Paper 3018-2019

Predicting Inside the Dead Zone of Complete Separation in Logistic Regression

Robert Derr, SAS Institute Inc., Cary, NC

ABSTRACT

Logistic regression modeling can result in perfect prediction of the responses in your training data, which unfortunately can correspond in practice to very imperfect prediction of the responses in a new data set. This imperfect prediction is often due to complete separation of your data, which means that there is a way to fit the model so that all the events have high predicted probabilities and all the nonevents have low predicted probabilities. In particular, there is a region that separates your event data from your nonevent data and that contains no observations. This "dead zone" has no information about your model, and hence you cannot evaluate the correctness of any predictions inside this region. This paper illustrates the dead zone and provides methods to find and measure it.

INTRODUCTION

Increasingly complex programs for machine learning and predictive analysis often include fitting a logistic regression model as one of their steps. Logistic regression is a generalized linear model that you can use to predict which of two response levels is more likely for a given observation. A logistic regression program can fail for a number of reasons, but even when it does not fail, other problems can appear. This paper focuses on one of those problems: complete separation and the existence of a dead zone in your data. Proceeding with analyses under complete separation is not necessarily fatal; the results can be successfully used as a practical tool, but you should handle them appropriately.

First, a brief review of logistic regression. You use a logistic regression model when you have a response variable y with two (or more) possible levels; call them the events and the nonevents. A logistic regression models the probability that an observation that contains explanatory variables $\mathbf{x}$ is an event by using a linear function of the predictors:

$$\text{logit}(\Pr(y = \text{event})) = \log\left(\frac{\Pr(y = \text{event})}{1 - \Pr(y = \text{event})}\right) = \mathbf{x}'\boldsymbol{\beta}$$

Assuming that your data are a simple random sample from a binomial distribution, this model is fit by using an iterative algorithm to maximize the likelihood function. The maximum likelihood estimates for the parameters of this fitted model are denoted by $\hat{\boldsymbol{\beta}}$, and the model-predicted event probability for an observation with predictors $\mathbf{x}$ is denoted by $\hat{\pi} = \hat{\pi}(\mathbf{x}) = \Pr(y = \text{event}|\mathbf{x}, \hat{\boldsymbol{\beta}})$.

Albert and Anderson (1984) investigate the existence of these maximum likelihood estimates and divide data sets into the following three categories:

- *complete separation*: There exists a vector $\boldsymbol{\beta}$ such that $\mathbf{x}'\boldsymbol{\beta} > 0$ for events and $\mathbf{x}'\boldsymbol{\beta} < 0$ for nonevents. Solving the logistic function for the probability shows that $\Pr(y = \text{event}) > \frac{1}{2}$, and similarly $\Pr(y = \text{nonevent}) < \frac{1}{2}$. In this case, the log likelihood converges to 0, the dispersion matrix becomes unbounded (so the standard errors go to infinity), and the maximum likelihood estimates $\hat{\boldsymbol{\beta}}$ diverge to infinity.
- *quasi-complete separation*: Same as complete separation except that there is at least one event observation and at least one nonevent observation that satisfies $\mathbf{x}'\boldsymbol{\beta} = 0$. In this case, the log likelihood converges to a value less than 0, the dispersion matrix becomes unbounded, and at least one maximum likelihood estimate $\hat{\beta}_i$ diverges to infinity.
- *overlap*: The maximum likelihood estimates are unique and finite.

By default, the four dedicated logistic regression procedures (LOGISTIC, SURVEYLOGISTIC, HPLOGISTIC, and LOGSELECT) detect these three categories. This paper focuses on the case of complete separation, where the model perfectly separates your events from your nonevents.

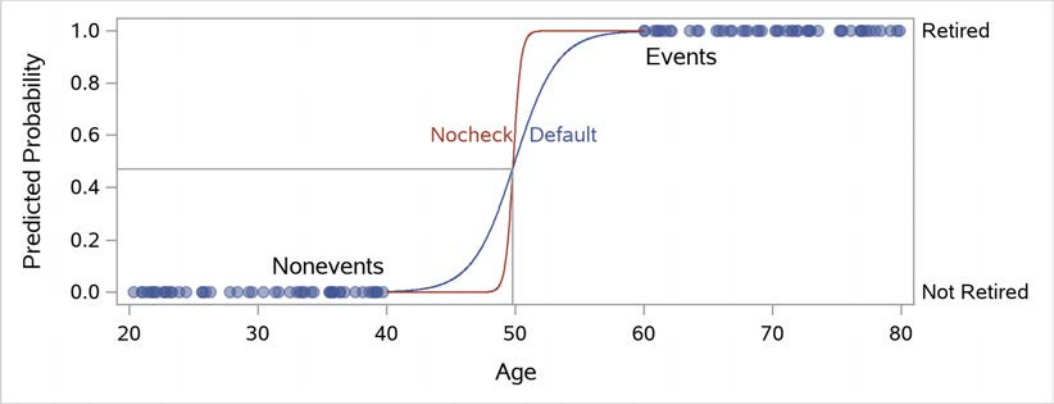
A "dead zone" is a region in oceans that has too little oxygen to support life, or a region that has poor cell-phone service. In this paper, it is the region between your events and nonevents where you have no data and cannot validate

your predictions. In particular, when your data are completely separable, there are infinitely many $\boldsymbol{\beta}$ vectors that separate your data; the dead zone is the set of all hypersurfaces defined by these separating vectors that are "parallel" to the fitted $\hat{\boldsymbol{\beta}}$, and the expanded dead zone is the set of all hypersurfaces that separate your data.

Complete separation often occurs when you have an overspecified model or not enough data, given your predictors. When your data are completely separated, PROC LOGISTIC stops after eight iterations but before numerical convergence. Stopping early is a useful way to deal with complete separation, because the resulting estimate of $\boldsymbol{\beta}$ still separates and predicts events and nonevents in a way that is conditioned by the data and the model, while avoiding potential computational problems (for example, dividing by numbers near zero). Dealing with complete separation simply by using the default results from PROC LOGISTIC is referred to in this paper as *accepting* the model.

Figure 1 displays a sample of 100 individuals, indicating their ages and whether or not they are retired from their career. Notice that all the subjects in the lower third of the range of ages, $20 \leq$ **Age** ≤ 40, are not retired, and subjects in the upper third, $60 \leq$ **Age** ≤ 80, are retired.

Figure 1 Two Logistic Regression Fits with One Continuous Covariate



The following program fits a simple logistic regression to these data:

```
proc logistic;
   model Retired(event='1') = Age;
run;
```

PROC LOGISTIC warns you about the existence of a dead zone by issuing this log message:

```
WARNING: There is a complete separation of data points.
         The maximum likelihood estimate does not exist.
```

PROC LOGISTIC sets the value of the system macro variable &SYSERR to 4 to indicate that a warning has been displayed (PROC HPLOGISTIC and PROC LOGSELECT do not display a warning and leave the &SYSERR value set to 0 if the program otherwise completed successfully), and it displays the convergence status in an output table (Figure 2).

Figure 2 Convergence Status Table

Model Convergence Status
Complete separation of data points detected.

