Linear Decision Boundaries from Generalized Linear Models

Maximilian Topps

9 October 2018

Introduction

This short write-up explains the method and mathematical background of plotting a decision boundary using a generalized linear model. The planned use of this method is to compare the weight of two acoustic cues in a phonetic contrast using linear gradients.

A generalized linear model (GLM) is a type of logistic regression despite its misleading name (not to be confused with general linear models). These are performed with the glm() function in R.

A GLM comprises a linear function and a link function. Using a logit link function, glms can be applied to binary output data in a multivariate model. Without polynomial factors, this produces as linear boundary.

For a binary discrimination task, we codify one choice as 1, and the other as 0.

Decision Boundary

The logit link function is used because of the binary nature of the output data. This is what makes the model logistic. It is a sigmoid function that takes continuous input and outputs between 0 and 1.

Where $h_\theta(x)$ represents outcome h_θ with input x, we know that:

$$0 \leq h_\theta(x) \leq 1$$

When plotting a decision boundary, we want to know the point at which one classification becomes another. In other words, the point at which each outcome is equally likely, such that "above" this boundary, we predict 1, and "below" it we predict 0. This would be where $h_\theta(x) = 0.5$

Owing to the sigmoid function, we can intuit that we are looking for line for which $z = 0$. This is because $z = 0$ is where $g(z) = 0.5$. See the below image for a graphical depiction.

Logit Link Function

The logit link function can be expressed as follows:

$$h_\theta(x) = \frac{1}{1 + (e^{-z})}$$

Where in an analysis with two predictor variables, x_1 and x_2:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

Note: θ_n represents coefficient for factor x_n

Algebra

Reminder that the logit link function is:

$$h_\theta(x) = \frac{1}{1 + (e^{-z})}$$

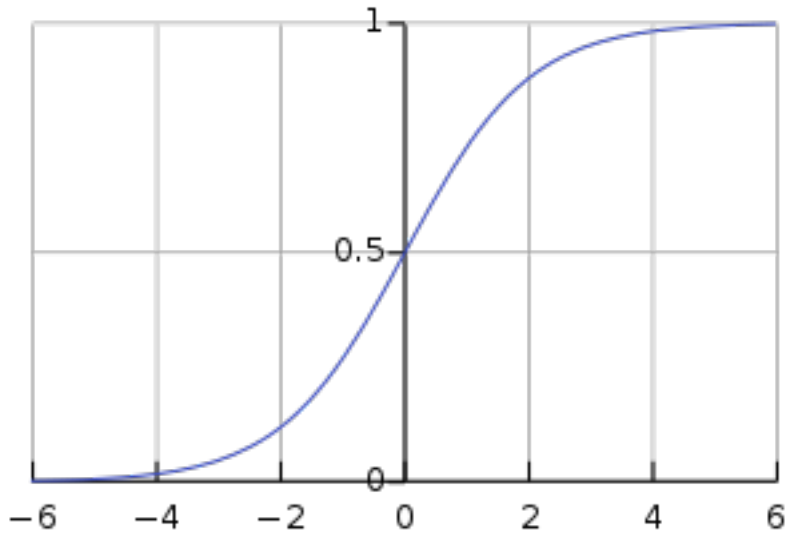


Figure 1: Sigmoid Function, from Wikipedia

We're looking for $h_\theta(x) = 0.5$. So that leaves us with:

$$0.5 = \frac{1}{1 + (e^{-z})}$$

We need to rearrange the equation. First, let's multiply both sides by $1 + (e^{-z})$

$$0.5 \times (1 + (e^{-z})) = 1$$

Next, multiply by 2 in order to remove the $0.5 \times \ldots$

$$1 + (e^{-z}) = 2$$

Minus 1 from each side

$$e^{-z} = 1$$

Natural log of each side

$$-z = 0$$

Remember that z is the linear function we saw above. We can plug that back in to get:

$$0 = -\theta_0 - \theta_1 x_1 - \theta_2 x_2$$

Note that we're plugging this in to $-z$, so $+$ becomes $-$

How do we plot this?

