Lecture #12: More on Logistic Regression

CS 109A, STAT 121A, AC 209A: Data Science

Weiwei Pan, Pavlos Protopapas, **Kevin Rader**

Fall 2016

Harvard University

Announcements

- **Midterm**: will be graded this week. Results on Multiple Choice are promising :)
- Don't worry about some since Canvas isn't smart, unfortunately. Score could [only] increase.
- **Project**: Milestone #2 due on Wednesday, but will be open until Saturday. **Optional** to customize your project based on your requests.
- Expect an email from your project TF today!
- **Survey**: Results, we are incorporating suggestions, thank you for your feedback!
- **OHs** Update: Pavlos: Wed 4-5pm, Weiwei 5-6pm (updating on Canvas).
- **HW Grading**: A 4 is good ("A-")!

Quiz Time

Time for Quiz...password is **vicugnapaco**

Lecture Outline

Multiple Logistic Regression

Last time we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model. What was the model statement (in terms of linear predictors)?

Multiple Logistic Regression

Last time we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model. What was the model statement (in terms of linear predictors)?

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

Multiple Logistic Regression

Last time we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model. What was the model statement (in terms of linear predictors)?

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

Multiple logistic regression is a generalization to multiple predictors. More specifically we can define a multiple logistic regression model to predict $P(Y=1)$ as such:

$$\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p$$

where there are p predictors: $X = (X_1, X_2, ..., X_p)$.

Note: statisticians are often lazy and use the notation log to mean ln (the text does this). We will write $\log_{10}$ if this is what we mean.

Fitting Multiple Logistic Regression

The estimation procedure is identical to that as before for simple logistic regression: a likelihood approach is taken, and the function is maximized across all parameters $(\beta_0, \beta_1, ..., \beta_p)$ using an iterative method like Newton-Raphson.

The actual fitting of a Multiple Logistic Regression is easy using software (of course there's a python package for that) as the iterative maximization of the likelihood has already been hard coded.

In the `sklearn.linear_model` package, you just have to create your multidimensional X matrix to be used as predictors in the `LogisticRegression` function.

Interpretation of Multiple Logistic Regression

Interpreting the coefficients in a multiple logistic regression is similar to that of linear regression.

Key: since there are other predictors in the model, the coefficient $\hat{\beta}_j$ is the association between the j^{th} predictor and the response (on log odds scale). But we do we have to say?

Interpretation of Multiple Logistic Regression

Interpreting the coefficients in a multiple logistic regression is similar to that of linear regression.

Key: since there are other predictors in the model, the coefficient $\hat{\beta}_j$ is the association between the j^{th} predictor and the response (on log odds scale). But we do we have to say?

Controlling for the other predictors in the model.

We are trying to attribute the partial effects of each model controlling for the others (aka, controlling for possible *confounders*).

Interpreting Multiple Logistic Regression: an Example

Let's get back to the NFL data. We are attempting to predict whether a play results in a TD based on location (yard line) and whether the play was a pass. The simultaneous effect of these two predictors can be brought into one model.

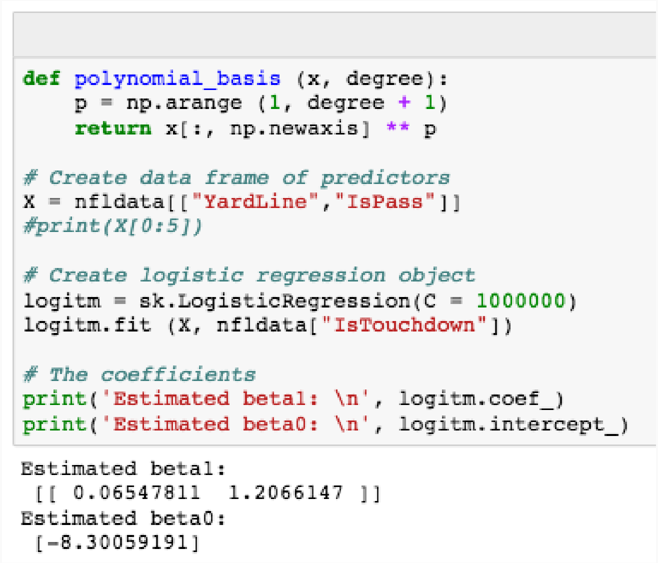
Recall from last time we had the following estimated models:

$$\log\left(\frac{\widehat{P(Y=1)}}{1-\widehat{P(Y=1)}}\right) = -7.425 + 0.0626 \cdot X_{yard}$$

$$\log\left(\frac{\widehat{P}(Y=1)}{1-\widehat{P(Y=1)}}\right) = -4.061 + 1.106 \cdot X_{pass}$$

The results for the multiple logistic regression model are on the next slide.