The model that PROC LOGISTIC reports is

$$\text{logit}(\Pr(\textbf{Retired} = 1)) = -30.3054 + 0.6064 \cdot \textbf{Age}$$

Figure 1 displays the probability of being retired as predicted by this simple logistic regression fit as the blue curve (labeled "Default"). The model-predicted probabilities are near 0 for subjects younger than 40 and near 1 for subjects older than 60.

The one-dimensional hypersurface that separates the events (**Retired** $= 1$) from the nonevents (**Retired** $= 0$) is

$$\{\textbf{Age} : -30.3054 + 0.6064 \cdot \textbf{Age} = 0\}$$

That is, **Age** $= 49.98$. Because you have a finite amount of data, there is a linear subspace of the range of **Age** that contains the separating hypersurface (**Age** $= 49.98$) and contains no data. You can see that there are no subjects between the ages of 40 and 60 to influence how the model transitions from 0 to 1; the middle third of the data, $40 \leq$ **Age** ≤ 60, represent the dead zone.

PROC LOGISTIC responds to complete separation by stopping the optimization process early; this action yields the gently sloping blue curve. Your resulting model is from an incomplete model fit; the model separates the data, but its parameter estimates should be infinite and the predicted probabilities should be exactly 0 or 1. The LOGISTIC procedure provides the NOCHECK option to prevent the separation detection; rerunning the model with the NOCHECK option produces the red curve (labeled "Nocheck") in Figure 1. The optimization claims to converge, but this simply means that at least one convergence criterion is satisfied numerically—the estimates are heading to infinity. For this example, when you perform more than 39 iterations, the Hessian becomes noninvertible and PROC LOGISTIC automatically sets certain parameters to 0, which is not useful for this paper. You can see that the "Nocheck" curve is approaching a step function, where the predicted probabilities switch from $\hat{\pi} = 0$ to $\hat{\pi} = 1$ when the value of **Age** is about 50. When you have complete separation, predicted probabilities that are not nearly 0 or 1 are due to the fact that the optimization was stopped prematurely.

Figure 1 also shows the subjects that are used to *train* the model. If you have a new subject that was not used to fit the model, then it makes sense to classify it as a nonevent (not retired) if **Age** < 40 and as an event (retired) if **Age** > 60. But if you are scoring a new observation in which $40 \leq$ **Age** ≤ 60, then either you accept the results of the trained model and use **Age** $= 49.98$ as the decision cutpoint or you can choose to do something different.

This introductory section concludes by reviewing previous work in classification of observations in the dead zone and by discussing some general ideas regarding how to handle dead-zone observations. Subsequent sections discuss measuring and identifying the dead zone by using a variety of methods, from random sampling to linear programming. Complete SAS® programs and macros for all data sets and figures in this paper are available from the conference proceedings and online.[1]

Previous Work in This Area

To the best of my knowledge, this topic has not raised any red flags in the past, except for scoring beyond the range of the data that you use to train your model. There are a variety of classification and discrimination procedures in the literature and implemented in SAS, and one of their features is to classify observations, or assign probabilities to observations, no matter where they are. There are also many different methods to model binary response data and to produce predictions inside the dead zone, but the focus is always on producing a good model. Because complete separation implies that the covariance matrix is infinite, you cannot perform hypothesis testing with the results of such a fit, and it is suggested that you overspecified the model and you must do something ensure that your data have overlap so that finite maximum likelihood estimates exist (Allison 2012, pp. 47–59).

In the fields of predictive analytics and machine learning, a support vector machine (SVM) model finds an optimal hypersurface that divides the events from the nonevents; that is, it attempts to separate the two data clouds in a fashion similar to logistic regression. For SVMs, the lack of data in the dead zone is considered to be a *desirable* feature (Hamel 2009).

Should You Make Predictions Inside the Dead Zone?

Suppose you are writing a large, complex program that has one component in which a logistic regression model is fit to some data, and that model is then used to score (that is, compute predicted probabilities for) new data. If your new data lie inside this dead zone, should you accept the model-based predictions? You know that the predicted probability should be 0 or 1, but in the dead zone it might not be. Will you accept a value of, say, 0.4? Is it acceptable to classify a person as a 0 when he or she should be a 1? Is it reasonable to decide that a person needs medical treatment when it is not needed, or vice versa, especially when you have no data to back up the decision? One rule of thumb is not to predict data that exist (too far) outside the range of data that you used to create your model. Should this rule be extended to not making predictions *inside* the dead zone?

[1] Go to http://support.sas.com/rnd/app/stat/examples/sgf19_logisticregression.html.

These are decisions that you have to make for the analysis you are performing. You have three options:

- *Amend*: Change the model or method, or gather more data.
- *Accept*: Accept what the model provides.
- *Act*: Find the dead zone and act on your knowledge.

If you are fitting a single model and are more interested in testing hypotheses about the data, such as "this variable is important" or "the response is significantly different for this set of factors compared to that set of factors," then you will probably choose to *amend* your analysis. For example, you can drop predictors because of their complexity or according to an importance measure like partial correlations, or you can combine categories of classification variables. You can also use PROC LOGISTIC's FIRTH option to use a penalized likelihood, or, if your data set is small enough, you can try fitting an exact logistic regression, because both these methods are robust to separation effects. (For more information, see Allison 2012, pp. 47–59.)

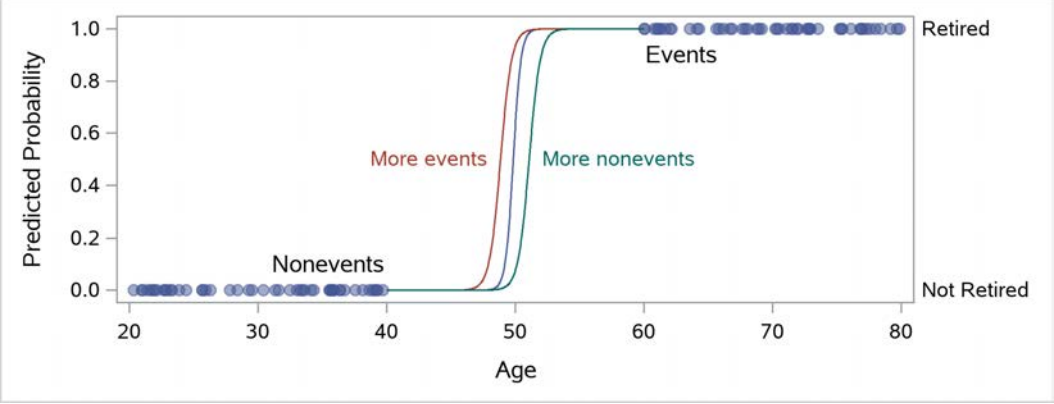
If you are interested in inference or predictive modeling and are using logistic regression for classifying new observations, then you might prefer one of the other two options: either ignoring the problem and accepting the model's prediction, or recognizing the model's limitations with dead-zone observations and using other principles to guide prediction. Either way, be aware that any predictions you make are not based purely on the data.

