WSABI Summer Research Lab **Summer 2025**

Lecture 4: Logistic Regression

Instructor: Jonathan Pipping *Author: Ryan Brill*

4.1 Example: Sinking Putts

4.1.1 Problem Setup

Question: Predict the **probability** that a putt is sunk as a function of the distance to the hole.

Data: 5,988 putts including distance to the hole and whether the putt was sunk or not. Each row in the dataset is a single putt and includes the following variables:

- i: index of the i^{th} putt in our dataset
- y_i: 1 if the i^{th} putt was sunk, 0 otherwise
- x_i: distance to the hole of the i^{th} putt

Visualization: The first thing we should do is plot the data, which we do in Figure 4.1. What do you notice?

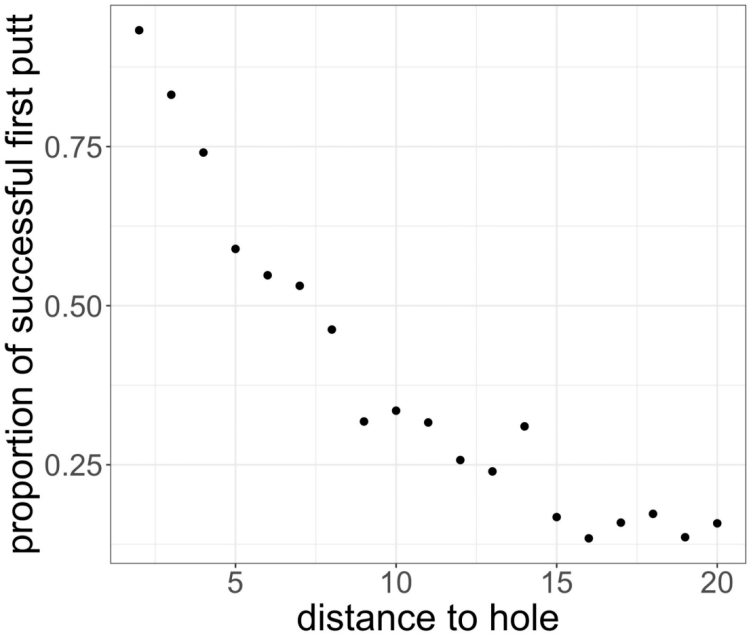

Figure 4.1: Scatter plot of the putt data

Like we would expect, the observed proportion of putts sunk decreases as the distance to the hole increases. Let's try to use what we've learned about linear regression to model this relationship.

4.1.2 Modeling

Model 1: Simple Linear Regression

We know from Lecture 1 that the this model takes the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \mathbb{E}[\epsilon_i] = 0$$

We already know how to estimate the coefficients for this model. Recall that the least squares solution to this model is given by:

$$\widehat{\beta}_0 = \bar{Y} - \widehat{\beta}_1 \bar{X}$$
$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

We fit this model in `R` with the following code:

```
model_1 = lm(putt_made ~ distance, data = putt_data)
summary(model_1)
```

When we plot the fitted line against the data in Figure 4.2, we see that the model does not fit the data well.

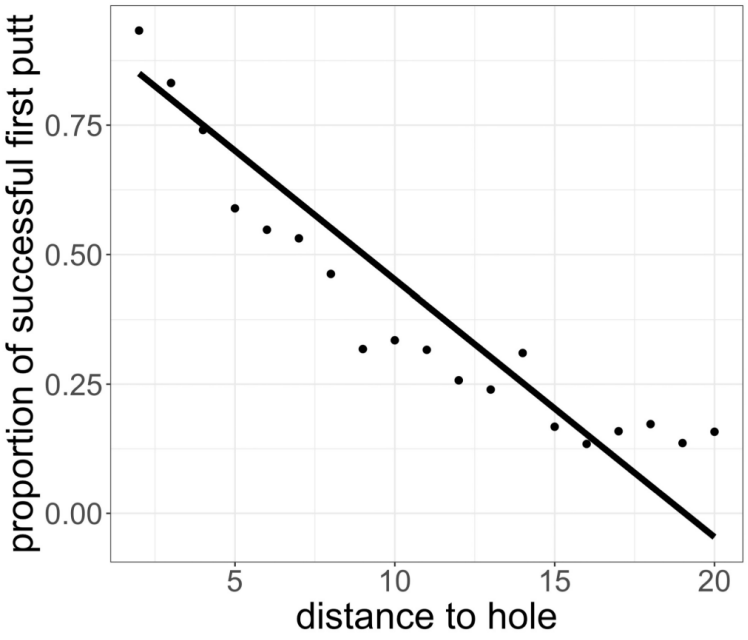

Figure 4.2: Putt data and fitted simple linear regression

We then move on to a more complex model: a cubic regression similar to the quadratic model we fit in Lecture 2.

