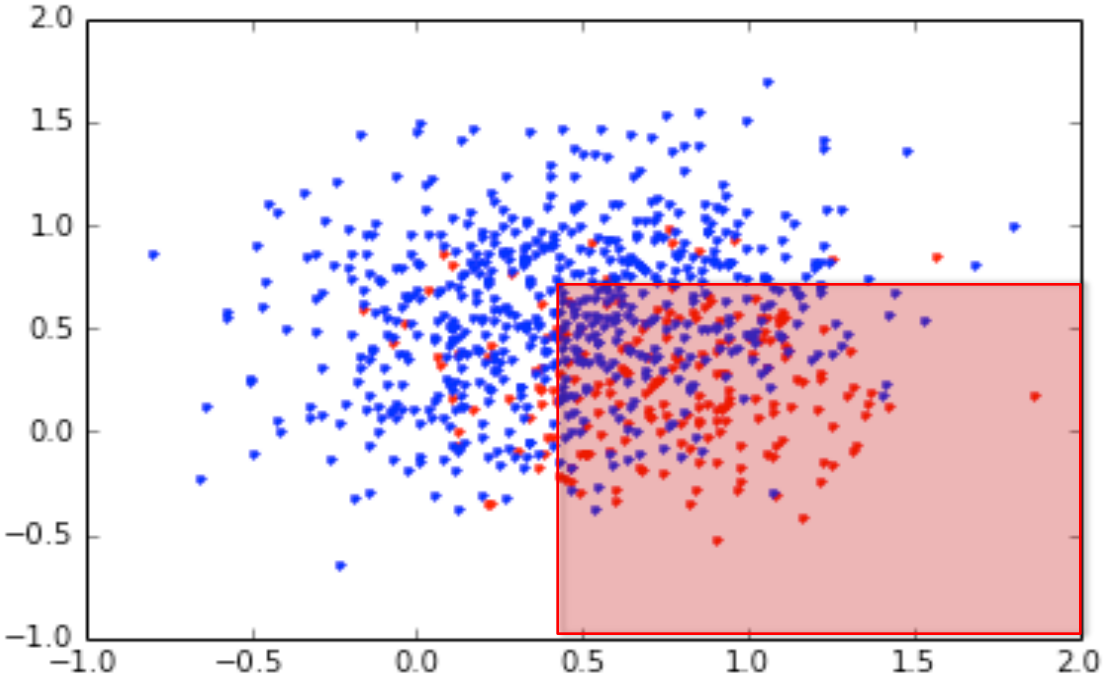
Introduction to Data Science

LOGISTIC REGRESSION

BRIAN D'ALESSANDRO

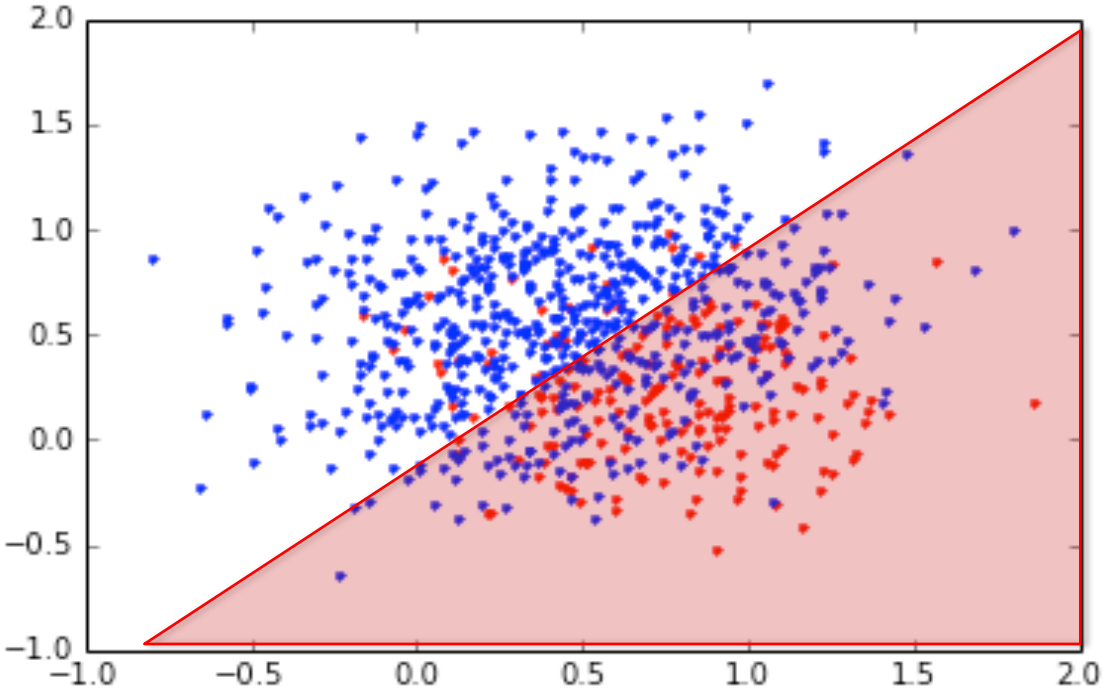
REVIEW – HOW TO CLASSIFY

Decision Trees use a greedy algorithm to minimize the entropy.