By choosing to *accept* the model, you are letting the model decide what to do in the dead zone. Your $\hat{\boldsymbol{\beta}}$ reflects the distribution and weight of the observations in a manner that is defined by your model. Complete separation is an attractive feature in predictive analytics because it gives you perfect prediction for the observed data; in particular, the receiver operating characteristic (ROC) curve goes through the point (0,1). In a linear regression, a perfect fit results in a curve that is too bumpy to reflect reality and is not particularly useful for scoring new observations. However, with binary response data, this problem is mitigated by using logistic regression as a classification technique; you are directly predicting a latent probability, and you don't really care what the exact value of the predicted probability is, as long as it gives you a model-based method of discriminating between events and nonevents.

You should also be aware that predictions in the dead zone are, appropriately, more heavily weighted to the prevalence of the events in your data. That is, if you have more events than nonevents, then you will predict more events in the dead zone; if you have more nonevents, then you will predict more nonevents. These dead-zone predictions are conditioned not by how the response depends on the covariates, but only by how many events and nonevents there are.

To illustrate this, the graph in Figure 3 compares the preceding logistic regression model to a model where the events are weighted more heavily and to another model where the nonevents are weighted more heavily. The center (blue) curve shows the original fit. You can see that having more events shifts the logistic fit to the left (red) curve, so that a larger portion of the dead zone is predicted as events. Similarly, the right (green) curve shows that having more nonevents in the data results in predicting more nonevents in the dead zone.

Figure 3 Logistic Regression Fit with More Events or Nonevents



None of the three fits in Figure 3 is any more accurate than the others for observations in the dead zone because there are no data available to validate the predictions inside this zone.

The remainder of the paper is concerned with how to *act* on your knowledge about the dead zone. If you choose this option, first you need to identify the dead zone, and then you need to treat predictions in a principled manner.

Your Observation Is Inside the Dead Zone. Now What?

What you should do with observations inside the dead zone varies according to your situation. Suppose that your program eventually makes a recommendation for some action—for example, to approve a new car loan, to make an offer to a potential customer, or to recommend a new medication. If you know that your new observation is in the dead zone, the following list suggests four ways to proceed:

- Allow the model to make a prediction, but flag it as suspect and take the recommendation with a grain of salt.
- Duplicate that observation, but for one observation predict an event and for the other predict a nonevent. Allow both predictions to flow through the rest of your program, and see whether the two final recommendations agree. If they do not, can you make a reasonable recommendation by averaging the two results or by using other information?
- If your response variable is numeric, you can try setting dead-zone observations to a value halfway between those of the events and nonevents, and then see how they proceed through your program.
- Flip a coin, or use the fitted logistic regression model to make a model-based flip; that is, choose to *accept* the model results and ignore separability.

IDENTIFYING THAT YOU HAVE A DEAD ZONE

The first step in handling the dead zone is to find out whether you have one—that is, you need to detect whether your data set is completely separable.

SAS procedures produce system macro variables (&SYSERR, &SYSRC) that indicate the results of an analysis. If you have complete separation, then both PROC LOGISTIC and PROC SURVEYLOGISTIC indicate a warning by setting the value of &SYSERR to 4; however, this setting can refer to any number of problems. PROC HPLOGISTIC and PROC LOGSELECT do not treat perfect prediction as a problem, so their return codes are not changed by separation. You should still check that these macro variables are both 0 in case other problems arise; in particular, the Figure 1 example with the NOCHECK option returns a nonzero &SYSERR value when you force PROC LOGISTIC to make more than 39 iterations because of a noninvertible Hessian.

All the dedicated logistic regression procedures identify separation in some fashion. Most other procedures that can perform logistic regression (for example, the BGLIMM, GENMOD, GLIMMIX, HPGENSELECT, MDC, PROBIT, and QLIM procedures) do not. For those procedures and actions that do not check for separation, you should check the parameter estimates and their standard errors because under separation the standard errors are moving to infinity, and large values are most likely due to separation. For the Bayesian BGLIMM procedure, the corresponding effect is that the posterior becomes very wide. Alternatively, as noted in the introduction, you can check for separation by scoring the training data and seeing whether using a predicted probability of 0.5 as the cutpoint between predicted events and nonevents perfectly predicts your response values.

You can obtain information about separation from the dedicated logistic regression procedures by using the "Convergence Status" table. The following table shows the values of variables in the "Convergence Status" table when you have complete separation.

Procedure	Reason	Status	Separation
PROC LOGISTIC	Complete separation of data points detected.	1	N/A
PROC SURVEYLOGISTIC	Complete separation of data points detected.	1	N/A
PROC HPLOGISTIC	Complete separation detected.	0	2
PROC LOGSELECT	Complete separation detected.	0	N/A

You can use the following program to check the "Convergence Status" table from any of these four procedures for complete separation:

```
%let csep=0;
data _null_; set ConvergenceStatus;
   if (find(reason,'Complete')) then call symput('csep',1); run;
```

The result is a macro variable, &CSEP, whose value is 1 if complete separation is detected and 0 otherwise.

MEASURING THE DEAD ZONE

When you know that you have a dead zone, the next thing you might want to do is determine its size.

If the dead zone is "wide," then there is a large separation between the event and nonevent data clouds, and you are more likely to have future observations that lie between them. If your goal is to distinguish between letters of the alphabet, which can reasonably be expected to lie comfortably in one cloud or the other, then a large separation might be considered a good feature. But if your goal is to recommend medical treatment for a wide variety of patients according to the results of a small sample, then this might be a bad feature.

If the dead zone is "narrow," then although you have little risk of trying to predict inside the dead zone, there is little separation between the data clouds, and your events and nonevents are quite close. It is possible that sampling more observations will create overlap in your data, which will remove the dead zone and enable you to produce appropriate estimates in that region and perform testing.

This section defines two different dead-zone measures: one based on sampling and the other based on the model. To begin, note that, given the logistic regression model for a completely separated data set, the fitted hyperplane that divides events from nonevents can be written as follows:

$$\{\mathbf{x} : \text{logit}(0.5) = 0 = \mathbf{x}'\hat{\boldsymbol{\beta}}\}$$

Let $\hat{\pi}_{\text{e}}$ be the minimum $\hat{\pi} = \text{logit}^{-1}(\mathbf{x}'\hat{\boldsymbol{\beta}})$ of the observed events, and let $\hat{\pi}_{\text{n}}$ be the maximum $\hat{\pi}$ of the observed nonevents. All predictions $\hat{\pi}$ inside the dead zone must satisfy $\hat{\pi}_{\text{n}} < \hat{\pi} < \hat{\pi}_{\text{e}}$. In fact, because the logistic regression model is linear in its parameters, each of these $\hat{\pi}$ defines another parallel hyperplane inside the dead zone. Estimate the dead zone by finding these two bounding values—any new observation whose model-predicted probability lies between these two values must be in the dead zone. So you can write the dead zone as follows:

$$\begin{aligned} \mathbf{D} &= \{\mathbf{x} : \text{logit}(\hat{\pi}_{\text{n}}) < \mathbf{x}'\hat{\boldsymbol{\beta}} < \text{logit}(\hat{\pi}_{\text{e}})\} \\ &= \{\mathbf{x} : \hat{\pi}_{\text{n}} < \hat{\pi}(\mathbf{x}) < \hat{\pi}_{\text{e}}\} \end{aligned} \tag{1}$$

The boundaries of the model-predicted dead zone are given by the following two hypersurfaces:

$$\begin{aligned} \mathbf{D}_{\text{n}} &= \{\mathbf{x} : \text{logit}(\hat{\pi}_{\text{n}}) = \mathbf{x}'\hat{\boldsymbol{\beta}}\} \\ \mathbf{D}_{\text{e}} &= \{\mathbf{x} : \text{logit}(\hat{\pi}_{\text{e}}) = \mathbf{x}'\hat{\boldsymbol{\beta}}\} \end{aligned}$$

One way of measuring the dead zone is to estimate the proportion of your data window that is covered by the dead zone; your data window is defined by the ranges of your main effects. You can estimate this measure by randomly sampling points from your data window, computing their predicted probabilities to determine whether they are in the dead zone $\mathbf{D}$, and reporting the proportion inside the dead zone; call this estimate the "sample proportion." You should take enough samples so that the standard error is small relative to the mean.

