BIOINF 585: Machine Learning for Systems Biology & Clinical Informatics

Lecture 10: Logistic Regression

Jie Wang
Department of Computational Medicine & Bioinformatics
University of Michigan

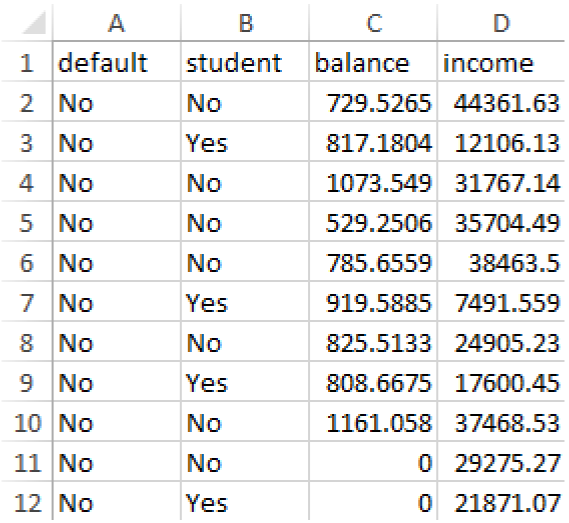
Outline

- An Example
- Logistic Regression with One Predictor
- Multiple Logistic Regression
- Sparse Logistic Regression
- Tips for real applications
  - One vs All for multi-class classification
  - Imbalanced Data

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

An Example

- Predict if a individual will default based on annual income, monthly balance, and status (student?).



	A	B	C	D
1	default	student	balance	income
2	No	No	729.5265	44361.63
3	No	Yes	817.1804	12106.13
4	No	No	1073.549	31767.14
5	No	No	529.2506	35704.49
6	No	No	785.6559	38463.5
7	No	Yes	919.5885	7491.559
8	No	No	825.5133	24905.23
9	No	Yes	808.6675	17600.45
10	No	No	1161.058	37468.53
11	No	No	0	29275.27
12	No	Yes	0	21871.07

An subset of the default data set, which contains three predictors with a binary response.

- Notice that, the response is not continuous, and it takes *binary* values (Yes or No).
- This is a classification problem.

Why Not Linear Regression I

- For linear regression, the outcome is usually *continuous* (there is an order in the possible values of the outcome).
- For classification problem, the outcome takes *discrete/categorical* values, where order is meaningless.

An Example: Suppose you are trying to guess the identity of a superhero based on his/her special talents. He or she can be the Captain America, The Hulk, Black Widow, or Thor. How can we encode the outcome?

$$Y = \begin{cases} 1 & \text{Captain America} \\ 2 & \text{The Hulk} \\ 3 & \text{Black Widow} \\ 4 & \text{Thor} \end{cases}$$

$$Y = \begin{cases} 1 & \text{The Hulk} \\ 2 & \text{Black Widow} \\ 3 & \text{Thor} \\ 4 & \text{Captain America} \end{cases}$$

Which one is more reasonable?

Why Not Linear Regression II

- If we apply linear regression to the classification problems, different encodings of the outcome would produce fundamentally different linear models leading to *different* predictions for the same test observations.
- We do not have general principles to convert discrete outcomes with more than two possible levels to continuous ones, which are suitable for linear regression.
- A good news is that, if the outcome is binary, we can code it simply as

$$\boxed{Y = \begin{cases} 0 & \text{Captain America} \\ 1 & \text{Thor} \end{cases}}$$

Logistic Regression with Binary Outcome

- Consider the example in which the outcome Y is binary: default or not default.
- As Y is binary, we can code it by

$$Y = \begin{cases} -1 & \text{if default = no} \\ 1 & \text{If default = yes} \end{cases} \quad \text{or} \quad Y = \begin{cases} 0 & \text{if default = no} \\ 1 & \text{If default = yes} \end{cases}$$

Notational convenient for later use

- Different from linear regression, logistic regression models the *probability* of $Y = 1$ and $Y = -1$.
- For example, if balance is given, logistic regression models

$$P(\text{default} = \text{yes}|\text{balance})$$

$$P(\text{default} = \text{no}|\text{balance}) = 1 - P(\text{default} = \text{yes}|\text{balance})$$

Simple Logistic Regression

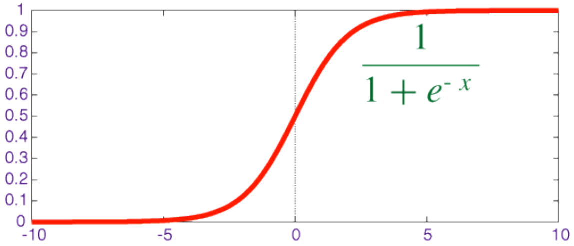
- For simplicity, we only consider one predictor: balance.
- As $P(\text{default} = \text{yes}|\text{balance}) \in [0,1]$, we make use of the *logistic function* to model the probability.