It does this by using Boolean operators to partition the feature space.



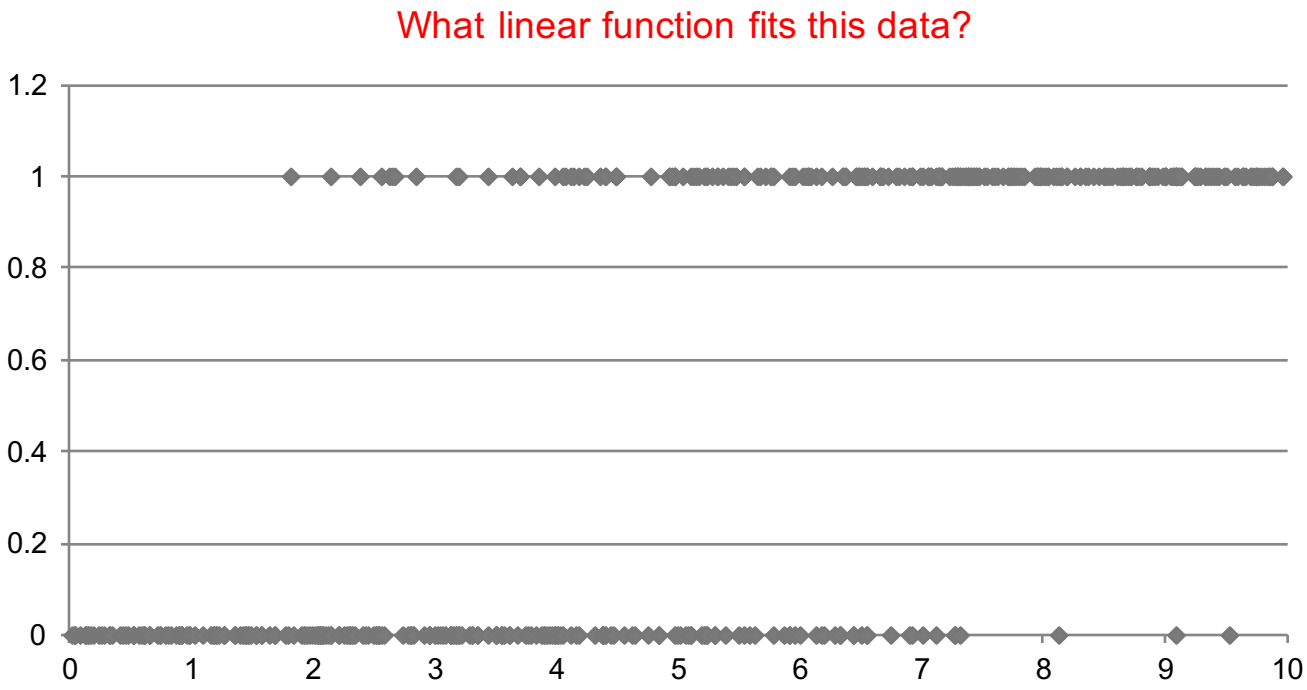
LINEAR MODELS

Linear models also partition the feature space, but do so by finding an optimal linear separating hyper-plane, which is found by using principles of statistical learning theory.