Another measure—call it the "margin proportion"—is based on computing the distance between the dead-zone boundaries, $d(\mathbf{D}_{\text{n}}, \mathbf{D}_{\text{e}})$, as a proportion of the maximum distance between any two observations, d_{max}. The distance between the two parallel hyperplanes along the vector normal to the two surfaces (the "margin" in SVM) is given by

$$d(\mathbf{D}_{\text{e}}, \mathbf{D}_{\text{n}}) = \frac{|\text{logit}(\hat{\pi}_{\text{e}}) - \text{logit}(\hat{\pi}_{\text{n}})|}{\| \hat{\boldsymbol{\beta}} \|}$$

where $\hat{\boldsymbol{\beta}}$ and the norm $\| \hat{\boldsymbol{\beta}} \| = \sqrt{\sum_i \beta_i^2}$ exclude the intercept. To compute d_{max}, let $\mathbf{x}_{\text{e}_{\text{max}}}$ be the observed event that is farthest from the fitted hyperplane, where the distance is computed by projecting $\mathbf{x}_{\text{e}_{\text{max}}}$ onto the normal vector to the fitted hyperplane. Let $\mathbf{x}_{\text{n}_{\text{max}}}$ be the observed nonevent that is farthest from the fitted hyperplane. Let $\hat{\pi}_{\text{e}_{\text{max}}}$ and $\hat{\pi}_{\text{n}_{\text{max}}}$ be their respective model-predicted event probabilities. The sum of these two distances equals the sum of the lengths of the two projections, which equals the distance between the two parallel hyperplanes that contain those two observations. So you can compute the maximum distance as follows:

$$d_{\text{max}} = \frac{|\text{logit}(\hat{\pi}_{\text{e}_{\text{max}}}) - \text{logit}(\hat{\pi}_{\text{n}_{\text{max}}})|}{\| \hat{\boldsymbol{\beta}} \|}$$

Taking the quotient of these two distances gives you a second measure of separation:

$$\frac{d(\mathbf{D}_e, \mathbf{D}_n)}{d_{max}} = \frac{|\text{logit}(\hat{\pi}_e) - \text{logit}(\hat{\pi}_n)|}{|\text{logit}(\hat{\pi}_{e_{max}}) - \text{logit}(\hat{\pi}_{n_{max}})|}$$

The MEASURE macro (available online) computes these two measures; the measures of separation for the data in Figure 1 are displayed in Figure 4.

Figure 4 Measures of Separation

Statistic	Value	StdError	n
Sample Proportion	0.3386	0.0047	10000
Margin Proportion	0.3413	.	.

As expected, both proportions are near $1/3$, corresponding to the fact that the dead zone ($40 \leq$ **Age** ≤ 60) takes up a third of the entire range of the data ($20 \leq$ **Age** ≤ 80).

Notice that these two statistics are computable and valid for a model that has any number of parameters; however, as you will see later, interactions and other dependencies can affect their computation.

SEARCHING FOR THE DEAD ZONE

Methods that work well in one dimension do not necessarily work well in multiple dimensions. This section and the next consider data that have two continuous predictors and discuss how to use logistic regression in different ways to find the dead zone.

Figure 5 displays 100 observations that are sampled from (part of) a bivariate normal distribution. Considering the predictor values only, it looks safe for you to make a prediction at (0,0) because it is near the center of the range of the **x1** and **x2** variables. However, when you consider where the events and nonevents occur (Figure 6), you see that the dead zone that was designed into the data, which is bounded by the gray diagonal lines, contains (0,0). If you run a logistic regression against these data, then you will detect complete separation. The MEASURE macro, the results of which are not shown here, reports the sample proportion as 0.1228 and the margin proportion as 0.1035.

Figure 5 Two Predictors

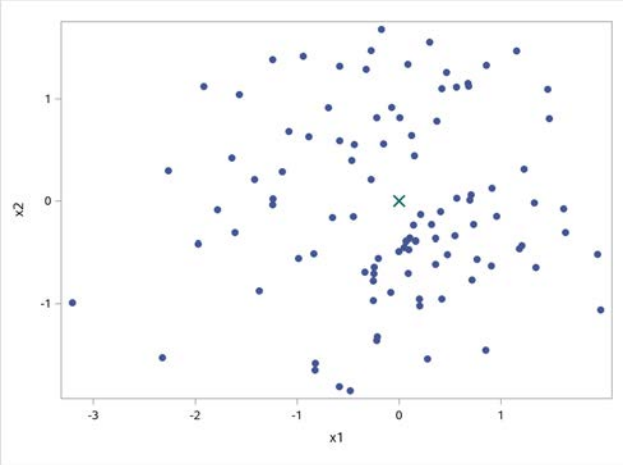
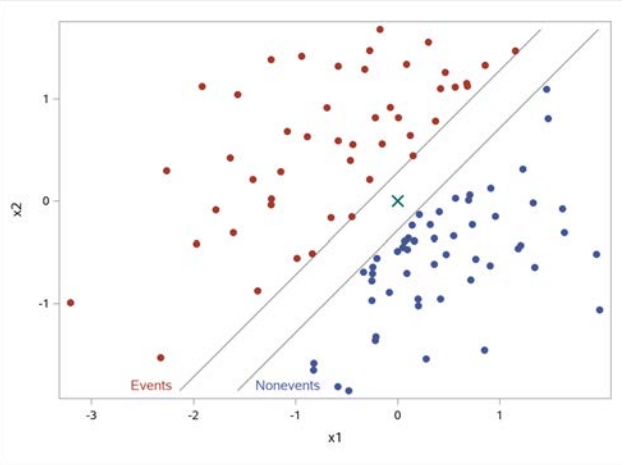


Figure 6 Two Predictors with Response Values Identified



Search Method 1: Using Predicted Probabilities from the Model

Figure 6 displays the true dead zone that is designed into the data, but how do you identify the dead zone if you did not simulate the data yourself? One method begins as in the last section, by computing the maximum predicted probability of the nonevents, $\hat{\pi}_n$, and the minimum predicted probability of the events, $\hat{\pi}_e$. Then you generate and score a grid of values across the space that is defined by the ranges of the predictors. The resulting predicted probabilities are compared to the $\hat{\pi}_n$ and $\hat{\pi}_e$ values, using equation (1) to determine whether the observation lies inside the dead zone.

