Logistic Regression

Introduction to Statistical Learning Sections 4.3.1 and 4.3.3

Binary Classification Problems

Classification: Predicting whether an observation is in a certain category. That is, we are working with a categorical response variable.

Binary Classification: Only two possible categories

Eg. “Is it going to rain today?”

Binary Classification Problems

Just guessing “yes” or “no” is not necessarily the best approach, especially if there is no perfect rule.

A better approach would take noise into account, and not just give a binary answer.

What we want are probabilities, meaning we need to fit a stochastic model!

Binary Classification Problems

In probability lingo, we want to find Pr (Y | X), the conditional distribution of the response Y, given the input variables X.

Benefits: can tell how precise our predictions should be.

Eg. our model outputting a 51% chance of rain and it not raining, is better than outputting 99% chance of rain and it not raining.

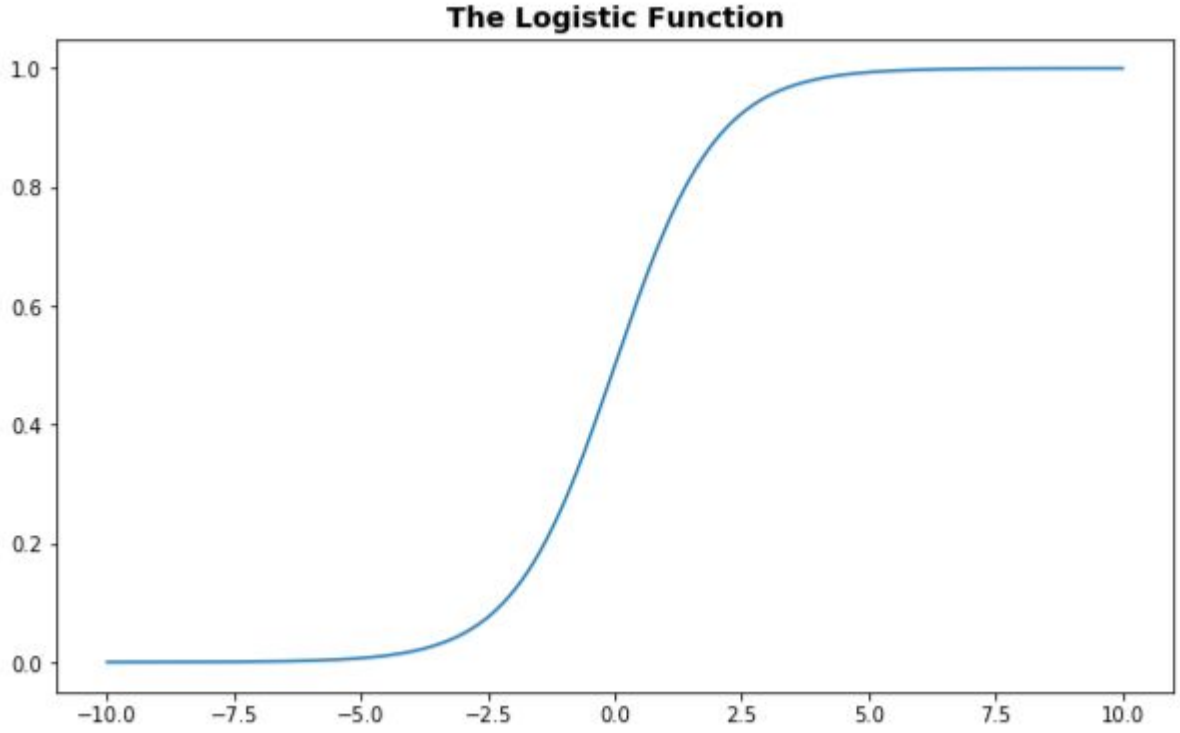
Idea 1: Use a Linear Model

$$P = \beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_k \cdot x_k$$

Here, we are saying that the probability (of True) is a linear function of the predictor variables.

Problems:

- Probabilities must be in [0,1], and there is no guarantee of that here
- We won't get "diminishing returns" — changing *P* by the same amount requires a bigger change in *x* when *P* is already large (or small) than when *P* is close to 1/2.


