A Brief Introduction to the c-Statistic

STA2101H Project
Robert Zimmerman (1005054600)

1 Background

In ordinary linear regression, the R-squared statistic is typically used for model selection, as it is an intuitive measure of goodness-of-fit (although often misunderstood [**Kva85**]). However, the R-squared statistic does not generalize easily to every generalized linear model (GLM); in particular, it loses its interpretation (and usefulness) for logistic regression, where the response is dichotomous rather than continuous. Hence, other measures must be used in the model building process. There are several "pseudo-R-squared" statistics in use, but they are more difficult for non-statisticians to interpret, and each comes with its own drawbacks; for example, the popular McFadden's R-squared and similar measures are defined in terms of the log-likelihoods of the "null" model (an intercept-only model) and the current fitted model [**All13**]. Other goodness-of-fit statistics, such as the Kolmogorov-Smirnov statistic or Akaike's Information Criterion, apply more widely, but may also be difficult to interpret intuitively.

In this paper, we discuss the *c-statistic* (or the *concordance statistic*), a measure of statistical discrimination that is particularly well suited to logistic regression. It is often used in industry, but is not commonly discussed in the statistical literature in the context of regression models. For example, it does not appear in McCullagh and Nelder's seminal 1989 treatise on GLMs [**MN89**], and may have been popularized by Hosmer and Lemeshow in their treatise on logistic regression in the 2000s [**HL00**]. The idea of discrimination, however, is not new, and the equivalent concept of the ROC curve (as discussed briefly below) was first developed during World War II. [**FUW06**] The c-statistic, a very easily interpretable and powerful tool for model selection, measures how well a logistic regression model can discriminate between positive and negative outcomes.

More precisely, we recall that in the logistic regression setting, we have a sequence of data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$ where we assume y_i to be a realization of a $Y|\mathbf{X} = \mathbf{x}_i \sim \text{Bernoulli}(p(\mathbf{x}_i))$ random variable, and we wish to estimate p via the logit link function $\eta(p(\mathbf{X})) = \log(\frac{p(\mathbf{X})}{1-p(\mathbf{X})}) = \mathbf{X}^T\boldsymbol{\beta}$, which is done via the standard Fisher Scoring method. Suppose we have arrived at such an estimate $\hat{p}$; we use it to score a subset of our data (typically one held-out for validation purposes) and thus obtain the estimated probability $\hat{p}(\mathbf{x}_i)$ for each i.[1] Call a pair (y_i, y_j) of observed responses a *0-1 pair* if $y_i = 0$ and $y_j = 1$. We say that this 0-1 pair of observations is *concordant* if $\hat{p}(\mathbf{x}_j) > \hat{p}(\mathbf{x}_i)$, *discordant* if $\hat{p}(\mathbf{x}_j) < \hat{p}(\mathbf{x}_i)$, and *tied* if $\hat{p}(\mathbf{x}_j) = \hat{p}(\mathbf{x}_i)$. In words, the 0-1 pair is concordant when the model assigns a higher probability to the observation for which the event did occur than to the observation for which the event did not occur.[2]

Let $C = |\{(i,j) \in \{1,2,\ldots,n\}^2 : y_i < y_j \text{ and } \hat{p}(\mathbf{x}_i) < \hat{p}(\mathbf{x}_j)\}|$ be the number of concordant 0-1 pairs, let $T = |\{(i,j) \in \{1,2,\ldots,n\}^2 : y_i < y_j \text{ and } \hat{p}(\mathbf{x}_i) = \hat{p}(\mathbf{x}_j)\}|$ be the number of tied 0-1 pairs, and let $P = |\{i \in \{1,2,\ldots,n\} : y_i = 1\}|$ and $N = |\{i \in \{1,2,\ldots,n\} : y_i = 0\}|$ be the number of positive and negative responses, respectively. Then the c-statistic is calculated as $c = \frac{C+T/2}{PN}$. That is, it is the number of concordant pairs plus half the number of ties, expressed as a proportion of the total number of 0-1 pairs. In words, the c-statistic is the proportion of opposite outcomes in the data which the model "agrees" with (i.e., discriminates correctly). This is easily visualized with a table. If, for instance, there are three positive examples $\{y_{P_1}, y_{P_2}, y_{P_3}\}$ and four negative examples $\{y_{N_1}, y_{N_2}, y_{N_3}, y_{N_4}\}$ whose scores are compared in the following table, then the c-statistic is simply the number of "$>$" signs plus half the number of "$=$" signs