When you have two main-effect predictors like this, you can find the dead-zone boundaries by solving the boundary equations for **x2** and computing **x2** for a range of given **x1** values. The USEPRED macro (available online) implements this approach.

The results for this data set are shown in Figure 7, which displays the data (events are red, nonevents are blue), two gray diagonal lines that mark the designed dead zone, a shaded gray region that the USEPRED macro has identified as the dead zone, and two green diagonal lines that show the boundary of the dead zone found by the USEPRED macro. The plot shows that this method is very effective in identifying the true dead zone. Of course, this result is too good to always be true.

This method of measuring the dead zone works not only with logistic regression but also with other approaches to making binary predictions. For example, consider support vector machines, for which the hypersurfaces that border the dead zone are the supporting hyperplanes, the observations on these boundaries are the support vectors, and the distance between these boundaries is the margin; this margin is maximized, and a hyperplane between the two supporting hyperplanes is the optimal solution (Hamel 2009). In Figure 2, **Age** $= 50$ corresponds to the decision surface, **Age** $\in \{40, 60\}$ corresponds to the boundaries, and the margin is $60 - 40 = 20$. SVM can handle nonlinear margins by using kernel estimators. The SVMACHINE procedure in SAS® Viya® implements the SVM algorithm and generates SAS code that you can use to score new observations, possibly with the SAS® Enterprise Miner™ procedure SVMSCORE. In SAS Enterprise Miner, PROC HPSVM and PROC SVMSCORE enable you to fit and score SVMs.

The same method that produces the plot in Figure 7 is used in PROC HPSVM to produce the plot in Figure 8, which shows that these procedures can find a region very similar to that found by logistic regression. For the examples in this paper, PROC HPSVM perfectly predicts the two sets just as PROC LOGISTIC does, and it also gives you the means to make predictions inside the dead zone.

Figure 7 Using Predicted Event Probabilities

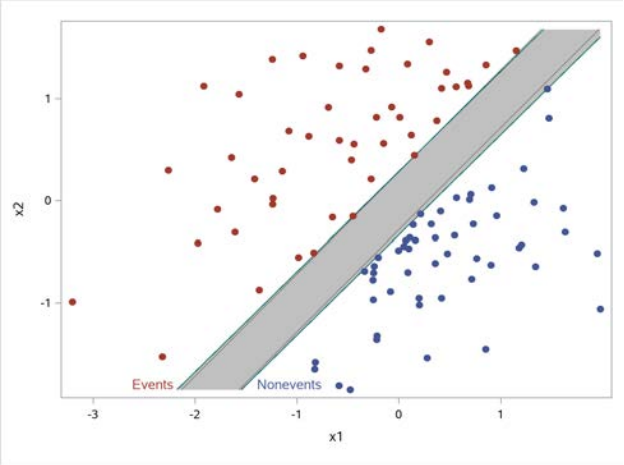
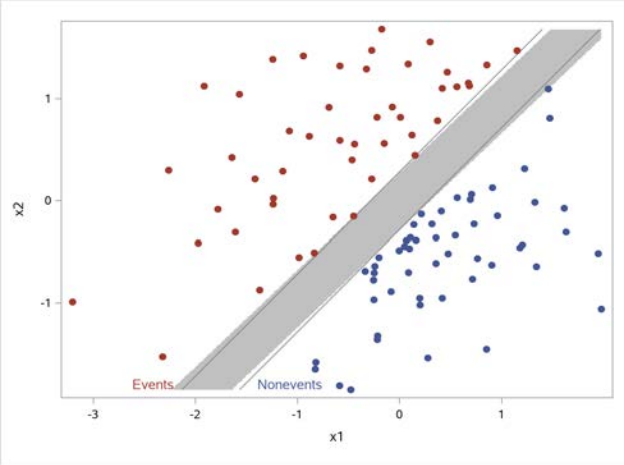


Figure 8 PROC HPSVM Results



Search Method 2: A Random Walk through the (Expanded) Dead Zone

The dead zone for the preceding data set is designed to be a slice through the data. The USEPRED macro finds a separating surface that best divides the events from the nonevents, given the data and the model, and the boundaries of the dead zone are parallel to this surface. Consider a dead zone like the one shown in Figure 9, which separates events from nonevents by a large X-shaped region. The USEPRED macro finds a slice through the zone, which is identified by the parallel green lines in the plot. The discussion in this section explores finding a more complex, expanded dead zone.

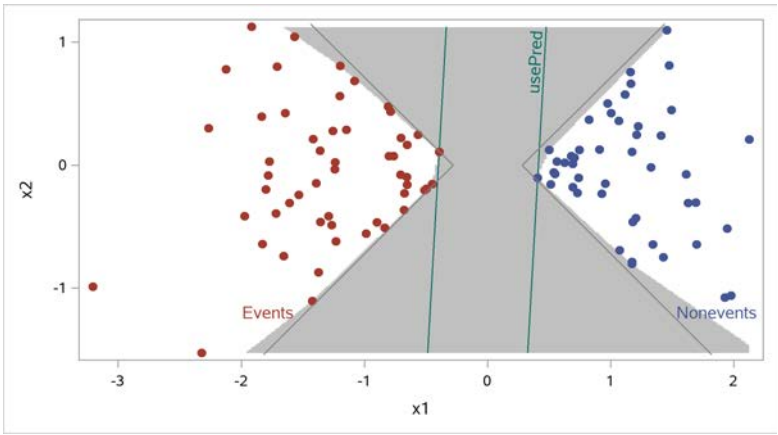
Borrowing ideas from Markov chain Monte Carlo methods and from random walks, start at the first identified separation equation from a logistic regression, and then repeat the following steps many times:

1. To each parameter estimate, add a value drawn from a Uniform(–0.5,0.5) distribution.
2. If this new surface does not completely separate the responses, then return to step 1. Otherwise, accept the new parameter estimates and continue to step 3.
3. Add the new dead zone to the expanded dead zone, replace the old parameter estimates with the new estimates, and return to step 1.

Because the reflective boundaries keep the iterations inside the expanded dead zone, this method can in principle scan over all possible models, given a sufficient number of iterations.

The WALK macro (available online) implements this method, and the WALKPLOT macro displays the plot in Figure 9. The expanded dead zone that is found by these macros, using 2,000 steps, is displayed as a shaded gray region. Also shown in this plot are the data (events are red, nonevents are blue), two gray polygonal lines that mark the designed dead zone, and two green diagonal lines that show the boundary of the dead zone identified by the USEPRED macro. The plot shows that this method does an exceptional job of identifying the expanded dead zone in very few steps. The MEASURE macro estimates the results for the zone that is bounded by the green lines; it reports that the sample proportion is 0.1524 and the margin proportion is 0.1543.