The Logistic Function
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

$$p(x) = \frac{1}{1 + e^{-x}}$$

Better Idea: Logistic Regression

Step 1:
Fit a linear function

$$y = \beta_0 + \beta \cdot \vec{x}$$

Step 2:
Squash the result into [0,1] by feeding the output into the logistic function

$$P(\vec{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot \vec{x})}}$$

Logistic Regression

$$P(\vec{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta \cdot \vec{x})}}$$

Now, we predict “True” when p >= 0.5 and “False” if p < 0.5

$$P(\vec{x}) \geq 0 \Rightarrow \beta_0 + \beta \cdot \vec{x} \geq 0$$

$$P(\vec{x}) < 0 \Rightarrow \beta_0 + \beta \cdot \vec{x} < 0$$

Logistic Regression

What is our decision boundary?

$$\beta_0 + \beta \cdot \vec{x} = 0$$

This determines a hyperplane in the predictor space (i.e., a point for one predictor, a line for 2 predictors, a plane for 3 predictors, etc.)

Also, the probability monotonically increases (or decreases) as points move farther away from this hyperplane

Warning: Math Incoming

Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

Logistic Regression

How do we determine the parameter values? By maximizing the "likelihood function".

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

If the observation is actually in class 1 (True/$y_i = 1$), then we get "credit" for the predicted probability of being in class 1.

Logistic Regression

How do we determine the parameter values? By maximizing the "likelihood function".

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}$$

If the observation is actually in class 0 (False/$y_i = 0$), then we get "credit" for the predicted probability of being in class 0 (which is 1 - probability assigned to class 1).

Logistic Regression

How do we determine the parameter values? By maximizing the “likelihood function”.

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

Overall, we do better by assigning high probabilities to the correct classes.

Logistic Regression

Let’s say we are trying to build a model to predict rain/no rain, and this is our training data (with some unspecified predictor variables).

Day	True Result
1	Rain
2	No Rain
3	No Rain
4	Rain
5	No Rain

We have two possible models we will chose from, whose predicted probabilities are shown in the next two slides.

Logistic Regression - Model A

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.78	0.22	Rain
2	No Rain	0.15	0.85	No Rain
3	No Rain	0.62	0.38	Rain
4	Rain	0.95	0.05	Rain
5	No Rain	0.25	0.75	No Rain

Accuracy = 4/5 = 80%

Logistic Regression - Model B

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.55	0.45	Rain
2	No Rain	0.48	0.52	No Rain
3	No Rain	0.45	0.55	No Rain
4	Rain	0.60	0.40	Rain
5	No Rain	0.39	0.61	No Rain

Accuracy = 5/5 = 100%

Evaluating Models

From the point of view of accuracy, Model B is superior.

However, this is not how logistic regression models are fit.

Logistic regression models maximize *likelihood*, which rewards high confidence when you're right, and lower confidence when your wrong.

It is better to be wrong sometimes, but highly confident when you're right than to be correct all of the time but with low confidence.

Logistic Regression

Recall that likelihood is calculated by multiplying together the predicted probabilities for the correct class.

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

Logistic Regression - Model A

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.78	0.22	Rain
2	No Rain	0.15	0.85	No Rain
3	No Rain	0.62	0.38	Rain
4	Rain	0.95	0.05	Rain
5	No Rain	0.25	0.75	No Rain

Likelihood = (0.78) * (0.85) * (0.38) * (0.95) * (0.75) = 0.180

Logistic Regression - Model B

Day	True Result	Predicted Probability of Rain	Predicted Probability of No Rain	Prediction
1	Rain	0.55	0.45	Rain
2	No Rain	0.48	0.52	No Rain
3	No Rain	0.45	0.55	No Rain
4	Rain	0.60	0.40	Rain
5	No Rain	0.39	0.61	No Rain

Likelihood = (0.55) * (0.52) * (0.55) * (0.60) * (0.61) = 0.058

Evaluating Models

This shows that model A is superior, when using the likelihood metric.

When fitting a logistic regression model, we are searching for the coefficient/weight values that produce the highest possible value for likelihood.

Logistic Regression - Regularization

We can also do constrain the coefficients of a logistic regression model using Ridge or LASSO regression. However, we'll maximize the *log* likelihood instead of the likelihood to formulate how this is done.

$$L(\vec{\beta}) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)))$$

Logistic Regression - Regularization

We can also do constrain the coefficients of a logistic regression model using Ridge or LASSO regression. However, we'll maximize the *log* likelihood instead of the likelihood to formulate how this is done.

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)))$$

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))) - \lambda \sum_{j=1}^{k} \beta_j^2$$

$$\log L(\vec{\beta}) = \sum_{i=1}^{n} (y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i))) - \lambda \sum_{j=1}^{k} |\beta_j|$$

Logistic Regression

When finding the parameter values, it is usually impossible to find the true optimal solution, so instead we must rely on a numerical approximation.

The default one used by scikit-learn is the L-BFGS (Low Memory, Broyden–Fletcher–Goldfarb–Shanno algorithm), but there are others.

This means that we will not necessarily find the optimal coefficients, but under most conditions will get an acceptable solution.