[1] In practice, it is computationally cheaper to obtain the estimated log-of-odds $\log\left(\frac{\hat{p}(\mathbf{x}_i)}{1-\hat{p}(\mathbf{x}_i)}\right)$, which would yield equivalent results owing to the monotonicity of the logit function.

[2] In theory, assuming independent observations, if our model includes at least one continuous covariate independent of any others, then the probability of a tie should be 0. However, the finite precision arithmetic used in all computer GLM implementations makes ties relatively common when dealing with millions of observations.

divided by the total number of cells (in this case, $\frac{7}{12} \approx 0.583$):

	y_{P_1}	y_{P_2}	y_{P_3}
y_{N_1}	$\hat{p}(\mathbf{x}_{P_1}) > \hat{p}(\mathbf{x}_{N_1})$	$\hat{p}(\mathbf{x}_{P_2}) = \hat{p}(\mathbf{x}_{N_1})$	$\hat{p}(\mathbf{x}_{P_n}) > \hat{p}(\mathbf{x}_{N_1})$
y_{N_2}	$\hat{p}(\mathbf{x}_{P_1}) < \hat{p}(\mathbf{x}_{N_2})$	$\hat{p}(\mathbf{x}_{P_2}) > \hat{p}(\mathbf{x}_{N_2})$	$\hat{p}(\mathbf{x}_{P_3}) < \hat{p}(\mathbf{x}_{N_2})$
y_{N_3}	$\hat{p}(\mathbf{x}_{P_1}) > \hat{p}(\mathbf{x}_{N_3})$	$\hat{p}(\mathbf{x}_{P_2}) > \hat{p}(\mathbf{x}_{N_3})$	$\hat{p}(\mathbf{x}_{P_3}) = \hat{p}(\mathbf{x}_{N_3})$
y_{N_4}	$\hat{p}(\mathbf{x}_{P_1}) > \hat{p}(\mathbf{x}_{N_4})$	$\hat{p}(\mathbf{x}_{P_2}) < \hat{p}(\mathbf{x}_{N_4})$	$\hat{p}(\mathbf{x}_{P_n}) < \hat{p}(\mathbf{x}_{N_n})$

While the R-squared statistic measures how well a linear regression model *fits* the data, the c-statistic measures how well a logistic regression model *discriminates* the data. Thus, a c-statistic of $\frac{1}{2}$ corresponds to a random guess, which clearly has no discriminatory power. On the other hand, a c-statistic of 1 corresponds to perfect discrimination. In practice, the c-statistic is never less than $\frac{1}{2}$, and therefore some authors even define it as a $[\frac{1}{2}, 1]$-valued function. Indeed, many GLM implementations initialize the Fisher Scoring algorithm with $\boldsymbol{\beta}^{(0)} = \mathbf{0}$, producing an intercept-only "null" model for which $\hat{p}(\mathbf{x}_i) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$ is constant for each $\mathbf{x}_i$, necessarily yielding a tie for *every* pair of observations[3]. Hence, the c-statistic here simply reduces to $\frac{0+PN/2}{PN} = \frac{1}{2}$. At each subsequent iteration, the algorithm can only produce a vector of coefficient estimates $\hat{\boldsymbol{\beta}}$ with at least as high a log-likelihood as that of the null model. In Section 2, we will show how to (artificially) achieve the extreme c-statistics of 0 and 1 on any binary dataset, and we will perform basic model selection on the `mathcat` data using the c-statistic, measuring our success via the cross-entropy.