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

a simplified form of $P(Y = 1|X)$

the predictor

- An illustration of the logistic function.



The figure is from http://artint.info/figures/ch07/sigmoidc.gif

Odds and Log-odds

- The so-called odds is defined by

the probability that the outcome is Yes

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

the probability that the outcome is No

Odds is popular in gambling, e.g., horse-racing.

- The log-odds is defined by the log of the odds

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

Log-odds is linear regarding to the predictor, although $P(X)$ is not.

Estimating the Regression Coefficients

- We estimate the regression coefficients by *maximum likelihood*.
- Intuitively, the idea of *maximum likelihood* is to find a set of coefficients such that the probability of the observations is maximized (under the assumption that the observations are drawn *independently*).
- For our example, *maximum likelihood* means that we try to find $\widehat{\beta_0}$ and $\widehat{\beta_1}$ such that $P(X)$ *is close to one for all individuals who default* and $P(X)$ *is close to zero for all individuals who do not default*.

Maximum Likelihood Estimate I

- Given a set of observations $\{(x_i, y_i): i = 1, \cdots, n\}$, the likelihood function is

$$
\begin{aligned}
\ell(\beta_0, \beta_1) &= \prod_{i:y_i=1} p(x_i) \prod_{j:y_j=-1} \left(1 - p(x_j)\right) \\
&= \prod_{i:y_i=1} \frac{1}{1 + e^{-(\beta_0+\beta_1 x_i)}} \prod_{j:y_j=-1} \left(1 - \frac{1}{1 + e^{-(\beta_0+\beta_1 x_j)}}\right) \\
&= \prod_{i:y_i=1} \frac{1}{1 + e^{-(\beta_0+\beta_1 x_i)}} \prod_{j:y_j=-1} \left(\frac{1}{1 + e^{(\beta_0+\beta_1 x_i)}}\right) \\
&= \prod_i \frac{1}{1 + e^{-(y_i(\beta_0+\beta_1 x_i))}}
\end{aligned}
$$

Do NOT confuse with the probability function

Maximum Likelihood Estimate II

- Maximum likelihood aims to maximize the likelihood function.

$$\max_{\beta_0,\beta_1}\left\{\ell(\beta_0,\beta_1)=\prod_i \frac{1}{1+e^{-\left(y_i(\beta_0+\beta_1 x_i)\right)}}\right\}$$

- The maximization problem is equivalent to a minimization problem (we first convert the product to summation by Logarithm).

$$\boxed{\min_{\beta_0,\beta_1}\left\{-\log\ell(\beta_0,\beta_1)=\sum_i \log\left(1+e^{-\left(y_i(\beta_0+\beta_1 x_i)\right)}\right)\right\}}$$

Gradient Descent

- Let us denote

$$(\widehat{\beta_0}, \widehat{\beta_1}) = \underset{\beta_0,\beta_1}{\operatorname{argmin}} \left\{ -\log \ell(\beta_0, \beta_1) = \sum_i \log\left(1 + e^{-\left(y_i(\beta_0+\beta_1 x_i)\right)}\right) \right\}$$

- There is no closed form solution of $\left(\widehat{\beta_0}, \widehat{\beta_1}\right)$.
- One of the simplest method is to iteratively decrease the objective function value $L(\beta_0, \beta_1) = -\log \ell(\beta_0, \beta_1)$ by gradient descent:

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \nabla_{\beta_0} L\left(\beta_0^{(k)}, \beta_1^{(k)}\right)$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \nabla_{\beta_1} L\left(\beta_0^{(k)}, \beta_1^{(k)}\right)$$

η is the so-called learning rate

- For properly chosen learning rate η (e.g., back-line tracking), the sequence $L\left(\beta_0^{(k)}, \beta_1^{(k)}\right), k = 1,2,\cdots$ will converge to $L(\widehat{\beta_0}, \widehat{\beta_1})$, and the following holds

$$L\left(\beta_0^{(k+1)}, \beta_1^{(k+1)}\right) \leq L\left(\beta_0^{(k)}, \beta_1^{(k)}\right)$$

Batch Training

- We write the updates of (β_0, β_1) explicitly by

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \sum_i \frac{-y_i e^{-\left(y_i\left(\beta_0^{(k)}+\beta_1^{(k)}x_i\right)\right)}}{1 + e^{-\left(y_i\left(\beta_0^{(k)}+\beta_1^{(k)}x_i\right)\right)}}$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \sum_i \frac{-y_i x_i e^{-\left(y_i\left(\beta_0^{(k)}+\beta_1^{(k)}x_i\right)\right)}}{1 + e^{-\left(y_i\left(\beta_0^{(k)}+\beta_1^{(k)}x_i\right)\right)}}$$

- The above updates need to pass through *all* of the training data. Thus, it is called *batch* update.
- If the training data has millions or billions (very common in real applications) of data samples, the updates can be very time-consuming.