Model 2: Cubic Regression

We know the form of this model as well:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i, \quad \mathbb{E}[\epsilon_i] = 0$$

We also know how to estimate these coefficients from Lecture 2. Recall that the least squares solution to this model is given by:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

We fit this model in `R` with the following code:

```
model_2 = lm(putt_made ~ distance + I(distance^2) + I(distance^3),
             data = putt_data)
summary(model_2)
```

When we plot the fitted line against the data in Figure 4.3, we see that the model fits the data much better when $x_i \in [0, 20]$.

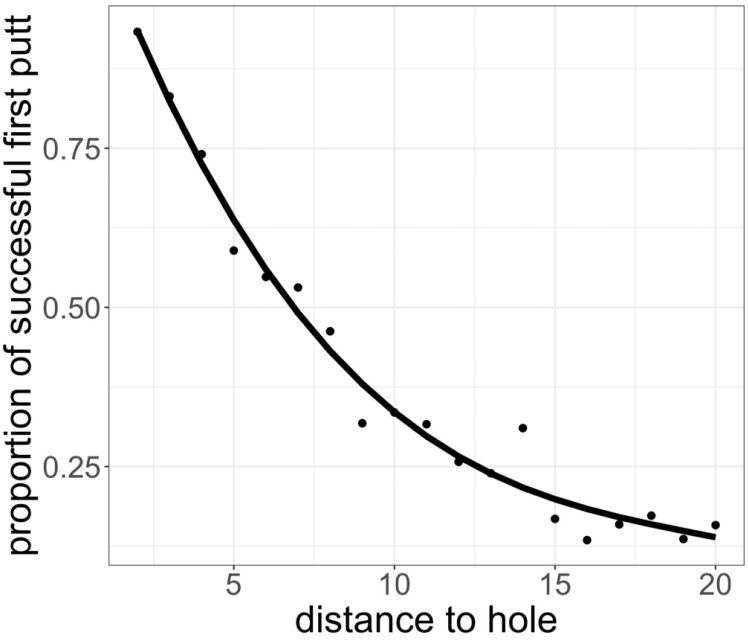

Figure 4.3: Putt data and fitted cubic regression

However, when we zoom out in Figure 4.4, we see that the model is unable to extrapolate when $x_i > 20$: in fact, the estimated probability of sinking the putt goes below 0.

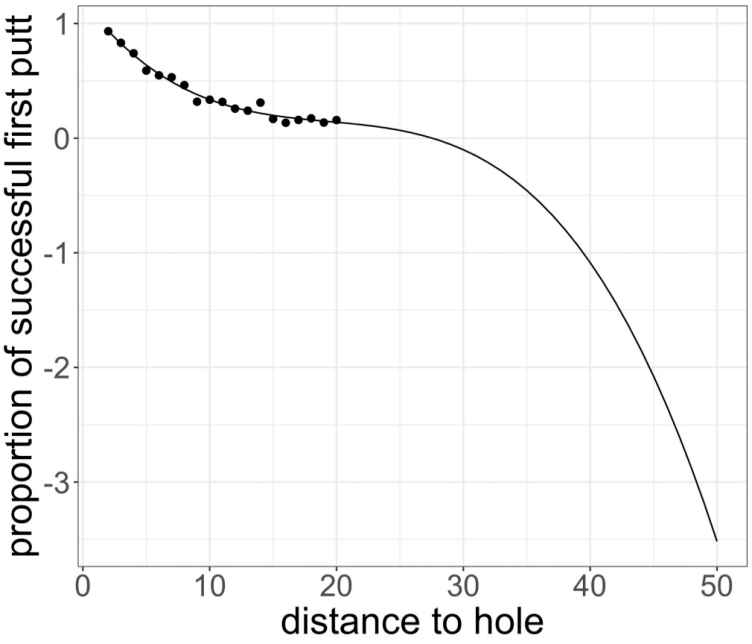

Figure 4.4: Putt data and fitted cubic regression (zoomed out)

This presents a problem: we need our predictions $\widehat{y}_i$ to be between 0 and 1, but ordinary linear regression does **not** guarantee this. How can we constrain our model to ensure that our predictions in $[0, 1]$?

