
HL Logit - Combination of high and low logit models by SAS using WOE transformed variables
Logit - Logistic model by SAS using WOE transformed variables
LS Logit - Least Square Logit model by our own software implementation using WOE transformed variables
Naïve Bayes -Naïve Bayesian model by our own software implementation using WOE transformed variables
Mixture - 3-component mixture model by our own software implementation using WOE transformed variables
Neural Net - 3-component single layer neural network by our own software implementation using WOE transformed variables

We focused on EAD Factor modelling and similar results were obtained for the LGD case.

We first applied the WOE transformation to all independent variables following Section 4.2. example B (WOE transformations for the following 8 variables were verified and confirmed with business partners), and selected only those variables, using the Naïve Bayesian model selection methodology, that demonstrated significant incremental

performance lift. Below is a list of variables selected from this stepwise selection procedure:

1. Borrower level utilization
2. Facility level primary collateral value percentage
3. Facility authorized amount
4. Ratio of limit increase to undrawn
5. Total assets value
6. Industry code
7. Facility level utilization
8. Total facility collateral value percentage

As we are working on facility level EAD (not borrower level), both borrower and facility level utilization and collateral percentage are relevant. Authorized amount acts as a potential exposure, and total assets value measures the size of the entity. We know both entity size and industry segment are important risk drivers. We did not include any macroeconomic variables.

For the above 7 models, we used the same 8 variables. Except for the Logit Raw, which uses the raw form of these 8 variables, all other models were trained using the same 8 WOE transformed variables. We dropped a variable unless its corresponding coefficient in the model was positive (according to our interpretation of WOE definition). This was done essentially to avoid the potential risk of including noise effect in the model as mentioned before, making the model simpler and robust.

The table below shows the summary performance statistics for each of the listed 7 models (MAD=mean absolute deviation (error), RMSE=root mean square error, RSQ=R-square, KSD= Kolmogorov–Smirnov statistic):

(Inset Table 8.1 here)

Here, the number of parameters of HL Logit is 18 (last column) because each High and Low model uses 9 parameters including the intercept. For the Mixture model we would have used 29 parameters ($=3\times 9+2$) but we used only 17 after dropping those variables with negative parameter values. Similarly, for Neural Net we would have used 30 parameters ($=3\times 9+3$), but we used only 17 after dropping those variables with negative parameter values (dropped one more than that for the Mixture model).

From this table, we see that all other 6 models are significantly better than the Logit Raw model. This shows the value of the WOE transformation. Even for a simple model like the Naïve Bayesian model WOE provided decent improvements. It turns out that Mixture model or Neural Net of 3-components is particularly a good candidate for EAD Factor or LGD modelling.

Performance usually lifts up significantly when the predicted EAD Factor is translated into a prediction of an EAD dollar amount or the EAD dollar amount as a fraction of the current authorized limit using the following formulas:

$$EAD = Bal_0 + EAD\ Factor \times Undrawn_0,$$
$$Util_1 = \frac{Bal_0 + EAD\ Factor \times Undrawn_0}{Auth_0}.$$

For example, for the LS Logit model, we have $RSQ = 0.27$. But when it is converted into a prediction of the EAD dollar amount, we have $RSQ = 0.91$.

Finally, we present the performance results for the boost methodologies, boosting the LS Logit model built previously (the base model):

Scalar - Boosting by a scalar
Linear Reg – Boosting by a linear regression

For the Scalar boost, we cut the base model score *ptrend* into 8 score bands, using a decision tree software, and trained a scalar boost model as described in Section 7.1 using our own software implementation.

For the Linear Reg, we divided the sample into H and L segments using the base score *ptrend* , as described in Section 7.2, with the cutting point given by:

$$c = average\ \ y\ \ over\ the\ sample$$

The table below shows the summary performance statistics:

(Insert Table 8.2 here)

Performances for both models have slightly improved compared to the base model.

The above results are based on a sample of 500 commercial borrowers, which is relatively small compared to the retail case, where we usually have a much larger sample, depending on products we are working with. With retail EAD and LGD, industry code is replaced by product type, while collateral percentage and total assets value are simply not there in the data for retail revolving products. Utilization, credit bureau report, recent delinquency records, and the activeness of looking for more credits, are among the important drivers. With large retail samples, the risk patterns fitted from the sample for variable bins, through WOE transformation, are usually much stable, resulting in better model performance eventually.

REFERENCES

[1] Cris Marrison, The Fundamentals of Risk Measurement. McGrow-Hill, 2002

[2] Gery Geenens, Michel Delecroix, A Survey about Single-Index Models Theory, International Journal of Statistics and Systems, ISSN 0973-2675 Vol.1 No.2 (2006), pp. 203-230

[3] Greg M. Gupton, Roger M. Stein, LossCalc: Model for Predicting Loss Given Default (LGD). February 2002, Moody's KMV

[4] Jens Grunet, Martin Weber. Recovery Rates of Commercial Lending: Empirical Evidence for German Companies, SSRN, 2007

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, The Elements of Statistical Learning, 2nd Edition, Springer

[6] John Fox, Programming Exercise: Binomial Logistic Regression by Iteratively Weighted Least Square, An R and S-Plus Companion to Applied Regression

[7] Jorge Nocedal, Stephen J. Wright, Numerical Optimization, 2nd Edition, Springer

[8] Kim Larsen, Generalized Naive Bayes Classifiers, Sigkdd Exploration, Vol 7, Issue1, pp76-81

[9] McGullagh, P. and J.A. Nedler, Generalized Linear Models, Second Edition, London: Champan and Hall, 1989

[10] Mechial Jacobs, An Empirical Study of Exposure at Default. Risk Analysis, Division / Credit Risk Modelling Moody's KMV Credit Practitioner's Conference. September 9, 2009.

[11] Radovan Chalupka and Juraj Kopecsni. Modelling Bank Loan LGD of Corporate and SME Segments A Case Study. Ideas.Repec.Org, 2009

[12] SAS/Stat User's Guide, Vol 2, 4th Edition, pp1088

[13] T. M. Cover and P. E. Hart, Nearest Neighbour Pattern Classification. IEEE Transactions on Information Theory (1967) vol. 13 (1) pp. 21-27

[14] Tom M. Mitchell, Machine Learning, 2nd Edition

[15] Tommi Jaakkola, Lecture Notes, 6.867 Machine Learning, Fall 2002, MIT, Open Course Ware

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brain P. Flannery, Numerical Recipes in C++, Cambridge University Press, 2nd Edition, 2003