Interpreting Multiple Logistic Regression: an Example



```
def polynomial_basis (x, degree):
    p = np.arange (1, degree + 1)
    return x[:, np.newaxis] ** p

# Create data frame of predictors
X = nfldata[["YardLine","IsPass"]]
#print(X[0:5])

# Create logistic regression object
logitm = sk.LogisticRegression(C = 1000000)
logitm.fit (X, nfldata["IsTouchdown"])

# The coefficients
print('Estimated beta1: \n', logitm.coef_)
print('Estimated beta0: \n', logitm.intercept_)
```

```
Estimated beta1:
 [[ 0.06547811  1.2066147 ]]
Estimated beta0:
 [-8.30059191]
```

Some questions

1. Write down the complete model. Break this down into the model to predict log-odds of a touchdown based on the yard line for passes and the same model for non-passes. How is this different from the previous model (without interaction)?
2. Estimate the odds ratio of a TD comparing passes to non-passes.
3. Is there any evidence of multicollinearity in this model?
4. Is there any confounding in this problem?

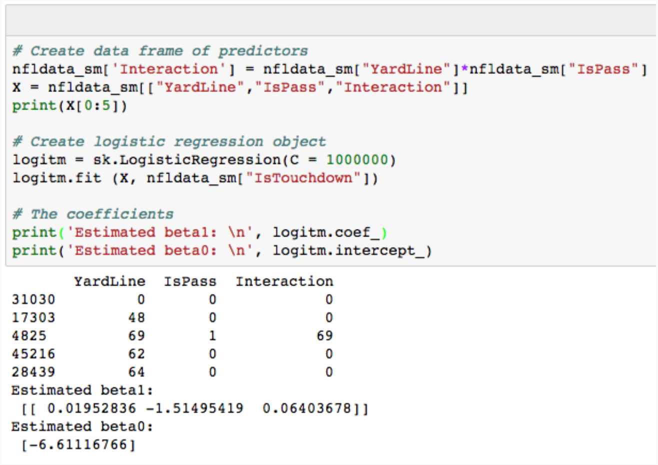
Interactions in Multiple Logistic Regression

Just like in linear regression, interaction terms can be considered in logistic regression.

An interaction terms is incorporated into the model the same way, and the interpretation is very similar (on the log-odds scale of the response of course).

Write down the model for the NFL data for the 2 predictors plus the interactions term.

Interpreting Multiple Logistic Regression with Interaction: an Example



```
# Create data frame of predictors
nfldata_sm['Interaction'] = nfldata_sm["YardLine"]*nfldata_sm["IsPass"]
X = nfldata_sm[["YardLine","IsPass","Interaction"]]
print(X[0:5])

# Create logistic regression object
logitm = sk.LogisticRegression(C = 1000000)
logitm.fit (X, nfldata_sm["IsTouchdown"])

# The coefficients
print('Estimated beta1: \n', logitm.coef_)
print('Estimated beta0: \n', logitm.intercept_)
```

```
       YardLine  IsPass  Interaction
31030         0       0            0
17303        48       0            0
4825         69       1           69
45216        62       0            0
28439        64       0            0
Estimated beta1:
 [[ 0.01952836 -1.51495419  0.06403678]]
Estimated beta0:
 [-6.61116766]
```

Some questions

1. Write down the complete model. Break this down into the model to predict log-odds of a touchdown based on the yard line for passes and the same model for non-passes. How is this different from the previous model (without interaction)?
2. Use this model to estimate the probability of a touchdown for a pass at the 20 yard line. Do the same for a run at the 20 yard line.
3. Use this model to estimate the probability of a touchdown for a pass at the 99 yard line. Do the same for a run at the 99 yard line.
4. Is this a stronger model than the previous one? How would we check?

Lecture Outline

Multiple Logistic Regression

Regularization in Logistic Regression

Classification Boundaries in Logistic Regression

Multinomial Logistic Regression

Regularization in Linear Regression

Based on the Likelihood framework, a loss function can be determined based on the likelihood function.

