Logistic Regression: Behind the Scenes

Chris White

Capital One

October 9, 2016

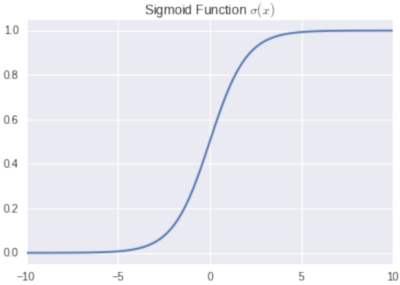
Generative Model

$$y_i|\beta, x_i \sim \text{Bernoulli}\big(\sigma(\beta, x_i)\big)$$

where

$$\sigma(\beta, x) := \frac{1}{1 + \exp\left(-\beta \cdot x\right)}$$

is the *sigmoid* function.



Interpretation

- This setup implies that the *log-odds* are a *linear* function of the inputs

$$\log\left(\frac{\mathbb{P}[y=1]}{\mathbb{P}[y=0]}\right) = \beta \cdot x$$

Interpretation

- This setup implies that the *log-odds* are a *linear* function of the inputs

$$\log\left(\frac{\mathbb{P}[y=1]}{\mathbb{P}[y=0]}\right) = \beta \cdot x$$

- This linear relationship makes the individual coefficients β_k easy to interpret: a unit increase in x_k increases the odds of $y = 1$ by a factor of β_k (all else held equal)

Interpretation

- This setup implies that the *log-odds* are a *linear* function of the inputs

$$\log\left(\frac{\mathbb{P}[y=1]}{\mathbb{P}[y=0]}\right) = \beta \cdot x$$

- This linear relationship makes the individual coefficients β_k easy to interpret: a unit increase in x_k increases the odds of $y = 1$ by a factor of β_k (all else held equal)

Question: Given data, how do we determine the "best" values for β?

Typical Answer

Maximum Likelihood Estimation

$$\beta^* = \arg\max_{\beta} \prod_i \sigma\left(\beta \cdot x_i\right)^{y_i} \left(1 - \sigma\left(\beta \cdot x_i\right)\right)^{1-y_i}$$

Typical Answer

Maximum Likelihood Estimation

$$\beta^* = \arg\max_{\beta} \prod_i \sigma\left(\beta \cdot x_i\right)^{y_i} \left(1 - \sigma\left(\beta \cdot x_i\right)\right)^{1-y_i}$$

- Log-likelihood is *concave*

Typical Answer

Maximum Likelihood Estimation

$$\beta^* = \arg\max_{\beta} \prod_i \sigma\left(\beta \cdot x_i\right)^{y_i} \left(1 - \sigma\left(\beta \cdot x_i\right)\right)^{1-y_i}$$

- Log-likelihood is *concave*
- Maximum Likelihood Estimation allows us to do classical statistics (i.e., we know the asymptotic distribution of the estimator)

Typical Answer

Maximum Likelihood Estimation

$$\beta^* = \arg\max_{\beta} \prod_i \sigma\left(\beta \cdot x_i\right)^{y_i} \left(1 - \sigma\left(\beta \cdot x_i\right)\right)^{1-y_i}$$

- Log-likelihood is *concave*
- Maximum Likelihood Estimation allows us to do classical statistics (i.e., we know the asymptotic distribution of the estimator)
- **Note:** *the likelihood is a function of the predicted probabilities and the observed response*

This is not the end of the story

Now we are confronted with an optimization problem...

This is not the end of the story

Now we are confronted with an optimization problem...

The story doesn't end here!

Uncritically throwing your data into an off-the-shelf solver could result in a bad model.

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
    - Coefficient accuracy vs. speed vs. predictive accuracy
    - Multicollinearity leads to both statistical and numerical issues

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
  - Coefficient accuracy vs. speed vs. predictive accuracy
  - Multicollinearity leads to both statistical and numerical issues
- Desired Input (e.g., missing values?)

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
  - Coefficient accuracy vs. speed vs. predictive accuracy
  - Multicollinearity leads to both statistical and numerical issues
- Desired Input (e.g., missing values?)
- Desired Output (e.g., p-values)

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
  - Coefficient accuracy vs. speed vs. predictive accuracy
  - Multicollinearity leads to both statistical and numerical issues
- Desired Input (e.g., missing values?)
- Desired Output (e.g., p-values)
- Handling of edge cases (e.g., quasi-separable data)

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
  - Coefficient accuracy vs. speed vs. predictive accuracy
  - Multicollinearity leads to both statistical and numerical issues
- Desired Input (e.g., missing values?)
- Desired Output (e.g., p-values)
- Handling of edge cases (e.g., quasi-separable data)
- Regularization and Bayesian Inference

Important Decisions

- Inference vs. Prediction: your goals should influence your implementation choices
  - Coefficient accuracy vs. speed vs. predictive accuracy
  - Multicollinearity leads to both statistical and numerical issues
- Desired Input (e.g., missing values?)
- Desired Output (e.g., p-values)
- Handling of edge cases (e.g., quasi-separable data)
- Regularization and Bayesian Inference

Goal

We want to explore these questions with an eye on statsmodels, scikit-learn and SAS's proc logistic.