4.2 Logistic Regression

4.2.1 The Logistic Function

We are looking for a "squishification" function that takes numbers in $\mathbb{R}$ and maps them to numbers in $[0, 1]$. One such function is the **logistic (or sigmoid) function**:

Definition 4.1 (Logistic Function)**.** *The* ***logistic function*** *is defined as:*

$$Logistic(z) = sigmoid(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

We plot the logistic function in Figure 4.5.

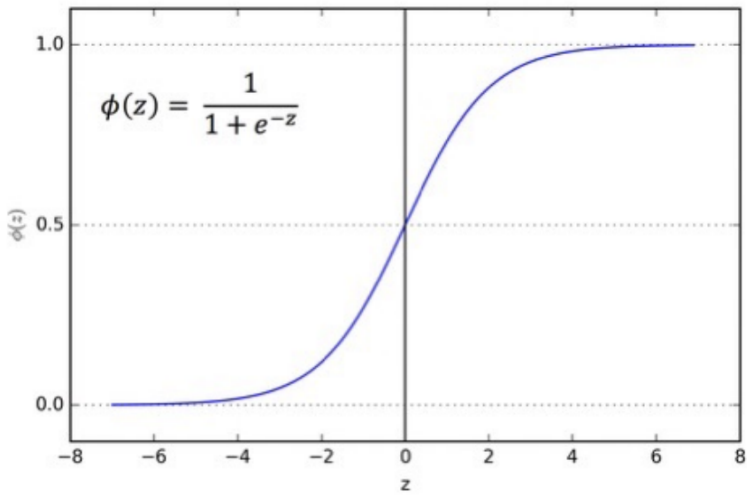

Figure 4.5: The Logistic Function

How can we implement this in our regression modeling?

4.2.2 The Model

Before, we had a model of the form:

$$\widehat{y}_i = \beta_0 + \beta_1 x_i$$

Now, we modify this model so that our predictions $\widehat{y}_i$ are between 0 and 1. We can do this by making our response variable $\widehat{y}_i$ the output of the logistic function:

$$\widehat{y}_i = \text{Logistic}(\beta_0 + \beta_1 x_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_i)}}$$

This is a **logistic regression model**, and allows us to model the **probability** that a putt is sunk directly. Mathematically, we can write this model as:

$$p_i = \mathbb{P}(y_i = 1 \mid x_i) = \frac{1}{1 + e^{-(x_i^T \boldsymbol{\beta})}}$$

$$\text{where } y_i \sim \text{Bernoulli}(p_i) = \begin{cases} 1 & \text{w.p. } p_i \\ 0 & \text{w.p. } 1 - p_i \end{cases}$$

4.2.3 Estimating the Coefficients

Our data is in terms of $y_i \in \{0, 1\}$ and $x_i \in \mathbb{R}$, not in terms of $p_i \in [0, 1]$. How can we then estimate the coefficients $\boldsymbol{\beta}$ in logistic regression?

In linear regression, we estimate $\boldsymbol{\beta}$ by minimizing the Residual Sum of Squares (RSS), or the squared error.

Definition 4.2 (Residual Sum of Squares)**.** *The* ***Residual Sum of Squares*** *(RSS) is defined as:*

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip}))^2$$

In logistic regression, we estimate $\boldsymbol{\beta}$ by minimizing the **log loss**, or the cross-entropy loss.