Online Training

- Different from batch training (that involves all the data samples in each iteration), online training update the parameters by visiting one data sample in each iteration.
- Let $f_i(\beta_0, \beta_1) = \log\left(1 + e^{-\left(y_i(\beta_0+\beta_1 x_i)\right)}\right)$. Then,

$$\left(\widehat{\beta_0}, \widehat{\beta_1}\right) = \underset{\beta_0,\beta_1}{\operatorname{argmin}} \left\{ -\log \ell(\beta_0, \beta_1) = \sum_i f_i(\beta_0, \beta_1) \right\}$$

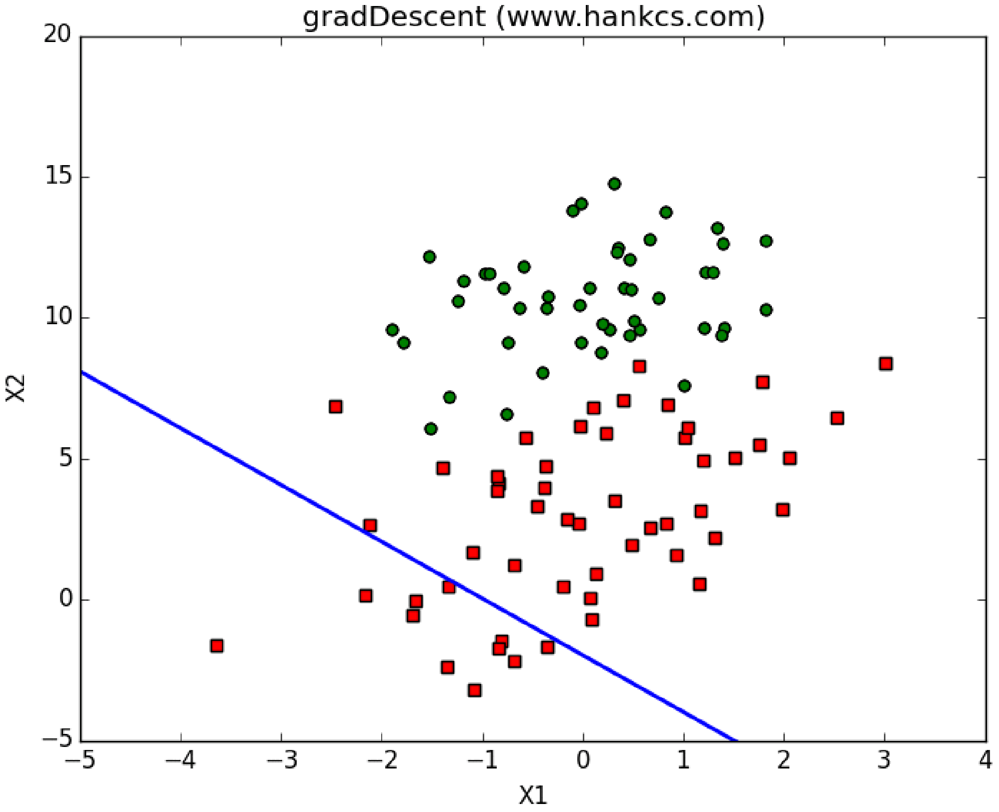
Stochastic gradient descent

- For online training, in each iteration, the updates are

$$\beta_0^{(k+1)} = \beta_0^{(k)} - \eta \nabla_{\beta_0} f_k\left(\beta_0^{(k)}, \beta_1^{(k)}\right) = \beta_0^{(k)} - \eta \frac{-y_k e^{-\left(y_k\left(\beta_0^{(k)}+\beta_1^{(k)} x_k\right)\right)}}{1 + e^{-\left(y_k\left(\beta_0^{(k)}+\beta_1^{(k)} x_k\right)\right)}}$$

$$\beta_1^{(k+1)} = \beta_1^{(k)} - \eta \nabla_{\beta_1} f_k\left(\beta_0^{(k)}, \beta_1^{(k)}\right) = \beta_1^{(k)} - \eta \frac{-y_k x_k e^{-\left(y_k\left(\beta_0^{(k)}+\beta_1^{(k)} x_k\right)\right)}}{1 + e^{-\left(y_k\left(\beta_0^{(k)}+\beta_1^{(k)} x_k\right)\right)}}$$