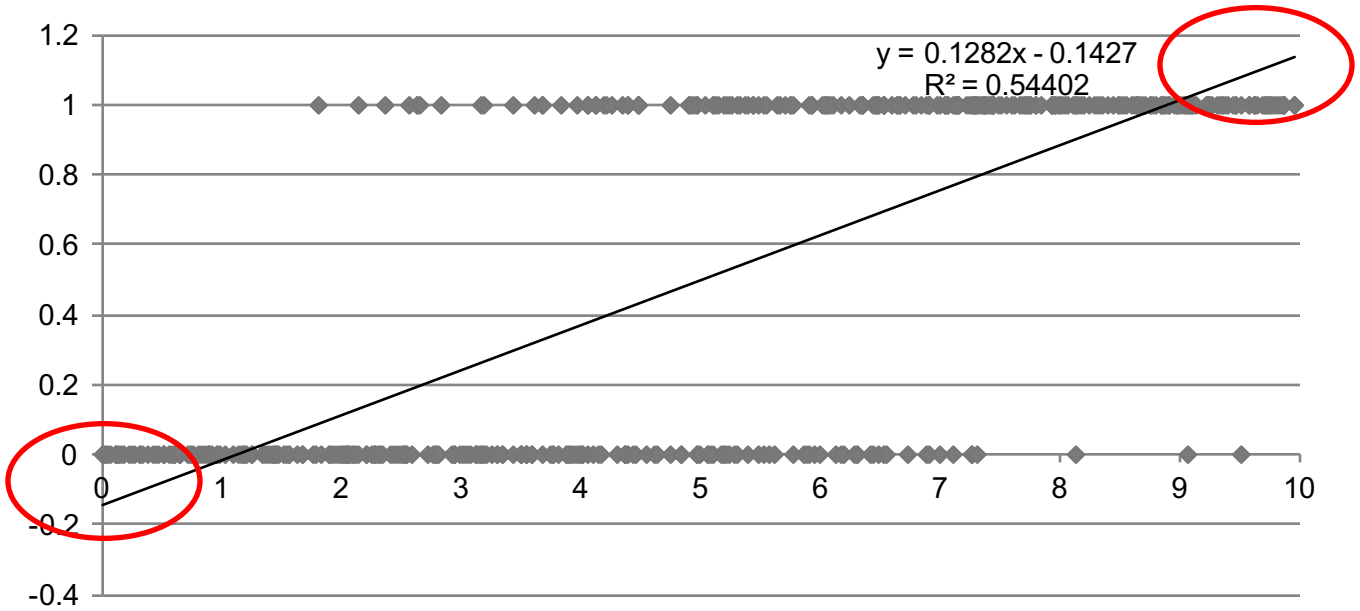
A BINARY REGRESSION?

Logistic Regression was created as a binary extension of Ordinary Least Squares Regression.