Figure 9 Random Walk with Boundaries



You can determine whether a new observation lies inside the expanded dead zone by rounding the new observation's covariates to the nearest value of the grid that the WALKPLOT macro uses to shade the dead zone, and seeing whether that grid value is identified as being in the dead zone. However, this approach does not generalize well to multiple predictors, because it quickly becomes time-consuming to process a large multidimensional grid. Instead, the WALK macro also stores all the coefficients for all the boundaries that successfully separate the events from the nonevents along with their corresponding $\text{logit}(\hat{\pi}_n)$ and $\text{logit}(\hat{\pi}_e)$. Given a new observation, score the observation to obtain $\hat{\pi}(\mathbf{x})$, and if $\text{logit}(\hat{\pi}_n) \leq \hat{\pi}(\mathbf{x}) \leq \text{logit}(\hat{\pi}_e)$ for any of these boundaries, then flag that observation as being inside the expanded dead zone. The WALKDEADZONE macro (available online) implements this method. The following program produces the table in Figure 10 to verify that the origin lies inside the expanded dead zone but the observation for which **x1** $= 1$ and **x2** $= 0$ does not:

```
data tmp; x1=0; x2=0; output; x1=1; x2=0; output; run;
%walkDeadZone(scoredata=tmp)
proc print data=tmp; run;
```

Figure 10 Origin Is in the Dead Zone

Obs	x1	x2	deadzone
1	0	0	1
2	1	0	0

The WALK and WALKDEADZONE macros are more time-consuming to run than the simpler USEPRED macro. The USEPRED macro might suffice for most purposes and is used in the remainder of this paper.

Three Dimensions

The two preceding search methods generalize easily to higher dimensions. This section presents two examples of data in three dimensions.

The first example has two predictors, **x1** and **x2**, and including their interaction term **x1*x2** as the third predictor embeds a two-dimensional saddle surface in the resulting three-dimensional space, where the interaction is the third dimension. Figure 11 shows the data on this surface projected onto the X1–X2 plane, and the USEPRED macro finds the dead zone, which is the shaded gray region. Although this is actually a two-dimensional problem, a logistic regression can find the dead zone only when looking at it in three dimensions; a main-effects model does not detect separation because the dead zone in Figure 11 is obviously not a linear subspace of the X1–X2 plane. If you compute the separation measures in three-dimensional space, then the sample proportion will be (nearly) 0, because you are trying to sample an embedded two-dimensional surface within a three-dimensional volume. If you compute the measures on the projection instead, then the margin is obviously not constant and the margin proportion is not well defined. In this example, the sample proportion in the X1–X2 plane is 0.0365, and the margin proportion in three-dimensional space is 0.012.

The second example has three main effects, **x1**, **x2**, and **x3**. The events and nonevents are sampled from non-intersecting normal ellipsoids in three-dimensional space. A main-effects logistic regression model determines this to be a completely separable data set. Figure 12 presents a two-dimensional view of this space rotated around the X2 axis so that you can look along the edge of the dead zone. The MEASURE macro computes both measures in three dimensions; the sample proportion is 0.0151, and the margin proportion is 0.0136.

Figure 11 Interaction Projected onto the X1–X2 Plane

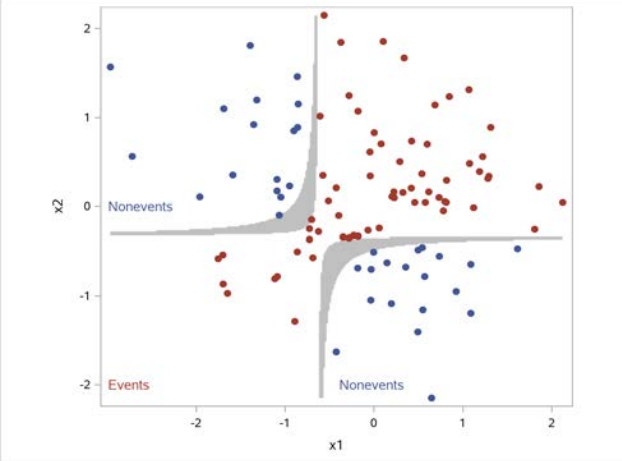
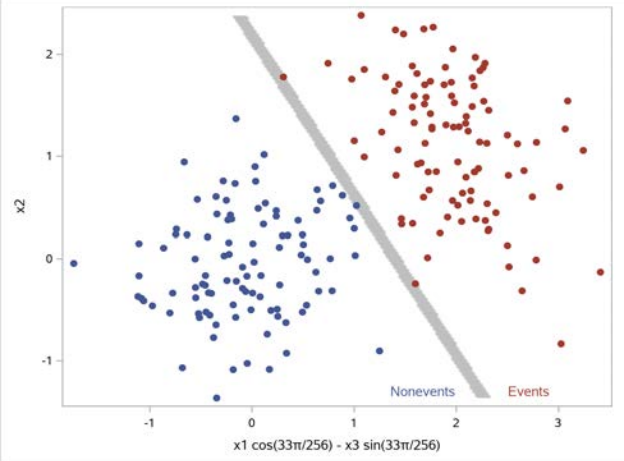


Figure 12 Ellipsoids Rotated and Projected



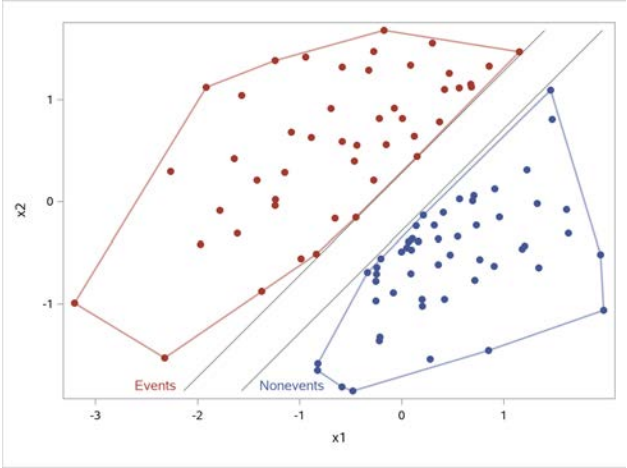
SEARCHING FOR THE CONVEX HULLS

The methods in the preceding sections enable you to search for the dead zone between the events and the nonevents, but they do not address scoring data that lie beyond the data cloud. Using the range of the observed data to limit which observations you score is always a smart policy, especially for complex heuristic models. If you want to limit your scoring even further, you can try to estimate the event and nonevent data clouds and avoid scoring outside the clouds. In two dimensions, this is simply finding the convex hull of these two sets. In SAS, it is convenient to use the CVEXHULL statement in SAS/IML® software to compute two-dimensional convex hulls.

The CVEXHULL macro (available online) computes the convex hull of the events and nonevents in the data set in Figure 6 and displays the results in Figure 13, along with the bounds of the dead zone that was designed into this example.

Restricting predictions only to the convex hulls of the events and nonevents is rather draconian, especially for points that lie far outside the dead zone. Computing convex hulls also becomes much harder as you add more predictors and observations. If you are interested in pursuing this approach in higher dimensions, then the work of Chazelle (1993) might be a good place to start.

