Logistic Regression

Prof. Dennis Sun

Data 401

Classification Problems

So far, the problems we have considered all involved predicting a quantitative variable $Y$, like rating or income. These problems are called regression problems.

What if instead $Y$ is a categorical variable, like the type of beer or whether or not an applicant was accepted? These problems are called classification problems.
Binary Classification

We will start by looking at **binary classification**—when the categorical variable we are trying to predict takes on two possible values, which we can call 0 and 1.

What if we fit linear regression to this data set?

How do you interpret the values the line predicts?

What is wrong with this model?
Odds and Log-Odds

The problem is that the linear predictor, $\beta_0 + \beta_1 x$, assumes values between $-\infty$ and $\infty$, while the quantity we want to model, $P(Y = 1)$, assumes values between 0 and 1.

- If we instead model the odds, $\frac{P(Y=1)}{1-P(Y=1)}$, this assumes values between 0 and $\infty$.
- If we instead model the log-odds, $\log \frac{P(Y=1)}{1-P(Y=1)}$, this assumes values between $-\infty$ and $\infty$, which matches the linear predictor!

SOLUTION: Set the linear predictor to model the log-odds!

\[
\log \frac{P(Y_i = 1)}{1 - P(Y_i = 1)} = \beta_0 + \beta_1 x_i
\]

Solve for $P(Y_i = 1)$ in the above equation.
(Reminder: log is the natural logarithm.)

Hmm...does this function look familiar?

The Logistic Function
The Logistic Function

\[ P(Y_i = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_i)}} \]

Logistic Regression as a Neural Network

**Exercise:** How would you represent the logistic regression model

\[ \hat{p} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}} \]

as a neural network? What is the activation function?
To estimate the weights $\beta_0, \beta_1, \beta_2$, we need a loss function that compares $Y$ (0 or 1) to $\hat{p}$ (a probability).

- Mean-Squared Error: $L(Y, \hat{p}) = (\hat{p} - Y)^2$
- Log Loss (a.k.a. Negative Log-Likelihood):

$$L(Y, \hat{p}) = - \log [\hat{p}^Y (1 - \hat{p})^{1-Y}] = -Y \log \hat{p} - (1-Y) \log (1-\hat{p}).$$

We can now estimate $\beta_j$ by gradient descent: $\beta_j \leftarrow \beta_j - \eta \frac{\partial L}{\partial \beta_j}$.

$$\frac{\partial L}{\partial \beta_j} = \frac{\partial L}{\partial \hat{p}} \frac{\partial \hat{p}}{\partial \beta_j}$$

$$= \left( - \frac{Y}{\hat{p}} + \frac{1-Y}{1-\hat{p}} \right) \left( \frac{\hat{p}(1-\hat{p})}{g'} \cdot x_j \right)$$

$$= (\hat{p} - Y)x_j$$
Estimating Logistic Regression Coefficients

Logistic regression coefficients cannot be calculated analytically. They must be calculated using an iterative method, like gradient descent.

- Start with a random guess of $\beta_j$.
- Update each coefficient according to
  \[ \beta_j \leftarrow \beta_j + \eta(Y - \hat{p})x_j \]
  until convergence.

- **Intuition:** If we are underpredicting (so that $Y - \hat{p} > 0$), increase $\beta_j$ for positive $x_j$ (and decrease $\beta_j$ for negative $x_j$) to increase $\hat{p}$.
  Similarly, if we are overpredicting (so that $Y - \hat{p} < 0$), decrease $\beta_j$ for positive $x_j$ (and increase $\beta_j$ for negative $x_j$) to decrease $\hat{p}$.

What’s Next

We can generalize logistic regression to a neural network with hidden layers.

We will also see how to generalize it to multi-class classification.