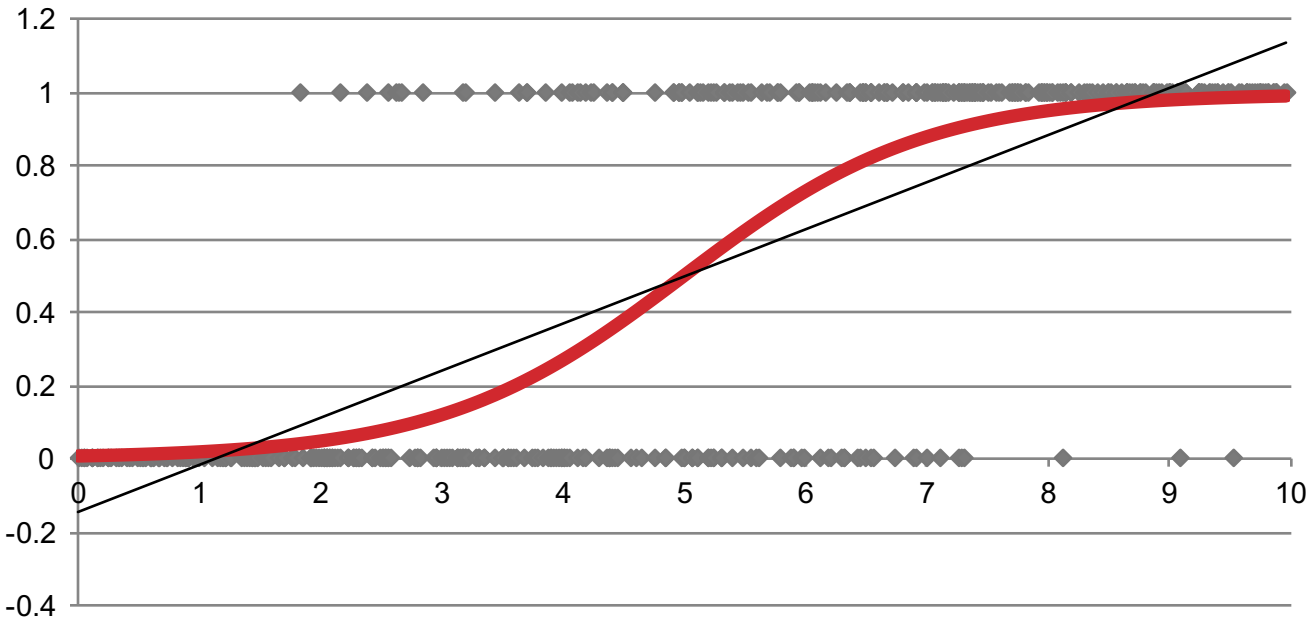
LINEAR LEAST SQUARES – NOT SO GOOD

The linear least squares curve gives us a reasonable fit in certain areas, but is not constrained to the interval [0,1]. This is bad when you want to estimate a probability.



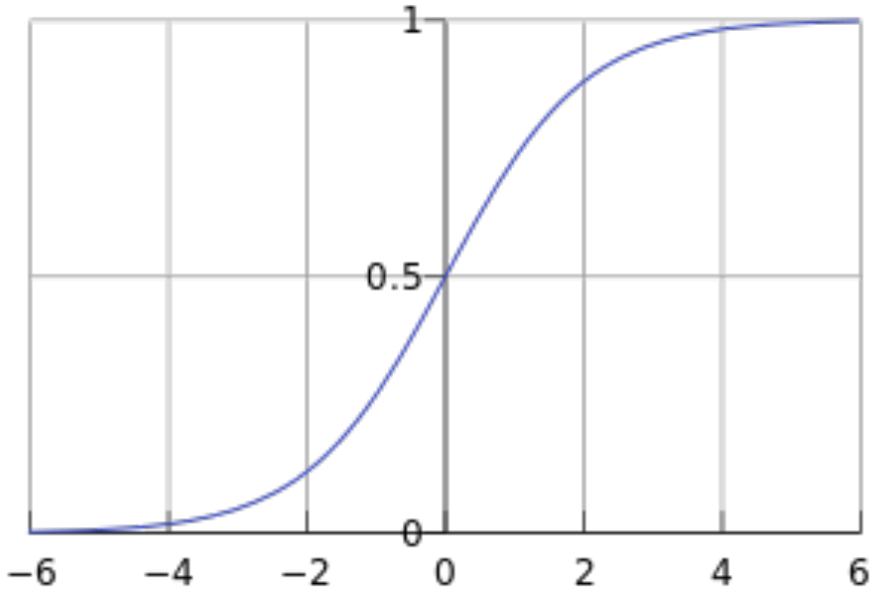
SOMETHING BETTER?