Definition 4.3 (Log Loss)**.** *The **log loss** is defined as:*

$$L(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log p_i + (1-y_i)\log(1-p_i)\right)$$
$$\text{where } p_i = \mathbb{P}(y_i = 1 \mid x_i, \boldsymbol{\beta}) = \frac{1}{1+e^{-(x_i^T\boldsymbol{\beta})}}$$

1. If $y_i = 1$, then $L(\boldsymbol{\beta}) = -\log p_i$
   - If $p_1 \approx 1$, then $\log p_i$ is close to 0 and $L(\boldsymbol{\beta})$ is low
   - If $p_1 \approx 0$, then $\log p_i$ is very negative and $L(\boldsymbol{\beta})$ is high
2. If $y_i = 0$, then $L(\boldsymbol{\beta}) = -\log(1-p_i)$
   - If $p_1 \approx 1$, then $\log(1-p_i)$ is very negative and $L(\boldsymbol{\beta})$ is high
   - If $p_1 \approx 0$, then $\log(1-p_i)$ is close to 0 and $L(\boldsymbol{\beta})$ is low

The log loss is a **convex** function, which means that it has a single global minimum. This means we can solve for $\boldsymbol{\beta}$ by setting its gradient equal to 0.

$$\begin{aligned}\nabla_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) &= -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \nabla_{\boldsymbol{\beta}} \log p_i + (1-y_i)\nabla_{\boldsymbol{\beta}} \log(1-p_i)\right)\\ &= 0\end{aligned}$$

Solving for $\boldsymbol{\beta}$ analytically can be difficult, so we use gradient descent in `R` to find the minimum iteratively.

4.3 Back to Sinking Putts

Recall the variables we have:

- i: index of the i^{th} putt in our dataset
- y_i: 1 if the i^{th} putt was sunk, 0 otherwise
- x_i: distance to the hole of the i^{th} putt

4.3.1 Fitting the Model

We use the `glm` function to fit the logistic regression model.

```
model = glm(putt_made ~ distance, data = putt_data, family = "binomial")
summary(model)
```

We plot the fitted line against the data in Figure 4.6. It seems to fit the data well, but we should check the model's performance from longer distances.

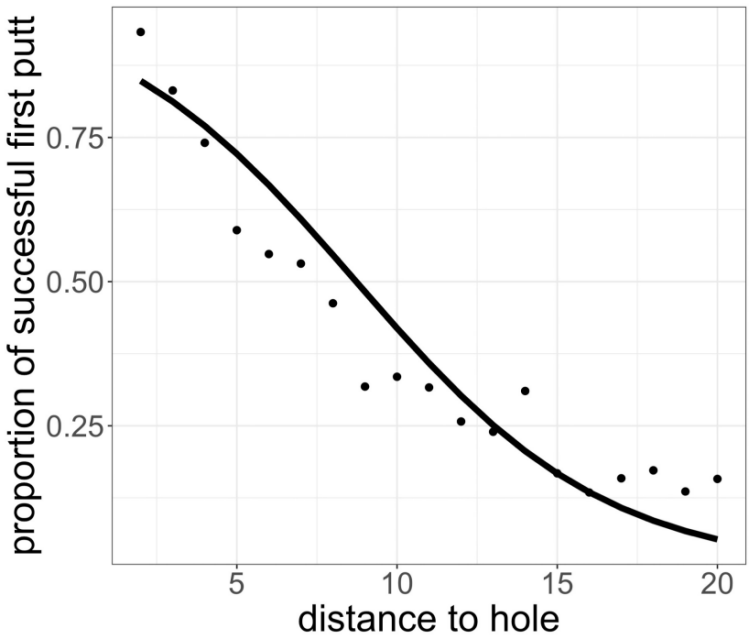

Figure 4.6: Putt data and fitted logistic regression

In Figure 4.7, we see that the model is able to extrapolate when $x_i > 20$ because we forced our output to be between 0 and 1.