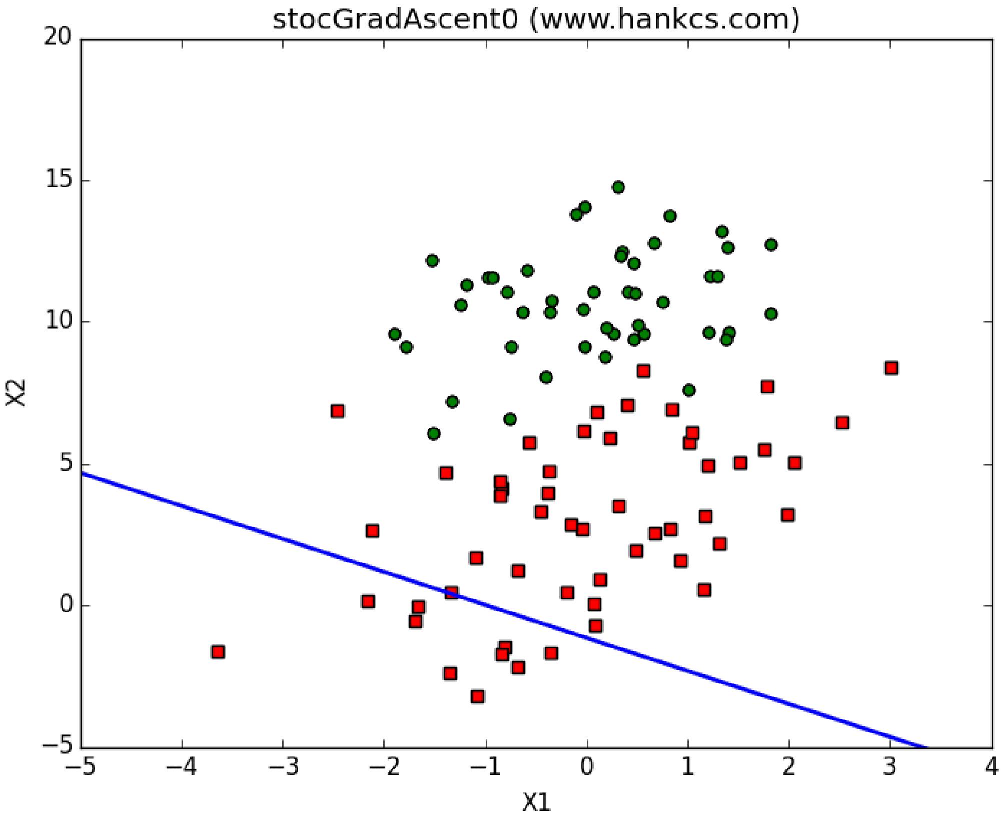
Batch Training vs Online Training I



Batch training (gradient descent).

Figure is from http://www.hankcs.com/ml/the-logistic-regression-and-the-maximum-entropy-model.html