Prediction vs. Inference

Coefficients and Convergence

In the case of inference, coefficient sign and precision are important.

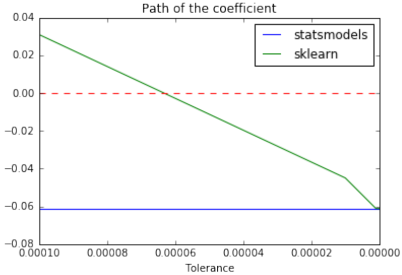
Example: A model includes economic variables, and will be used for ‘stress-testing‘ adverse economic scenarios.

Prediction vs. Inference

Coefficients and Convergence

In the case of inference, coefficient sign and precision are important.

Example: A model includes economic variables, and will be used for ‘stress-testing‘ adverse economic scenarios.



Prediction vs. Inference, cont'd

Coefficients and Multicollinearity

- Linear models are invariant under affine transformations of the data; this leads to a determination problem in the presence of "high" multicollinearity
- Multicollinearity can threaten long term model stability
- Can test for multicollinearity with condition number or Variance Inflation Factors (VIF)

Default Convergence Criteria

Tool	Algorithm	Convergence	Default Tol
SAS proc logistic	IRWLS	Relative Gradient	10^{-8}
scikit-learn	Coordinate Ascent	Log-Likelihood	10^{-4}
statsmodels Logit	Newton	ℓ_∞	10^{-8}
statsmodels GLM	IRWLS	Deviance	10^{-8}

Default Convergence Criteria

Tool	Algorithm	Convergence	Default Tol
SAS proc logistic	IRWLS	Relative Gradient	10^{-8}
scikit-learn	Coordinate Ascent	Log-Likelihood	10^{-4}
statsmodels Logit	Newton	ℓ_∞	10^{-8}
statsmodels GLM	IRWLS	Deviance	10^{-8}

Note

Convergence criteria based on log-likelihood convergence emphasize *prediction stability*.

Behavior under different implementations

Especially in edge cases (e.g. quasi-separability, high multicollinearity) slightly different implementations can lead to drastically different outputs.

At github.com/moody-marlin/pydata_logistic you can find a notebook with various implementations of Newton's method for logistic regression, with explanations.

Input

Input

Missing value handling

- SAS and statsmodels skip incomplete observations in build, scikit-learn throws an error

Input

Missing value handling

- SAS and statsmodels skip incomplete observations in build, scikit-learn throws an error

Categorical input variables

- Packages which require numerical arrays (scikit-learn, ‘base‘ statsmodels) require the modeler to dummify
- SAS and statsmodels + formulas can handle them by identifier

Input

Missing value handling

- SAS and statsmodels skip incomplete observations in build, scikit-learn throws an error

Categorical input variables

- Packages which require numerical arrays (scikit-learn, ‘base‘ statsmodels) require the modeler to dummify
- SAS and statsmodels + formulas can handle them by identifier

Scoring conventions

- scikit-learn and ‘base’ statsmodels will score by *location*
- SAS and statsmodels + formula can score by name

Output

Do you have access to necessary output?

Output

Do you have access to necessary output?

- Fit metrics (AIC, BIC, Adj-R^2, etc.)
- Convergence flags
- p-values

Output

Do you have access to necessary output?

- Fit metrics (AIC, BIC, Adj-R^2, etc.)
- Convergence flags
- p-values

...and if not, can you compute it? For example, sklearn *always* uses regularization, computing p-values yourself will be incorrect!

Rant on *p*-values

Imagine a misspecified ordinary least squares model in which the true specification takes the form $y \sim \mathcal{N}(f(x), \sigma^2)$ for some *non-linear* $f(x)$.

Rant on p-values

Imagine a misspecified ordinary least squares model in which the true specification takes the form $y \sim \mathcal{N}(f(x), \sigma^2)$ for some *non-linear* $f(x)$.