We saw in linear regression that maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

Regularization in Linear Regression

Based on the Likelihood framework, a loss function can be determined based on the likelihood function.

We saw in linear regression that maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

$$\arg\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \arg\min \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi}))^2$$

And a regularization approach was to add a penalty factor to this equation. Which for Ridge Regression becomes:

Regularization in Linear Regression

Based on the Likelihood framework, a loss function can be determined based on the likelihood function.

We saw in linear regression that maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

$$\arg\min \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \arg\min \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi}))^2$$

And a regularization approach was to add a penalty factor to this equation. Which for Ridge Regression becomes:

$$\arg\min \left[\sum_{i=1}^{n} \left(y_i - \left(\beta_0 + \sum_{j=1}^{n} \beta_j x_{ji} \right) \right)^2 + \lambda^2 \sum_{j=1}^{n} \beta_j \right]$$

This penalty *shrinks* the estimates towards zero, and had the analogue of using a Normal prior in the Bayesian paradigm.

Loss function in Logistic Regression

A similar approach can be used in logistic regression. Here, maximizing the log-likelihood is equivalent to minimizing the following loss function:

$$\arg\min \left[-\sum_{i=1}^{n} \left(y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i \right) \right]$$

where $\hat{p}_i = \frac{\exp(\beta_0 + \sum_{j=1}^{n} \beta_j x_{ji})}{1 + \exp(\beta_0 + \sum_{j=1}^{n} \beta_j x_{ji})}$.

Why is this a good loss function to minimize? Where does this come from?

Loss function in Logistic Regression

A similar approach can be used in logistic regression. Here, maximizing the log-likelihood is equivalent to minimizing the following loss function:

$$\arg\min\left[-\sum_{i=1}^{n}\left(y_i\log(\hat{p}_i)+(1-y_i)\log(1-\hat{p}_i)\right)\right]$$

where $\hat{p}_i=\frac{\exp(\beta_0+\sum_{j=1}^{n}\beta_j x_{ji})}{1+\exp(\beta_0+\sum_{j=1}^{n}\beta_j x_{ji})}$.

Why is this a good loss function to minimize? Where does this come from?

The log-likelihood for independent $Y_i \sim \mathrm{Bern}\,(p_i)$:

Regularization in Logistic Regression

A penalty factor can then be added to this loss function and results in a new loss function that penalizes large values of the parameters:

$$\arg\min\left[-\sum_{i=1}^{n}\left[y_i\log(\hat{p}_i)+(1-y_i)\log(1-\hat{p}_i)\right]+\lambda^2\sum_{j=1}^{n}\beta_j\right]$$

The result is just like in linear regression: shrinkage towards zero of the parameters.

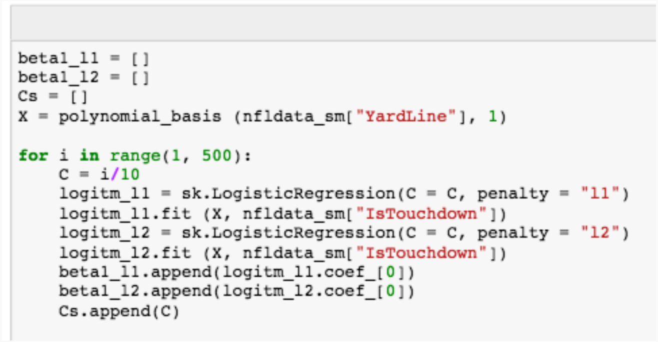
In practice, the intercept is usually not part of the penalty factor, and is thus not shrunk towards zero.

Note: the `sklearn` package uses a different tuning parameter: instead of λ they use a constant that is essentially $C = 1/\lambda$.

Regularization in Logistic Regression: an Example

Let's see how this plays out in an example in logistic regression.