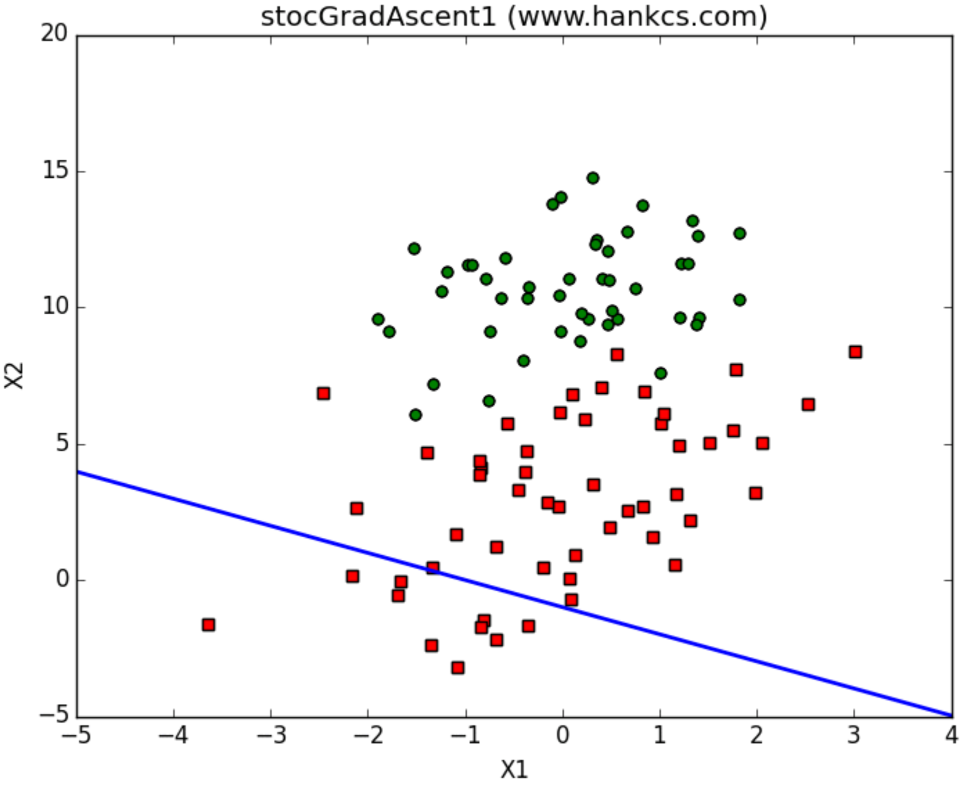
Batch Training vs Online Training II



Online training (stochastic gradient descent).

Figure is from http://www.hankcs.com/ml/the-logistic-regression-and-the-maximum-entropy-model.html

Batch Training vs Online Training III



Online training (improved stochastic gradient descent).

Figure is from http://www.hankcs.com/ml/the-logistic-regression-and-the-maximum-entropy-model.html

Multiple Logistic Regression

- When there is more than one predictors, logistic regression model the outcome by

$$P(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots\beta_p X_p}} = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots\beta_p X_p)}}$$

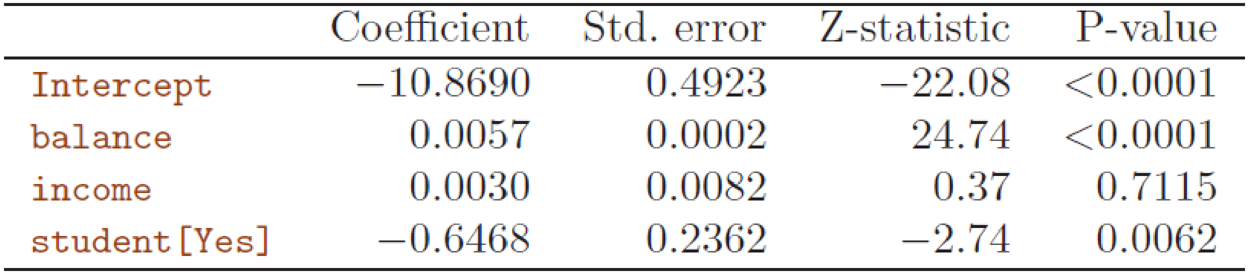
- The likelihood function is

$$\ell(\beta_0, \beta_1, \cdots, \beta_p) = \prod_i \frac{1}{1+e^{-\left(y_i\left(\beta_0+\beta_1 x_i^{(1)}+\cdots\beta_p x_i^{(p)}\right)\right)}}$$

- The maximum likelihood estimates of the parameters are

$$(\widehat{\beta_0}, \widehat{\beta_1}, \cdots, \widehat{\beta_p}) = \underset{\beta_0,\beta_1,\cdots,\beta_p}{\operatorname{argmin}} \left\{-\log \ell(\beta_0, \beta_1, \cdots, \beta_p) = \sum_i \log\left(1+e^{-\left(y_i\left(\beta_0+\beta_1 x_i^{(1)}+\cdots\beta_p x_i^{(p)}\right)\right)}\right)\right\}$$

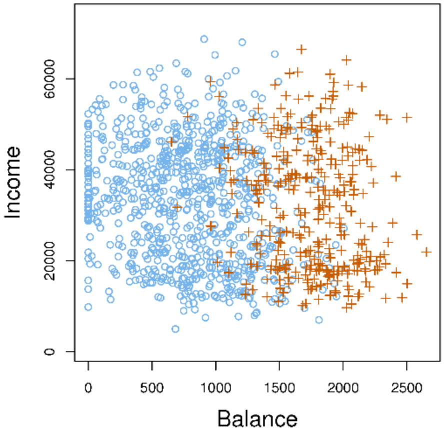
Logistic Regression on the Default Data



	Coefficient	Std. error	Z-statistic	P-value
Intercept	−10.8690	0.4923	−22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student[Yes]	−0.6468	0.2362	−2.74	0.0062

Std. error is used to compute confidence intervals. That is, there is approximately 95% chance that the interval $[\hat{\beta}_i - 2\mathrm{SE}(\widehat{\beta_1}), \hat{\beta}_i + 2\mathrm{SE}(\widehat{\beta_1})]$ contain the true value of β_i.