Figure 13 Convex Hulls of the Events and Nonevents



The next section proposes two other methods to cluster the observations as events and nonevents and to fit new observations into these clusters. One drawback to using standard clustering methods is that they do not take into account the known response values, so they can easily misclassify your observed data. PROC MBC, the new model-based clustering procedure in SAS® Visual Statistics on SAS Viya, clusters the data and also models the white noise, but it still does not necessarily classify the observed data perfectly. Unlike clustering procedures, discriminant methods take known response levels into account, and PROC DISCRIM is the procedure that is designed for this. The following sections use PROC DISCRIM to estimate the hulls and the dead zone of the data shown in Figure 6.

The DISCRIM Procedure

Because there are predictors that separate the data, and because the data all have response values, an approach to identifying the dead zone might work like this:

1. Find the centroid of the events, and determine the weighted distance of the event observation that is farthest from the event centroid.
2. Find the centroid of the nonevents, and determine the weighted distance of the nonevent observation that is farthest from the nonevent centroid.
3. Given a new observation, compute its weighted distance from both centroids.
4. If the distance of the new point from each centroid is greater than the two maximum distances, then the point is definitely beyond the two hulls of the data and could possibly be in the dead zone.

You can compute multivariate distances in a variety of ways. One way is to use the Mahalanobis distance, which considers the size and shape of the data. This is the method that PROC DISCRIM uses by default. The DISCRIM macro (available online) implements this method by using PROC DISCRIM.

To create the plot in Figure 14, PROC DISCRIM uses the minimum event posterior probability and minimum nonevent posterior probability to classify the plane; the portion of the plane that is not assigned to either data cloud is unshaded. This is a rather poor estimate of the dead zone.

To create the plot in Figure 15, PROC DISCRIM uses the minimum density value for the events and the minimum density value for the nonevents to determine the classification, and it shades the two data clouds. The normality assumption gives you the expected elliptically shaped clouds. This method completely fails to find the dead zone, but it provides a reasonable estimate of the hull.

Figure 14 Normal Assumption Posteriors

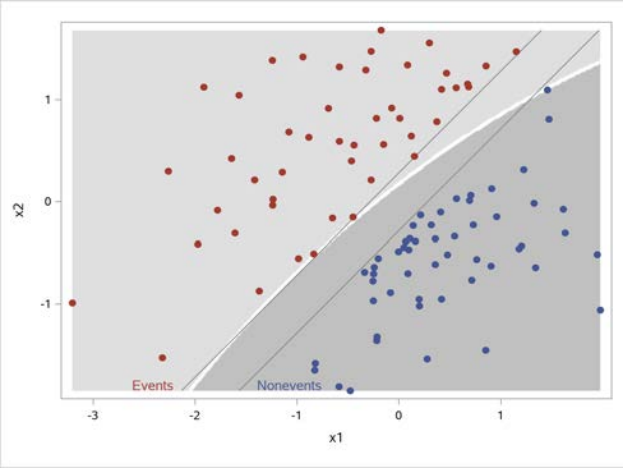
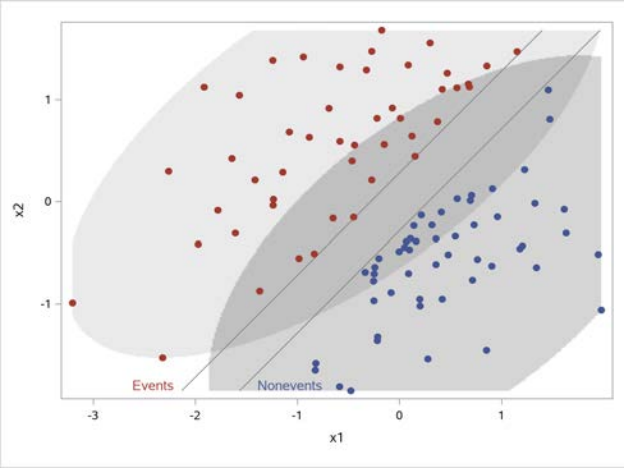


Figure 15 Normal Assumption Densities



You can implement the preceding approach by using the normal kernel density estimate (KDE) method instead of Mahalanobis distances; this method is also available in PROC DISCRIM and in the DISCRIM macro. The posterior probabilities that are greater than the observed minimum in each of the two data clouds are shaded in Figure 16. The unshaded region is a reasonable estimate of the dead zone. Finally, the densities from the nonparametric kernel density method are displayed in Figure 17. Using the minimum densities of the observed events and nonevents as the cutoffs for inclusion in the two clouds leaves you with disconnected sets as estimates of the hulls; using smaller cutoff values can enlarge and connect the clouds. Combining both of these results to estimate the hull and the dead zone also seems to be a reasonable approach.

Figure 16 KDE Posteriors

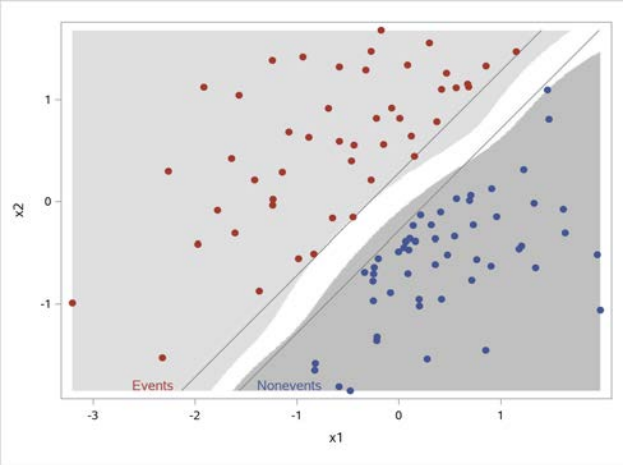
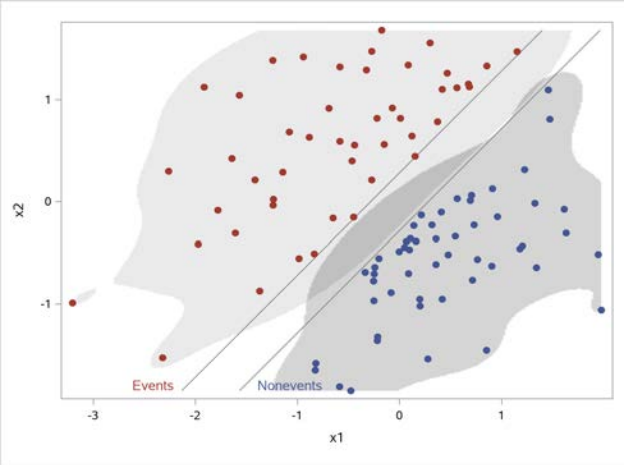


Figure 17 KDE Densities



SAMPLING FROM THE DEAD ZONE

In large-scale machine learning applications, logistic regression is often merely one component of a larger process. If you want to study the effect that data in the dead zone have on your entire process, or if you want to improve future data collection, you can run the process on a random sample of data points from the dead zone. Note that if you have the variables **x1** and **x2** and their interaction **x1*x2** in your model, then your sample space is actually a surface in a higher-dimensional space, and the probability of selecting a value on that surface is 0. One way to handle this is to sample only the main effects.

