CS6301: R For Data Scientists

LECTURE 18: LOGISTIC REGRESSION

Problem Setup

We will once again consider a classification problem, where our response variable *Y* is now binary (let us assume $Y \in \{0,1\}$).

The technique we will present will work for continuous (numerical) or discrete (categorical) predictors

- Easier to visualize with continuous first

First, suppose that there are no predictors: Then *Y* is just a Bernoulli random variable, and we know $P(Y = 1) = p$ for some fixed probability *p.*

- We can approximate *p* by taking a sample, counting the number of ‘1’s, and dividing by the sample size – basic point estimation.

Problem Setup

Now suppose *p* changes, based upon some predictor variable *X*.

- Suppose *X* is just some scalar for now (i.e., only one predictor)

We want to find $P(Y = 1 \,|X = x)$, i.e. the probability that *Y* is ‘1’ for some given value of the predictor *X*.

- We want to approximate this as a function

Finding the Bayes Classifier

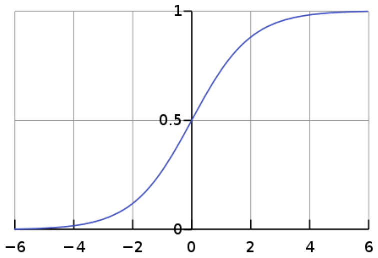
Recall the Bayes Classifier was given by $P(v_j \mid D)$, where v_j was a value our response variable took on and D was our training set.

So in our current setting, $P(Y = 1 \mid X = x)$ is exactly this classifier, except it is now a function of X.

We start off assuming a specific form; in the one dimensional case, it is given by

$$P(Y = 1 \mid X = x) = \frac{\exp(w_0 + w_1 x)}{1 + \exp(w_0 + w_1 x)}$$

Notice this curve looks like a "s-curve" or sigmoid function,
and may "curve up" or "down" depending on the signs of the weights



Finding the Weights

Given a training set $\{(x_i, y_i)\}$, how do we approximate the weights w_0, w_1?

We will use the Likelihood Function, which is given by

$$L(w_0, w_1) = \prod_i P(Y = y_i \mid X = x_i)$$

Note that (using the functional form for the conditional probabilities that we assumed on the last slide) this defines a function that in some sense measures how “likely” it is we got the training sample we did for particular values of w_0 and w_1.

Our goal is to find point estimators $\widehat{w}_0, \widehat{w}_1$ to the real values that maximize this likelihood – these are the **maximum likelihood estimators.**

Finding the Weights

How to do it: We want to find the weights that maximize the likelihood function

Take the partial derivatives and set them equal to zero!

Unfortunately, taking the derivatives of the product is messy …

Standard Trick: If we maximize the log of the function, we maximize the function …

… and the log turns the product into a sum!

This is usually done with a algorithm called “gradient ascent”

Let’s generalize …

Logistic Regression

Assume *Y* is a binary variable, taking on {0,1} values

Suppose our predictors $X = \{x_1, x_2, \dots, x_n\}$ are continuous (numeric)

We seek a functional approximation to the Bayes Classifier:

$$P(Y = 1\ |X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}$$

The estimators we find, $\widehat{w}_0, \widehat{w}_1, \dots, \widehat{w}_n$ are the maximum likelihood estimators, and they are typically found by gradient ascent

Once we have an approximation to the classifier, we can use it to make a prediction for a new input value: If the probability is greater than 0.5, predict a ‘1’, otherwise predict a ‘0’

Logistic Regression - Notes

Logistic Regression is still very popular, and frequently gives results comparable to more “sophisticated” models like Support Vector Machines and Neural Networks

The technique can be extended to handle situations where *Y* has more than two values

The technique can also be applied to categorical predictors, but they must be replaced with “dummy variables”

Because of the assumed form of the target function, the gradient ascent algorithm converges quickly

Logistic Regression is closely related to Gaussian Naïve Bayes

Measuring Model Fit - Classification

Precision – this is a measure of the false positives, i.e. Type I errors

Recall – this is a measure of false negatives, i.e. Type II errors

F-measure – combines both into one statistic:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Measuring Model Fit - Classification

	Actual		
Predicted		0	1
	0	n_1	n_2
	1	n_3	n_4

Precision = $\frac{n_4}{(n_3+n_4)}$

Recall = $\frac{n_4}{(n_4+n_2)}$

Odds, Log Odds/Logit

Notice that (unlike linear regression) it is generally hard to interpret the model here …

What impact does β_1 have on the probability that *Y = 1*?

We can analyze this by looking at the odds, which is basically the probability that *Y = 1* divided by the probability that it does not:

$$\frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)} = e^{\beta_0+\beta_1 x}$$

If we take the log of this, we get the *log odds* – notice, this is linear in the betas:

$$\log\left(\frac{P(Y = 1|X = x)}{1-P(Y = 1|X = x)}\right) = \beta_0 + \beta_1 x$$

Logistic Regression in R

R will produce a summary report for the logistic regression model, much like a linear regression model

The summary includes a table for betas, including standard error and p-value

Given new predictors, the model returns probabilities for *Y = 1*

How to use the model?

- Note that the prediction will give the conditional probability, not a Y value
- Can use the conditional probability to predict Y, say if probability > .5 or some other tolerance

Deviance

Deviance is a measure of how much the conditional probability we are trying to estimate improves with the model …

Recall we found our betas by maximizing the log likelihood function

Deviance compares the log likelihood of our model to the best possible model (called a *saturated* model) - a model that would fit the data perfectly

- The deviance is essentially the difference in the log likelihoods
- So smaller deviance is good, because it means our model is close to ideal

Measuring Model Fit in R

R will provide two deviances: Null and Residual

The Null Deviance is a measure of how well the Null Model (constant probability, no impact from X) will explain the data

- If this deviance is “small”, *X* has little impact on the probability of *Y = 1*
- It turns out this is a chi-squared variable with DoF *n – 1*

The Residual Deviance is a measure of how well the model you created explains the data

- Again, small is good
- This is a chi-squared variable with *n – p* DoF

If the Null Deviance large and Residual deviance is small, then our fitted model provides a substantial improvement over the Null (prior distribution) model

Questions …

Can we do Ridge/Lasso with logistic regression?

- Yes! The *glmnet()* function will accept “Binomial” as a distribution, meaning a binary response variable

What about using PCA to reduce dimensions?

- Yes! There is a new package (*logisticPCA*), or you can simply do PCA first and then use the transformed data in *glm*

Does this work with text data?

- Yes! Just create the DTM and use this as your input matrix. Note you need a classification response variable associated with each document

Issues: If the data separates cleanly, model may not converge. Also, hard to use if response has more than two categories.

Links

http://stats.stackexchange.com/questions/108995/interpreting-residual-and-null-deviance-in-glm-r

https://cran.r-project.org/web/packages/logisticPCA/vignettes/logisticPCA.html

http://cs229.stanford.edu/notes/cs229-notes1.pdf

http://amunategui.github.io/smote/