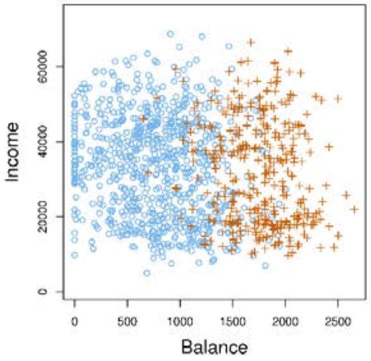
Large absolute value of Z-statistic or small value of P-value indicates strong evidence that the corresponding predictor is associated with the outcome.



An visualization of the Default data set. The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.

Motivation of Sparse Models

- Revisit of the default data set



  - There is no strong association between the predictor income and the outcome default.
  - The single predictor balance may lead to better generalization performance (on testing data).

- In many real word applications (especially biomedical studies), the number of features/predictors (genes, SNPs, etc.) is much larger than the number of samples (e.g., patients).
- Many studies indicate that only a very few of the features contribute to the outcome (symptoms of diseases, etc.).

Sparse Logistic Regression

- Introduction of sparsity to the probability

$$P(X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots\beta_p X_p)}}$$

Sparsity: many coefficients are zero, i.e., they have no contributions to the probability

- We can enforce sparsity by the ℓ_1 regularizer.

$$\left(\widehat{\beta_0}, \widehat{\beta_1}, \cdots, \widehat{\beta_p}\right) = \underset{\beta_0,\beta_1,\cdots,\beta_p}{\operatorname{argmin}} \left\{ -\log \ell(\beta_0, \beta_1, \cdots, \beta_p) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Usually, we do not penalize the intercept β_0.

$$= \underset{\beta_0,\beta_1,\cdots,\beta_p}{\operatorname{argmin}} \left\{ \sum_i \log\left(1 + e^{-\left(y_i\left(\beta_0+\beta_1 x_i^{(1)}+\cdots\beta_p x_i^{(p)}\right)\right)}\right) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Projected Gradient Descent

- Another popular formulation of sparse logistic regression is

$$\min_{\beta_0,\beta_1,\cdots,\beta_p} \left\{ \sum_i \log\left(1 + e^{-\left(y_i\left(\beta_0+\beta_1 x_i^{(1)}+\cdots\beta_p x_i^{(p)}\right)\right)}\right) \right\}$$

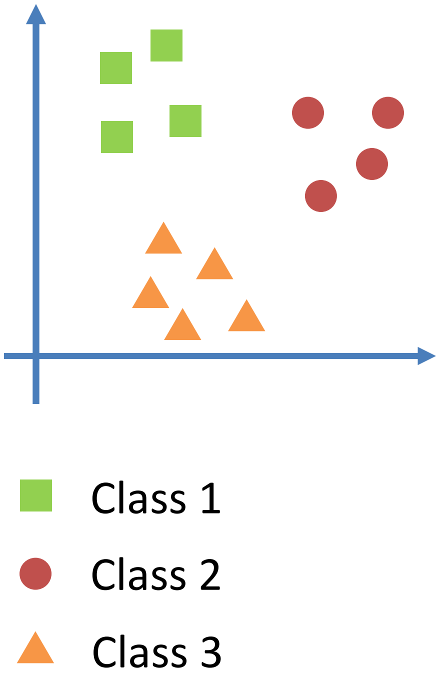
$$\text{s.t.} \quad \sum_{j=1}^{p} \left|\beta_j\right| \le r$$