The c-statistic has a wide variety of useful applications. For example, consider a logistic regression model that predicts the odds of developing some fatal disease; the results of this model might be used to decide whether to undertake some expensive preventive treatment. A low c-statistic suggests that the model would likely fail to distinguish adequately between patients who will develop the disease and those who will not, and such failures of prediction might lead to needless treatment and/or fatal omission of treatment.
The c-statistic is automatically output by SAS using the well-known PROC LOGISTIC procedure; however, it is not included in any standard functions or libraries in R. Note that c-statistic is equivalent to the area under the *Receiver Operating Characteristic (ROC) curve* (often used in binary classification machine learning tasks), which essentially measures the model's statistical power as a function of its Type I error [**HM82**]. Therefore, certain R libraries which approximate the area under the ROC curve can be used to estimate the c-statistic. However, these implementations typically approximate the area under the curve using the Trapezoid Rule, which can sometimes yield a somewhat different result from a direct calculation. In Appendix I, we have provided our own implementation as the function `cstat`, which is very inefficient from a computational perspective (with its nested `for` loop), but provides a transparent calculation of the c-statistic. A much faster method calculates the tensor product of the vectors induced by the positive and negative examples, which we include in Appendix II.

Despite its usefulness, the c-statistic does have some drawbacks. For example, it gives no information about how much the model discriminates positives from negatives, and could potentially lead to overfitting if relied on too heavily, as we will show in Section 2. The practice of "c-hacking"[4] – incrementally nudging up the c-statistic as much as possible by introducing nearly colinear explanatory variables and/or questionable interaction terms – is especially vulnerable to overfitting. Certainly, one should not rely solely on the c-statistic in the model-building process; rather, it is helpful in conjunction with other measures like AIC and traditional model diagnostic tools such as cross-validation, residual plots, and so forth.

[3] In this case, of course, $\hat{p}$ is simply the proportion of positive observations in the data.
[4] This is not standard terminology (yet).

2 Examples

Given any data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$, it is easy to see that the "estimates" $\hat{p}(\mathbf{x}_i) = y_i$ should produce a c-statistic of 1, while $\hat{p}(\mathbf{x}_i) = 1 - y_i$ produce a c-statistic of 0. We demonstrate this below, using the `cstat` function on a random sequence in $\{0,1\}$:

```
cstat(cbind(noise, noise))
[1] 1
cstat(cbind(noise, 1-noise))
[1] 0
```

This example suggests that relying on the c-statistic alone for model selection on a reasonably-sized dataset can be dangerous; as the c-statistic approaches 1, the closer we get to achieving a perfect discrimination of our sample data. Realistically, a complete separation of all possible data generated from the underlying distribution is likely unobtainable, assuming that our response is not a deterministic function of our explanatory variables. Having obtained a c-statistic which is very close to 1, we have likely overfit our data.[5]

To demonstrate a more interesting use of the c-statistic, we perform a simple model selection on the `mathcat` data, in which we aim to predict the probability of a student passing a course (as indicated by the binary response `passed`) given several other explanatory variables. We split the data into a 70% modelling sample and a 30% hold-out sample for validation, measuring our model's accuracy on the validation sample using the cross-entropy loss function, defined as

$$L(\mathbf{y}, \hat{\boldsymbol{\pi}}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{\pi}_i) + (1 - y_i)\log(1 - \hat{\pi}_i).$$

In spite of our earlier warning, we rely solely on the c-statistic for guiding our variable selection process for the purposes of demonstration. For convenience, we also define the wrapper functions `cstat_model` and `CE_model`, both of which both accept a "glm" object and format the predicted and actual responses into an array from which the c-statistic and the cross-entropy can be easily calculated (the former via the `cstat` function).

We begin with an intercept-only `model0`, whose c-statistic agrees with our discussion in Section 1. We also take note of the model's cross-entropy on the validation sample:

```
> model0 <- glm(passed ~ 1, family="binomial", data=mathcat.train)
> cstat_model(model0)
[1] 0.5

> CE_model(model0, mathcat.test, "passed")
[1] 0.6609283
```

Next, we create `model1` by adding in the variable `hsengl` representing a student's High School English grade[6]:

```
> model1 <- glm(passed ~ hsengl, family="binomial", data=mathcat.train)
> cstat_model(model1)
[1] 0.6004891
```

[5] However, it is usually considerably difficult to achieve a nearly-perfect c-statistic on "real" data, even if we are not concerned with overfitting. This can be easier when modelling with highly imbalanced datasets with a sufficiently large number of observations (such datasets may may arise in rare events modelling, for example).