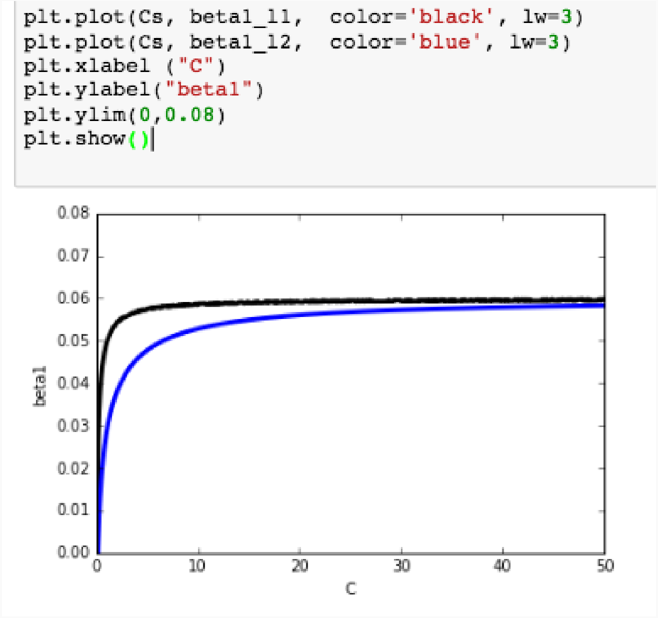
Regularization in Logistic Regression: an Example



```
betal_l1 = []
betal_l2 = []
Cs = []
X = polynomial_basis (nfldata_sm["YardLine"], 1)

for i in range(1, 500):
    C = i/10
    logitm_l1 = sk.LogisticRegression(C = C, penalty = "l1")
    logitm_l1.fit (X, nfldata_sm["IsTouchdown"])
    logitm_l2 = sk.LogisticRegression(C = C, penalty = "l2")
    logitm_l2.fit (X, nfldata_sm["IsTouchdown"])
    betal_l1.append(logitm_l1.coef_[0])
    betal_l2.append(logitm_l2.coef_[0])
    Cs.append(C)
```

Regularization in Logistic Regression: an Example



```
plt.plot(Cs, beta1_l1,  color='black', lw=3)
plt.plot(Cs, beta1_l2,  color='blue', lw=3)
plt.xlabel ("C")
plt.ylabel("beta1")
plt.ylim(0,0.08)
plt.show()
```

Regularization in Logistic Regression: an Example

Just like in linear regression, the shrinkage factor must be chosen.
How should we go about doing this?

Regularization in Logistic Regression: an Example

Just like in linear regression, the shrinkage factor must be chosen. How should we go about doing this?

Through building multiple training and test sets (through k-fold or random subsets), we can select the best shrinkage factor to mimic out-of-sample prediction.

How could we measure how well each model fits the test set? We could measure this based on the proposed loss function!

Lecture Outline

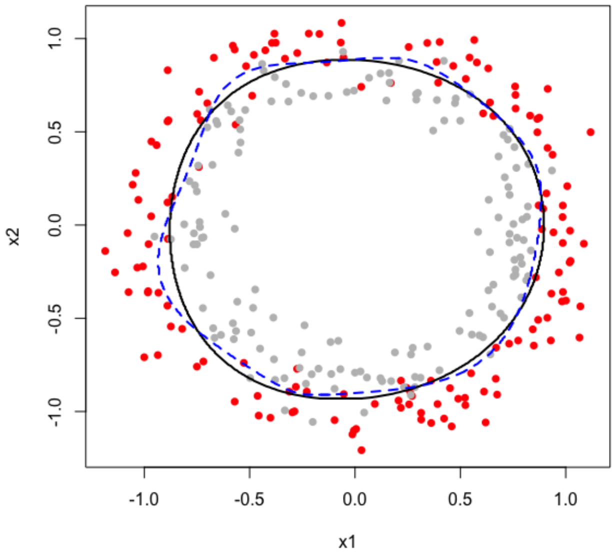
Classification

Recall from last time that we could attempt to purely classify each observation based on whether the estimated $P(Y = 1)$ from the model was greater than 0.5.

When dealing with 'well-separated' data, logistic regression can work well in performing classification.

We saw a 2-D plot last time which had two predictors, X_1 and X_2 and depicted the classes as different colors. A similar one is shown on the next slide.

2D Classification in Logistic Regression: an Example



2D Classification in Logistic Regression: an Example

Would a logistic regression model perform well in classifying the observations in this example?

2D Classification in Logistic Regression: an Example

Would a logistic regression model perform well in classifying the observations in this example?

What would be a good logistic regression model to classify these points?

2D Classification in Logistic Regression: an Example

Would a logistic regression model perform well in classifying the observations in this example?