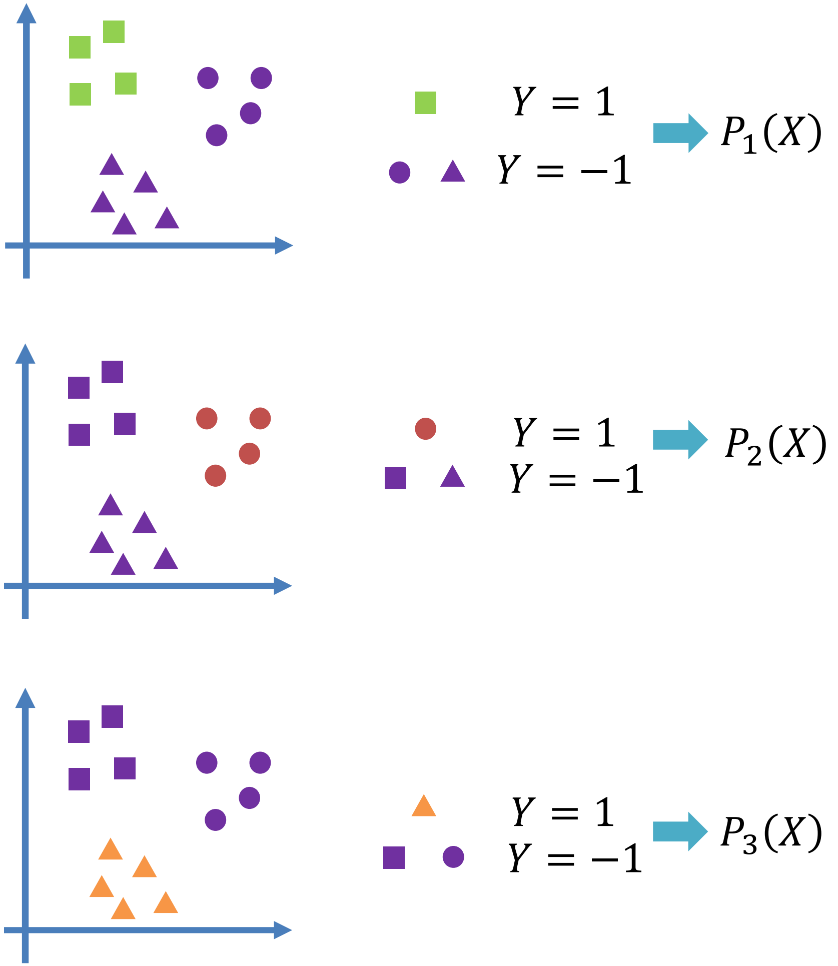
- Framework of projected gradient descent
    - in the k^{th} iteration
        - Update $\left(\beta_0^{(k)}, \tilde{\beta}_1^{(k)}, \cdots, \tilde{\beta}_p^{(k)}\right)$ by gradient descent from $\left(\beta_0^{(k-1)}, \beta_1^{(k-1)}, \cdots, \beta_p^{(k-1)}\right)$
        - Project $\left(\tilde{\beta}_1^{(k)}, \tilde{\beta}_2^{(k)}, \cdots, \tilde{\beta}_p^{(k)}\right)$ to the set

$$\mathcal{S} = \left\{\left(\beta_1, \beta_2, \cdots, \beta_p\right): \sum_{j=1}^{p} \left|\beta_j\right| \le r \right\}$$

Logistic Regression for Multi-Class Problems

- One vs All





Given a new instance x, we set the class label of x to k such that $P_k(x) \geq P_i(x)$ for all i.

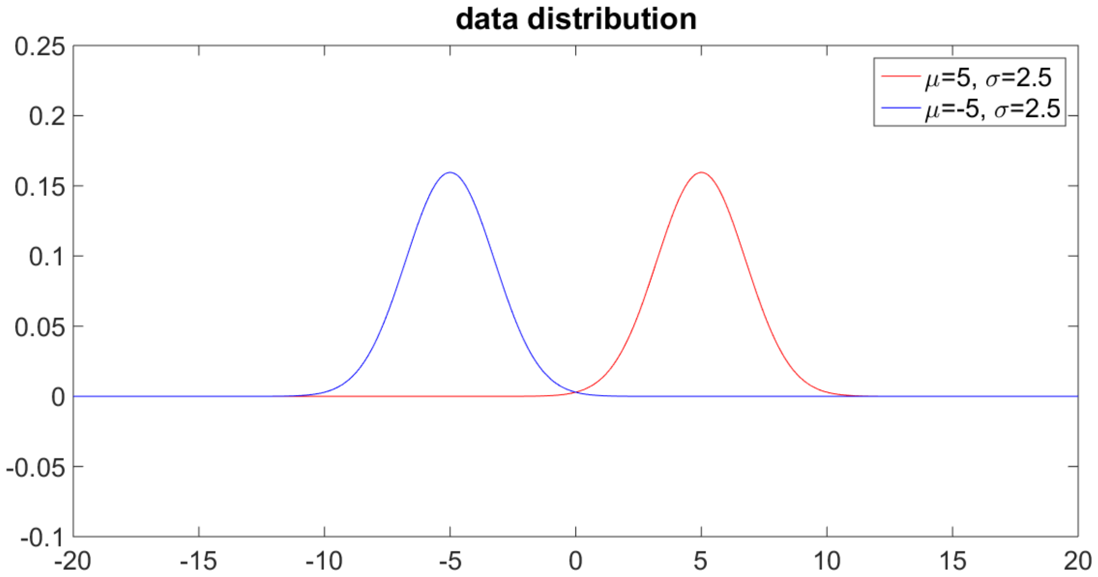
Tip for Unbalanced Data: Undersampling

- In many (probably most of them) real applications, the data are unbalanced. In other words, the number of positive samples is much larger or smaller than the number of negative samples, which may cause serious problem in training the modeling.