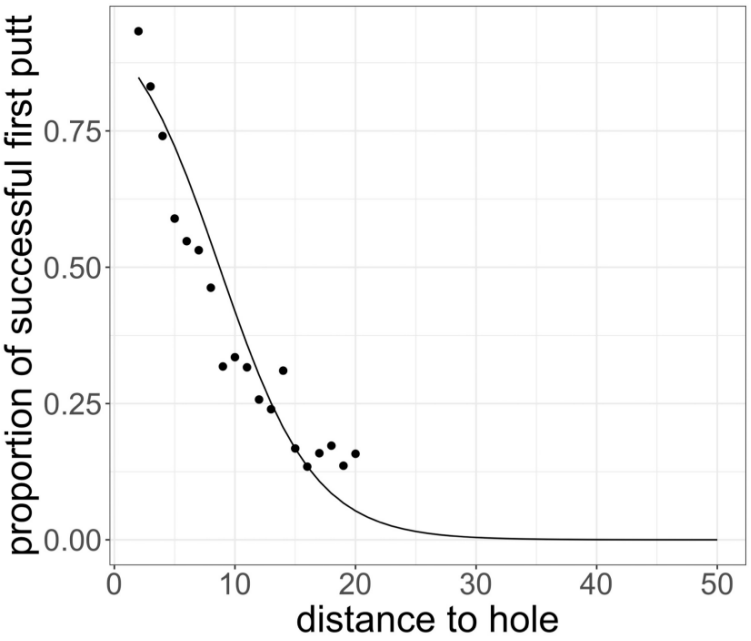

Figure 4.7: Putt data and fitted logistic regression (zoomed out)

4.3.2 Improving the Model

We can do even better by modeling the log odds with a cubic polynomial. Our new model is:

$$\mathbb{P}(y_i = 1 \mid x_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3)}}$$

We fit this model in `R` with the following code:

```
model = glm(putt_made ~ distance + I(distance^2) + I(distance^3),
            data = putt_data, family = "binomial")
summary(model)
```

We plot the fitted line against the data in Figure 4.8, and we can see that the model is able to extrapolate when $x_i > 20$.

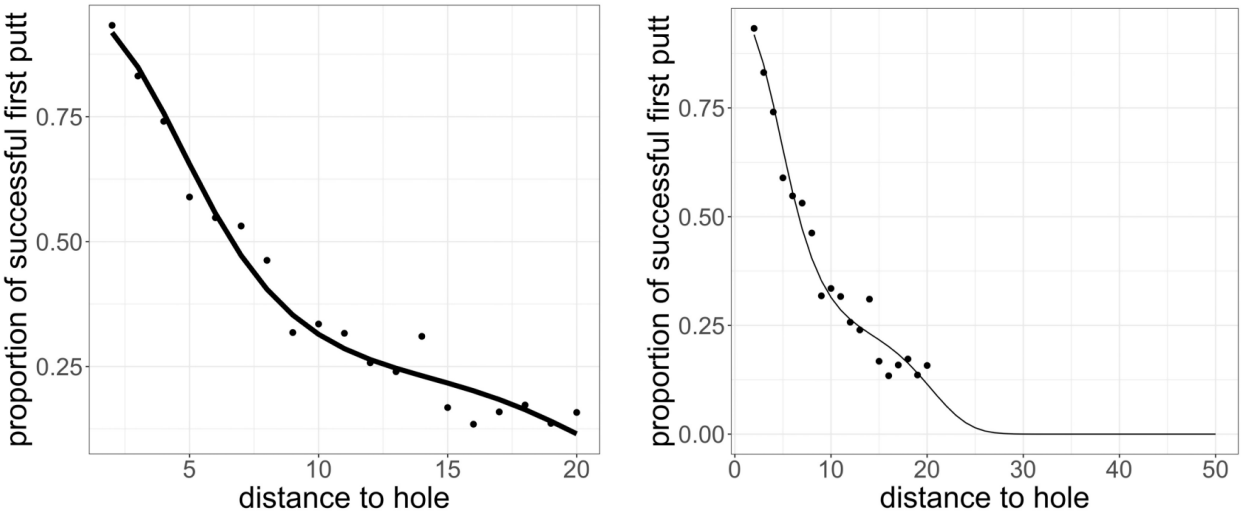

Figure 4.8: Putt data and fitted logistic regression with a cubic polynomial

4.3.3 Takeaways

What can we take away from this example?

- Use linear regression to predict a value that can take any real number
- Use logistic regression to predict a **probability** in $[0, 1]$
- In logistic regression, we interpret the coefficients as the degree to which our predictor variables impact the **log odds** of the response variable being 1.