We need to represent the equation in cartesian space. We already know that we're plotting two factors x_1 and x_2 against each other. This means that we can use one and the x-axis and the other as the y-axis. Let's say that we want x_2 to be the y-axes. We need to represent $0 = -\theta_0 - \theta_1 x_1 - \theta_2 x_2$ in terms of x_2. Start with:

$$0 = -\theta_0 - \theta_1 x_1 - \theta_2 x_2$$

Next, add $\theta_2 x_2$

$$\theta_2 x_2 = -\theta_0 - \theta_1 x_1$$

Then, divide by θ_2

$$x_2 = \frac{-\theta_0}{\theta_2} - \frac{\theta_1}{\theta_2} x_1$$

You may have noticed that this equation is in the form $y = mx + b$, a linear equation, or $y = b + mx$ in this case. That means we can use our coefficients and axes to plot the linear boundary.

Applying to glm() output

Let's generate a random dataset that approximates an imaginary two-dimensional vowel contrast involving F2-F1, coded as `f2f1`, and duration in ms, coded as `duration`.

```
# set a random seed
set.seed(4321)

# generate data for our first vowel
vowel_zero <- data.frame(0, rnorm(1000,100,10), rnorm(1000,2000,100))
names(vowel_zero) <- c("vowel","duration","f2f1")

#generate data for our second vowel
vowel_one <- data.frame(1, rnorm(1000,100,10), rnorm(1000,1750,100))
names(vowel_one) <- c("vowel","duration","f2f1")

#bind them together to create our dataset
vowel_data <- rbind(vowel_zero, vowel_one)
```

Now that we have a dataset, we'll use `glm()` to find coefficients.

```
vowel_model <- glm(vowel ~ duration + f2f1, family=binomial(link=logit), data=vowel_data)
vowel_coefs <- coef(vowel_model)
```

Now that we have the model, we calculate the coefficients. Remember our function from the previous section:

$$x_2 = \frac{-\theta_0}{\theta_2} - \frac{\theta_1}{\theta_2} x_1$$

We've coded `duration` as x_1 and `f2f1` as x_2. Also, remember that θ represents our corresponding coefficients. This leaves us with the equation:

$$f2f1 = \frac{-(Intercept)}{f2f1\_coefficient} - \frac{duration\_coefficient}{f2f1\_coefficient} duration$$

To calculate these in R:

```
# intercept
boundary_intercept <- -vowel_coefs[1]/vowel_coefs[3]

# slope
boundary_slope <- -vowel_coefs[2]/vowel_coefs[3]
```