[6] Of course, we know that `hsengl` is not a particularly strong predictor in the presence of others available in the dataset. In practice, we would begin with a variable which we already believed to be highly predictive of the response based on our prior knowledge – in this case, our experience in STA2101H.

The addition of `hsengl` alone results in a model which discriminates about 20% more accurately than a random guess, which is not very impressive. In place of `hsengl`, we try the categorical variable `course` representing the course type. However, the resulting c-statistic decreases slightly:

```
> model2 <- glm(passed ~ course, family="binomial", data=mathcat.train)
> cstat_model(model2)
[1] 0.5697554
```

We try `hsengl` and `course` together, and we obtain a more satisfying c-statistic:

```
> model3 <- glm(passed ~ hsengl + course, family="binomial", data=mathcat.train)
> cstat_model(model3)
[1] 0.6303533
```

Thus, `hsengl` and `course` provide more discrimination together than they do separately. Now we try adding in an interaction term between these two variables:

```
> model4 <- glm(passed ~ hsengl + course + hsengl*course, family="binomial", data=mathcat.train)
> cstat_model(model4)
[1] 0.6311141
```

The c-statistic is virtually unchanged, indicating that the interaction has little effect on the model's discrimination ability. Note that it is possible for the c-statistic to actually *decrease* following the inclusion of additional terms in the model. This highlights a difference between the c-statistic and measures such as R-squared, which cannot decrease when additional terms are included in a linear regression model. Often, a decrease in the c-statistic is caused by the introduction of extraneous interaction terms (as was almost the case here).

Instead of the interaction, we add in the variable `hsgpa` representing a student's High School GPA:

```
> model5 <- glm(passed ~ hsengl + course + hsgpa, family="binomial", data=mathcat.train)
> cstat_model(model5)
[1] 0.7821467
```

This is a considerable improvement; our model now discriminates over 56% more accurately than a random guess. We try an interaction between `course` and `hsgpa`:

```
> model6 <- glm(passed ~ hsengl + course + hsgpa + hsgpa*course,
+               family="binomial", data=mathcat.train)
> cstat_model(model6)
[1] 0.7824728
```

Aiming for a slightly more parsimonious model, we find that we can remove the separate `course` and `hsgpa` terms[7] and retain the previous c-statistic:

```
> model7 <- glm(passed ~ hsengl + hsgpa*course, family="binomial", data=mathcat.train)
> cstat_model(model7)
[1] 0.7824728
```

At this point, we are satisfied with the discriminatory ability we have achieved with our model. We could continue on and "c-hack" our way to a higher c-statistic if we so desired:

```
> model34 <- glm(passed ~ poly(hsgpa, 4) + poly(hsengl, 4) + i_hscalc
+               + i_Elite*sqrt_hsengl + i_Catchup*sqrt_hsengl + i_Elite*hsengl
+               + i_Catchup*hsengl + i_Elite*log_hsengl*hsgpa + i_Catchup*log_hsengl*hsgpa
```

[7] We reject the insistence of some textbook authors that interaction terms should not be included without their main effects.

```
+                , family="binomial", data=mathcat.train)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> cstat_model(model34)
[1] 0.8002446
```

Of course, this model is absurd; for only a slight increase in the c-statistic, we have reduced ourselves to a nonsensical model which overfits our data so badly that R even warns us that we have succeeded in interpolating some of our data at machine precision. We content ourselves with `model5`, and find that our validation cross-entropy has reduced considerably from that of `model0`:

```
> CE_model(model7, mathcat.test, "passed")
[1] 0.4962625
```

We also confirm that the improvement in our c-statistic agrees with an improvement in the AIC:

```
> model0$aic
[1] 375.8341
> model7$aic
[1] 321.6649
```

References

[**HM82**] J. A. Hanley and B. J. McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982). PMID: 7063747, pp. 29–36. DOI: 10.1148/radiology.143.1.7063747. eprint: https://doi.org/10.1148/radiology.143.1.7063747. URL: https://doi.org/10.1148/radiology.143.1.7063747.