The MLE is given by

$$\beta^* = x^T y \|x\|_2^{-2}$$

and converges to its expectation in the limit $n \to \infty$, i.e.,

$$\beta^* \to \mathbb{E}_x \left[x^T \vec{f}(x) \|x\|_2^{-2} \right]$$

Rant on p-values

Imagine a misspecified ordinary least squares model in which the true specification takes the form $y \sim \mathcal{N}(f(x), \sigma^2)$ for some *non-linear* $f(x)$.

The MLE is given by

$$\beta^* = x^T y \|x\|_2^{-2}$$

and converges to its expectation in the limit $n \to \infty$, i.e.,

$$\beta^* \to \mathbb{E}_x \left[x^T \vec{f}(x) \|x\|_2^{-2} \right]$$

Conclusion

Note that $\lim_{n\to\infty} \beta^* > 0$ if and only if $\mathbb{E}_x \left[x^T \vec{f}(x) \right] > 0$.

Rant on p-values

Consequently, in most situations the p-value for x will converge to 0!

Rant on p-values

Consequently, in most situations the p-value for x will converge to 0!

Takeaway

p-values and Big Data are not friends!

Regularization

Regularization is the process of "penalizing" candidate coefficients for undesirable complexity, i.e.,

$$\beta^* = \arg\max_{\beta} \mathcal{L}\left(x, y, \beta\right) - \tau g(\beta)$$

where $g(\beta)$ is some application-dependent measure of "complexity", and $\tau > 0$.

Regularization, cont'd

Popular examples of regularizations include:

- $g(\beta) = \|\beta\|_1$

Regularization, cont'd

Popular examples of regularizations include:

- $g(\beta) = \|\beta\|_1$
  - Arises as the convex relaxation of best subset problem
  - Results in *sparse* coefficients

Regularization, cont'd

Popular examples of regularizations include:

- $g(\beta) = \|\beta\|_1$
  - Arises as the convex relaxation of best subset problem
  - Results in *sparse* coefficients
- $g(\beta) = \|\beta\|_2^2$

Regularization, cont'd

Popular examples of regularizations include:

- $g(\beta) = \|\beta\|_1$
  - Arises as the convex relaxation of best subset problem
  - Results in *sparse* coefficients

- $g(\beta) = \|\beta\|_2^2$
  - Results in "ridged" coefficients
  - Historically used for inversion of poorly conditioned matrices

Regularization, cont'd

Popular examples of regularizations include:

- $g(\beta) = \|\beta\|_1$
    - Arises as the convex relaxation of best subset problem
    - Results in *sparse* coefficients

- $g(\beta) = \|\beta\|_2^2$
    - Results in "ridged" coefficients
    - Historically used for inversion of poorly conditioned matrices

Be aware

Regularization means your estimated coefficients are *no longer* maximum likelihood estimators, so be careful in your interpretation and inference!

Caution!

Many optimization options subtly result in regularization!

- the ‘FIRTH’ option in SAS (and ‘logistf’ in R) arise from setting $g(\beta) = \log\left(\det \mathcal{I}(\beta)\right)$ and $\tau = \frac{1}{2}$, where $\mathcal{I}(\beta)$ is the Fisher information matrix

Caution!

Many optimization options subtly result in regularization!

- the 'FIRTH' option in SAS (and 'logistf' in R) arise from setting $g(\beta) = \log\left(\det \mathcal{I}(\beta)\right)$ and $\tau = \frac{1}{2}$, where $\mathcal{I}(\beta)$ is the Fisher information matrix
- providing a "ridge_factor" value in statsmodels is equivalent to providing τ^{-1} with $g(\beta) = \|\beta\|_2^2$

Caution!

Many optimization options subtly result in regularization!

- the 'FIRTH' option in SAS (and 'logistf' in R) arise from setting $g(\beta) = \log(\det \mathcal{I}(\beta))$ and $\tau = \frac{1}{2}$, where $\mathcal{I}(\beta)$ is the Fisher information matrix
- providing a "ridge_factor" value in statsmodels is equivalent to providing τ^{-1} with $g(\beta) = \|\beta\|_2^2$
- sklearn *always* regularizes

Statistical Implications: Bayesian Interpretation

Any regularization function for which

$$\int_{\mathbb{R}^k} \exp\left(-g(\beta)\right) d\beta < \infty$$

results in a penalized log-likelihood with a Bayesian interpretation.

The regularized coefficients are maximum a posteriori (MAP) estimators for a Bayesian model with prior distribution given by

$$\exp\left(-g(\beta)\right)$$

.

Conclusion

Takeaway

Even if you don't run into any edge cases, understanding what's happening under the hood gives you access to *much more* information that you can leverage to build better, more informative models!

The End

Thanks for your time and attention! Questions?