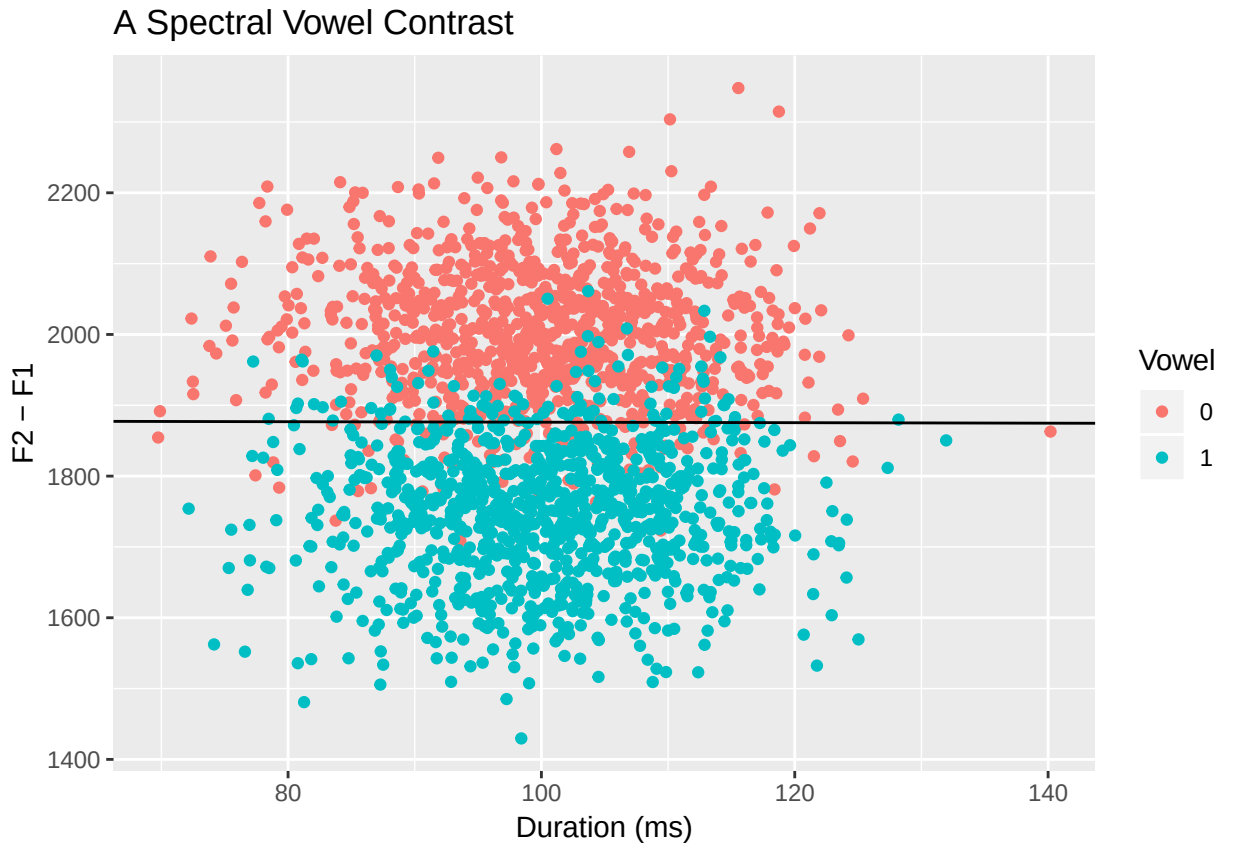
Plugging them into the equation:

$$y = intercept + slope \times x$$

So we plot it, along with the datapoints as follows:

```
# make sure we include ggplot2
library(ggplot2)
```