[**Kva85**] Tarald O. Kvalseth. "Cautionary Note about R^2". In: *The American Statistician* 39.4 (1985), pp. 279–285. ISSN: 00031305. URL: http://www.jstor.org/stable/2683704.

[**MN89**] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. London New York: Chapman and Hall, 1989. ISBN: 9780412317606.

[**HL00**] David W. Hosmer and Stanley Lemeshow. *Applied logistic regression*. New York: Wiley, 2000. ISBN: 0471356328.

[**FUW06**] Jerome Fan, Suneel Upadhye, and Andrew Worster. "Understanding Receiver Operating Characteristic (ROC) Curves". In: *Canadian Journal of Emergency Medicine* 8.1 (2006), pp. 19–20. DOI: 10.1017/S1481803500013336.

[**All13**] Paul D. Allison. *What's the Best R-Squared for Logistic Regression?* Feb. 2013. URL: https://statisticalhorizons.com/r2logistic/.

Appendix I - R Code

Note that the following does not include the commands already shown in Section 2 above.

```
set.seed(1729)

cstat <- function(scored) # accepts an array with rows (y_i, p(x_i))
{
  ones <- scored[which(scored[,1] == 1),2] # keep track of the 1's in the data
  zeros <- scored[which(scored[,1] == 0),2]
  onecount <- length(ones) # count how many 1's we have
  zerocount <- length(zeros)
  Ccount <- 0

  for (i in 1:onecount)
  {
    for (j in 1:zerocount)
    {
      Ccount <- Ccount + (ones[i] > zeros[j]) + 0.5*(ones[i] == zeros[j]) # count pair by pair
    }
  }

  return(Ccount/(onecount*zerocount))
}

cstat_model <- function(model){(as.numeric(cstat(cbind(as.numeric(model$y), predict(model)))))}

CE_model <- function(model, data, resp)
{
  y <- as.numeric(data[[resp]])
  phat <- predict(model, newdata=data, type="response") # can't use predicted log-odds this time!
  return(-1*mean(y*log(phat) + (1-y)*log(1-phat)))
}

noise <- round(runif(n=100)) # a random sequence in {0,1}

mathcat <- read.table("http://www.utstat.utoronto.ca/~brunner/data/legal/mathcat.data.txt")

# some potential transformations to try out
mathcat$passed <- as.numeric(mathcat$passed) - 1
mathcat$i_Elite <- as.numeric(mathcat$course == "Elite")
mathcat$i_Catchup <- as.numeric(mathcat$course == "Catch-up")
mathcat$i_hscalc <- as.numeric(mathcat$hscalc == "Yes")
mathcat$log_hsengl <- log(mathcat$hsengl)
mathcat$log_hsgpa <- log(mathcat$hsgpa)
mathcat$sqrt_hsengl <- sqrt(mathcat$hsengl)
mathcat$sqrt_hsgpa <- sqrt(mathcat$hsgpa)

# randomly split our data into 70/30 training/validation samples
mathcat.train <- mathcat[1:floor(nrow(mathcat)*0.7), ]
mathcat.test <- mathcat[(floor(nrow(mathcat)*0.7)+1):nrow(mathcat), ]
```

Appendix II - Alternate Implementation

The following code computes the *c*-statistic much more efficiently, using the `kronecker` function included in the base R package:

```
cstat2 <- function(scored) # still takes in an array with rows (y_i, p(x_i))
{
  ones <- scored[which(scored[,1] == 1),2]
  zeros <- scored[which(scored[,1] == 0),2]
  op <- kronecker(ones, zeros, "-")
  return(sum(1*(op > 0) + 0.5*(op == 0))/(length(ones)*length(zeros)))
}
```