An approach to drawing a random sample of points in the dead zone is to sample from the full range of the main effects and reject the draw if it is not in the dead zone. You can use the MEASURE macro to run this sample by subsequently extracting the observations for which **SP** $= 1$ from the **TMP** data set produced by that macro. However, this can be an inefficient way to sample, because it often results in a large number of rejections. For the data shown in

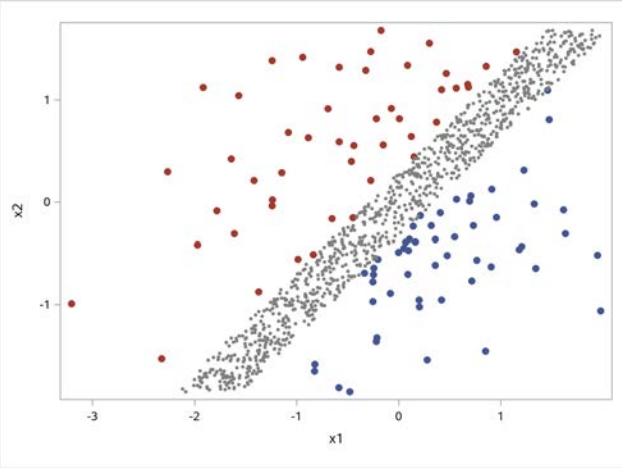
Figure 6, where the MEASURE macro reports that the "Sample Proportion" statistic is 0.1228, you will need about 1,000/0.1228=8,143 draws to obtain a sample of 1,000 observations from the dead zone.

To reduce the number of rejections, first choose a surface inside the dead zone by sampling a number π uniformly from $(\hat{\pi}_n, \hat{\pi}_e)$, then select a point in the "subspace" and project it onto the hypersurface defined by

$$\{\mathbf{x} : \text{logit}(\pi) = \mathbf{x}'\hat{\boldsymbol{\beta}}\}$$

You need to define "subspace" in such a way that no set of points is more likely to be chosen than another, so you cannot just sample from the range of your data. The SAMPLE macro (available online) assumes a main-effects model and solves the equation of the preceding hypersurface for the last main effect, samples points from the full range of the other main effects, then for each point finds the value of the extracted main effect that lies on the selected hypersurface and rejects that point if this value is outside the data range. Figure 18 displays a sample of 1,000 observations from the data shown in Figure 6, which requires 1,373 draws.

Figure 18 Sample from the Dead Zone



CONCLUSION

When you write a program that performs logistic regression or uses any other statistical modeling method, you should always programmatically monitor the results of the fit. If separation is detected, you should decide ahead of time how you will handle future predictions for observations from the dead zone. If you decide that your dead zone cannot support predictions, then the ideas discussed in this paper can aid you in detecting, measuring, and dealing with such data.

The paper uses continuous predictors primarily in main-effects models. For classification variables, separation is detected if the cell of a cross-classification has no data or if a cell contains only events or only nonevents. In these cases, simply dropping all observations in that cell (and noting that the cell is perfect) enables you to proceed. If you also have continuous predictors such that high values of a continuous predictor are events in one cell and nonevents in another cell, then separation is not detected. One computationally expensive approach to discovering more complex data separation is to fit a model to each cell, and then to handle cells for which separation is detected in a fashion as discussed in this paper.

The paper also ignores quasi-complete separation. The only difference from complete separation is that in quasi-complete separation, one or more observations lie on the surface defined by the $\hat{\boldsymbol{\beta}}$ that separates the events from the nonevents. If you can obtain more data, then it might be possible to create overlap in the data set. Otherwise, if you identify and remove these observations, then the remaining data set is completely separated and you can proceed from there.

If you use exact methods or Firth's penalized likelihood method, then complete separation is almost never detected, even if the data are truly separated. With these methods, likelihood-based confidence limits and likelihood ratio tests remain valid; however, neither of these methods completely solves the separation problem. Exact methods are computationally intensive even for small data sets and condition some of your parameters out of the model. The Firth algorithm uses a model's gradient and Hessian during the optimization, so selection methods require you to completely fit every candidate model, which can be computationally costly.

This paper investigates a question that deserves much more attention. Many other methods of analyzing binary data also make predictions in the presence of dead zones, including classification and regression trees, clustering and discriminant methods, Bayesian modeling, support vector machines, and generalized additive modeling. Because complex binary modeling is commonly used in modern machine learning applications, the effect of dead zones on these and other methods should also be explored.

APPENDIX

All data sets, programs, and macros that produce the outputs in this paper are available from the SAS® Global Forum 2019 conference proceedings and online. They assume that you have a response variable named **y** and a set of continuous predictors named **x1**, **x2**, and so on. The following macros are referenced in this paper:

MEASURE randomly samples from your data window, computes the proportion of observations in the dead zone along with a standard error, and reports this as the "Sample Proportion." Also finds the margin of the dead zone and the maximum possible distance between any two observations, then reports the quotient as the "Margin Proportion."

USEPRED fits a model by using PROC LOGISTIC or PROC HPSVM. The predicted event probabilities in your data window are compared to the $\hat{\pi}_{\mathrm{n}}$ and $\hat{\pi}_{\mathrm{e}}$ values to determine which points lie inside the dead zone. Results are stored in the **out** data set; observations in the dead zone set the indicator **sp** to 1.

WALK implements the random walk method of finding the expanded dead zone. In particular, an INEST data set inputs the candidate parameters, and PROC HPLOGISTIC with the MAXITER=0 option uses the candidate parameters to score the data and evaluate separation. The **out** data set contains the results. The **bounds** data set contains the parameters that define the separating surfaces and π_{e} and π_{n} for each separating parameter set.

WALKPLOT displays the plot in Figure 9.

WALKDEADZONE determines which observations in an input data set lie in the dead zone that is identified by the WALK macro. Computing $\boldsymbol{x}'\boldsymbol{\beta}$ in a SAS DATA step is computationally faster than scoring each model through PROC LOGISTIC. The indicator variable **deadzone**, whose value is 1 when an observation lies in the dead zone, is added to the input data set.

CVEXHULL draws the convex hull shown in Figure 13 by using the CVEXHULL statement in PROC IML.

DISCRIM uses PROC DISCRIM to estimate the dead zone and the convex hull discussed in the section "The DISCRIM Procedure."

SAMPLE generates a random sample of points from the dead zone for a given data set and displays a plot when you have two effects.

REFERENCES

Albert, A., and Anderson, J. A. (1984). "On the Existence of Maximum Likelihood Estimates in Logistic Regression Models." *Biometrika* 71:1–10.

Allison, P. D. (2012). *Logistic Regression Using SAS: Theory and Application*. 2nd ed. Cary, NC: SAS Institute Inc.

Chazelle, B. (1993). "An Optimal Convex Hull Algorithm in Any Fixed Dimension." *Discrete and Computational Geometry* 10:377–409.

Hamel, L. H. (2009). *Knowledge Discovery with Support Vector Machines*. Hoboken, NJ: John Wiley & Sons.

ACKNOWLEDGMENTS

The author is grateful to Dorothy Watson, and to David Schlotzhauer, Randy Tobias, and Ed Huddleston at SAS Institute for their valuable assistance in the preparation of this manuscript.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Bob Derr
bob.derr@sas.com