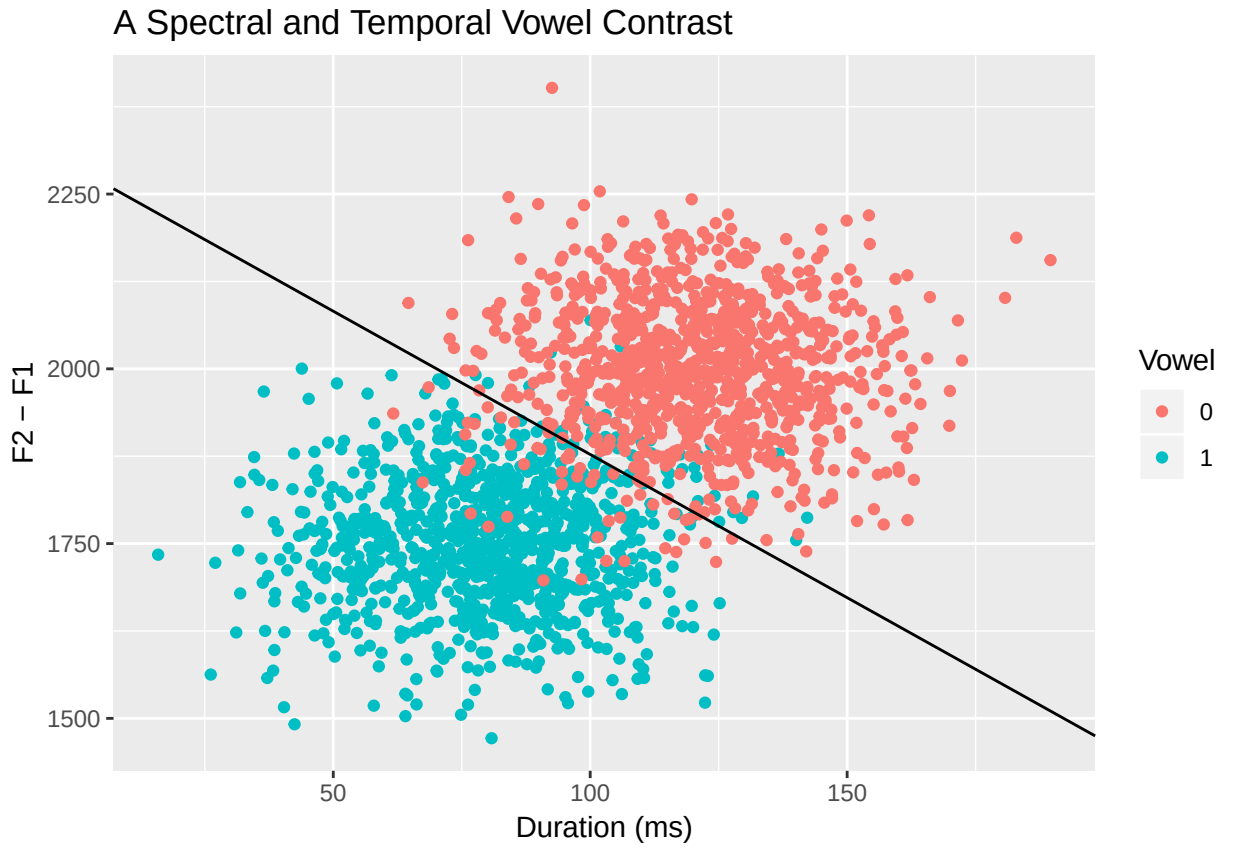
```
# plot the vowels
vowel_plot <- ggplot(vowel_data, aes(x=duration, y=f2f1)) +
  geom_point(aes(color=factor(vowel))) +
  geom_abline(intercept=boundary_intercept, slope=boundary_slope) +
  labs(title="A Spectral Vowel Contrast", colour="Vowel", x="Duration (ms)", y="F2 - F1")
vowel_plot
```



For this dataset, our decision boundary is very flat. This is expected, as we used the same input for the duration distribution, whilst changing the formants. Let's do the same with a contrast that varies on both cues.

```
# set random seed
set.seed(4321)

#generate vowel zero data - note the longer duration
vowel_zero_2 <- data.frame(0, rnorm(1000,2000,100), rnorm(1000,120,20))
names(vowel_zero_2) <- c("vowel","f2f1","duration")

#generate vowel one data - shorter duration
vowel_one_2 <- data.frame(1, rnorm(1000,1750,100), rnorm(1000,80,20))
names(vowel_one_2) <- c("vowel","f2f1","duration")

# put both vowels in one data frame
vowel_data_2 <- rbind(vowel_one_2,vowel_zero_2)

# calculate coefficients with glm()
vowel_model_2 <- glm(vowel ~ duration + f2f1, family=binomial(link=logit), data=vowel_data_2)
```

```r
vowel_coefs_2 <- coef(vowel_model_2)

# intercept and slope of linear decision boundary
boundary_intercept_2 <- -coef(vowel_model_2)[1]/(coef(vowel_model_2)[3])
boundary_slope_2 <- -coef(vowel_model_2)[2]/(coef(vowel_model_2)[3])

# plot
vowel_plot_2 <- ggplot(vowel_data_2, aes(x=duration, y=f2f1)) +
  geom_point(aes(color=factor(vowel))) +
  geom_abline(intercept=boundary_intercept_2, slope=boundary_slope_2) +
  labs(title="A Spectral and Temporal Vowel Contrast", colour="Vowel", x="Duration (ms)", y="F2 - F1")
vowel_plot_2
```



Now our plot has a steep boundary because we deliberately made the vowel durations consistently far apart. This represents a vowel contrast that operates both spectrally and temporally.