What would be a good logistic regression model to classify these points?

Based on these predictors, two separate logistic regression model were considered that were based on different ordered polynomials of X_1 and X_2 and their interactions. The ‘circles’ represent the classification boundary for classification.

How can the classification boundary be calculated for a logistic regression?

2D Classification in Logistic Regression: an Example

In the previous plot, which classification boundary performs better? How can you tell? How would you make this determination in an actual data example?

2D Classification in Logistic Regression: an Example

In the previous plot, which classification boundary performs better? How can you tell? How would you make this determination in an actual data example?

We could determine the misclassification rates

Lecture Outline

Logistic Regression for predicting more than 2 Classes

There are several extensions to standard logistic regression when the response variable Y has more than 2 categories. The two most common are ordinal logistic regression and multinomial logistic regression. Ordinal logistic regression is used when the categories have a specific hierarchy (like class year: Freshman, Sophomore, Junior, Senior; or a 7-point rating scale from strongly disagree to strongly agree). Multinomial logistic regression is used when the categories have no inherent order (like eye color: blue, green, brown, hazel, et...).

Multinomial Logistic Regression

The most common approach to estimating a nominal (not-ordinal) categorical variable that has more than 2 classes. The first approach sets one of the categories in the response variable as the *reference* group, and then fits separate logistic regression models to predict the other cases based off of the reference group. For example we could attempt to predict a student's concentration:

$$y = \begin{cases} 1 & \textit{if } \text{Computer Science (CS)} \\ 2 & \textit{if } \text{Statistics} \\ 3 & \text{otherwise} \end{cases}.$$

from predictors x_1 number of psets per week and x_2 how much time spent in Lamont Library.

Multinomial Logistic Regression (cont.)

We could select the $y = 3$ case as the reference group (other concentration), and then fit two separate models: a model to predict $y = 1$ (CS) from $y = 3$ (others) and a separate model to predict $y = 2$ (Stat) from $y = 3$ (others).

Ignoring interactions, how many parameters would need to be estimated?

How could these models be used to estimate the probability of an individual falling in each concentration?

One vs. Rest (ovr) Logistic Regression (cont.)

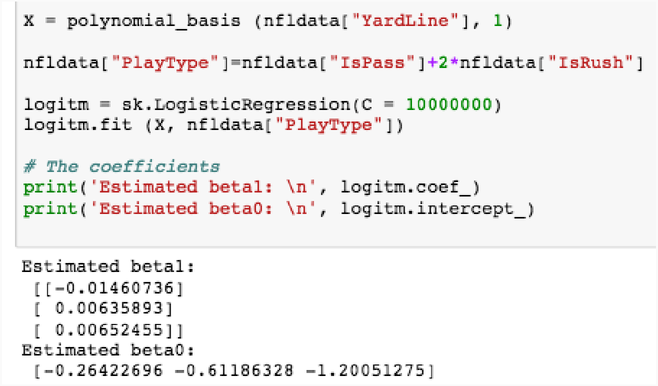
The default multiclass logistic regression model is called the 'One vs. Rest' approach.

If there are 3 classes, then 3 separate logistic regression are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

1. a first model would be fit to predict CS from (Stat and Others) combined
2. a second model would be fit to predict Stat from (CS and Others) combined
3. a third model would be fit to predict Others from (CS and Stat) combined

An example to predict play call from the NFL data follows...

OVR Logistic Regression in Python



```
X = polynomial_basis (nfldata["YardLine"], 1)

nfldata["PlayType"]=nfldata["IsPass"]+2*nfldata["IsRush"]

logitm = sk.LogisticRegression(C = 10000000)
logitm.fit (X, nfldata["PlayType"])

# The coefficients
print('Estimated beta1: \n', logitm.coef_)
print('Estimated beta0: \n', logitm.intercept_)
```

```
Estimated beta1:
 [[-0.01460736]
 [ 0.00635893]
 [ 0.00652455]]
Estimated beta0:
 [-0.26422696 -0.61186328 -1.20051275]
```

Classification for more than 2 Categories

When there are more than 2 categories in the response variable, then there is no guarantee that $P(Y = k) \geq 0.5$ for any one category. So any classifier based on logistic regression will instead have to .

The classification boundaries are then much more difficult to determine. We will not get into the algorithm for drawing these in this class.