It would be better if we had some function that was linear in its parameters, but behaved better as a probability estimator.



THE INVERSE LOGIT

The inverse logit is just the function we are looking for.



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

LOGISTIC REGRESSION

Logistic Regression: a member of the class of generalized linear models (glm) using the logit as its link function.

The goal of Logistic Regression is to model the posterior probability of membership in class c_i as a function of X. I.e.,

$$P(c_i|x) = f(x) = \frac{1}{1+e^{-(\alpha+\beta x))}}$$

To make this a linear model in X, we take the log of the odds ratio of p (called the log-odds):

$$ln\ \frac{P(c_i|x)}{1-P(c_i|x)} = ln\ \frac{1}{e^{-(\alpha+\beta x))}} = \alpha + \beta x$$

And effectively we do a linear regression against the log-odds of $P(c_i|x)$ (though we don't use least squares).

LOGISTIC REGRESSION AS ERM

How do we fit Logistic Regression into the ERM framework?

We find the parameters α and β using the method of Maximum Likelihood Estimation.

If we consider each observation to be an indepedent Bernoulli draw with $p_i = P(y_i|x_i)$, then the likelihood of each draw can be defined as: $p_i^{y_i}(1-p_i)^{1-y_i}$, with p_i given by the inverse logit function. In MLE, we wish to maximize the likelihood of observing the data as a function of the independent parameters of the model (i.e., α and β). The total likelihood function looks like:

$$L(\alpha, \beta|X, Y) = \prod_{i=1}^{n} P(x_i, y_i|\alpha, \beta) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

This is actually a difficult equation to maximize directly, so we do a little trick. We take the negative log and call this our loss function for ERM!

$$\mathbb{L}(f(X), Y) = -Ln[L(\alpha, \beta|X, Y)] = -\sum_{i=1}^{n} y_i \, ln\,(p_i) + (1-y_i)\, ln\,(1-p_i)$$

LR: STATS VS. MACHINE LEARNING?

MLE is a method used traditionally in statistics while ERM is used in machine learning. Logistic Regression works in both disciplines. An advantage of "statistical" point of view is that we can do hypothesis testing on the parameter estimates of the model.

	coef	std err	z	P>|z|	[95.0% Conf. Int.]
isbuyer	0.8421	0.562	1.499	0.134	-0.259 1.943
buy_freq	0.0588	0.397	0.148	0.882	-0.720 0.838
visit_freq	0.0469	0.026	1.828	0.067	-0.003 0.097
buy_interval	0.0320	0.020	1.591	0.112	-0.007 0.071
sv_interval	-0.0047	0.010	-0.488	0.626	-0.024 0.014
expected_time_buy	-0.0337	0.025	-1.372	0.170	-0.082 0.014
expected_time_visit	-0.0241	0.009	-2.801	0.005	-0.041 -0.007
last_buy	0.0038	0.006	0.634	0.526	-0.008 0.016
last_visit	-0.0518	0.006	-8.636	0.000	-0.064 -0.040
multiple_buy	-0.6713	1.099	-0.611	0.541	-2.825 1.482
multiple_visit	0.0493	0.278	0.177	0.859	-0.495 0.593
uniq_urls	-0.0107	0.002	-5.147	0.000	-0.015 -0.007
num_checkins	-6.112e-05	0.000	-0.481	0.631	-0.000 0.000

Note: I used pythons statsmodel package as opposed to scikit-learn to get this summary. See accompanying ipython notebook for examples.

Model Statistical Analysis

coef – the estimate for β

std err – the standard error for the estimate of β

z – the z-score for hypothesis testing on the estimate of β

P>|z| – the p-value (prob of type 1 error) for asserting that β!= 0.

[95% Conf Int] – the 95% conf. interval for the estimate of β

INTERPRETING BETAS

A Practical Aside

What exactly does the estimate of β really mean? How can we interpret it?

Recall that $Ln\ \frac{p}{1-p} = \alpha + \beta x$. This means that a unit change in the value of x changes the log-odds by the value of β. This is a mathematical statement that IMHO does not offer much intuitive value.

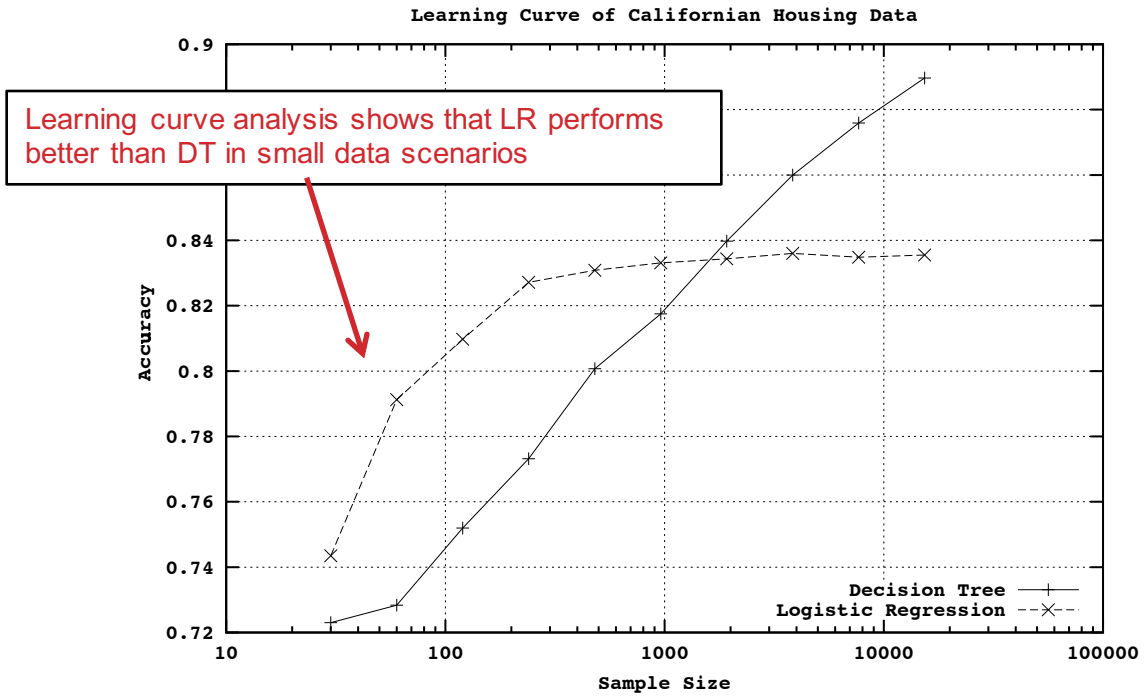
So what can we learn by looking at betas? (IMHO, not much!)

Some helpful tips, garnered from theory and experience:

- $|\beta_1| > |\beta_2|$ does not guarantee that feature X_1 is more predictive than X_2. The magnitude of β is inversely proportional to the scale of X, so comparing betas only makes sense when the features have the same scale (such as binary features).
- Likewise, the z-score of β is influences by sample size and should not be used to rank features by predictiveness
- sign(β) does tell you whether Y is positively or negatively correlated with X. However, if the features have a lot of multi-collinearity, sign(β) can be misleading.
- Multi-collinearity in X means the betas will have covariance with each other. The betas will "split" the effect. Sometimes they'll split the effect as positive numbers (i.e. 1=0.5+0.5) and other times they'll split as negatives (i.e., 1=2-1). This makes interpreting β that much more difficult.

ROBUSTNESS OF LR

Not all scenarios involve “Big Data.” Logistic Regression has been proven over and over to be very robust in small data problems.



Source: Tree Induction vs. Logistic Regression, a Learning Curve Analysis. Perlich, Provost and Simonoff