- The common used technique to handle the unbalanced data is undersampling, i.e., we undersample the *larger* class of data (the one that has more data samples) such that the two classes have roughly the same number of data samples.

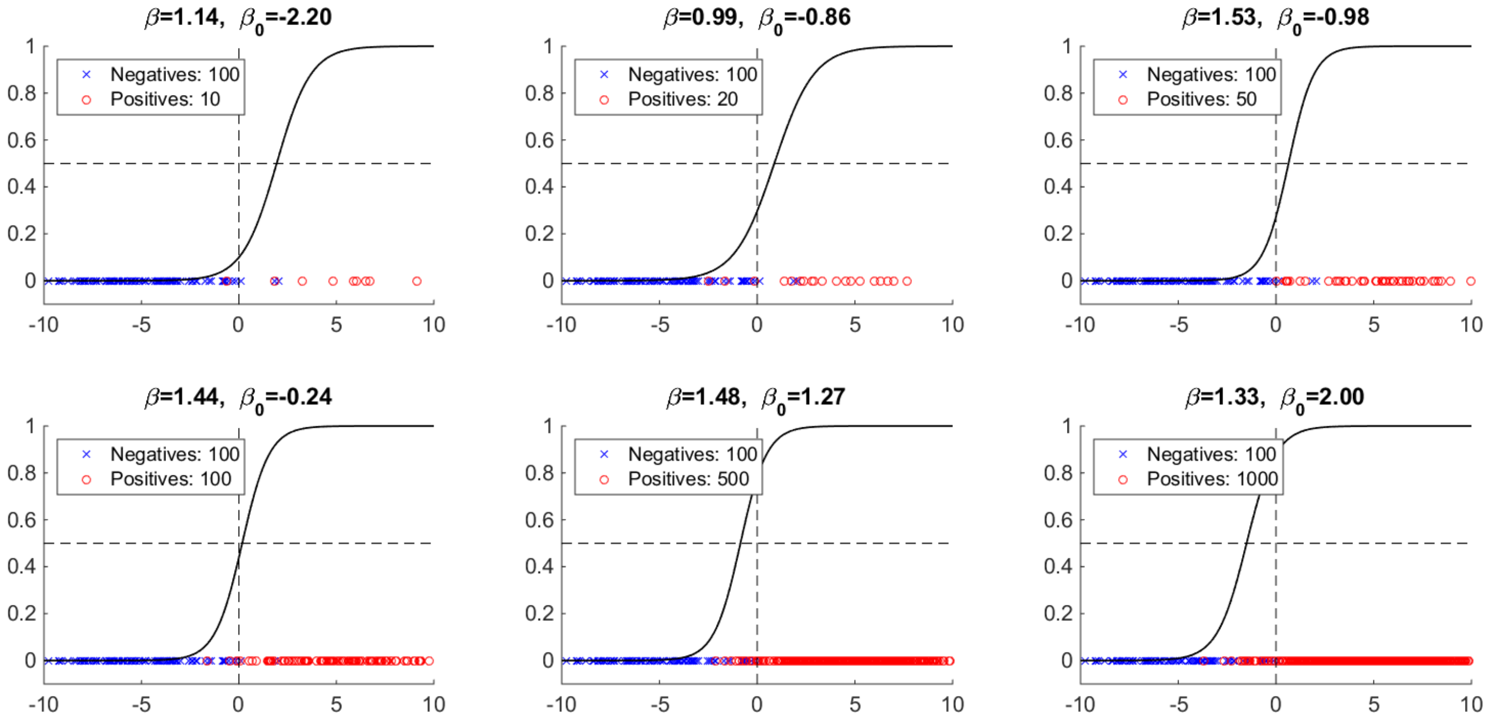
A Synthetic Data

- Why synthetic data?
  - We know the ground truth.



Positive samples: $\sim N(5, 2.5)$.
Negative samples: $\sim N(-5, 2.5)$.

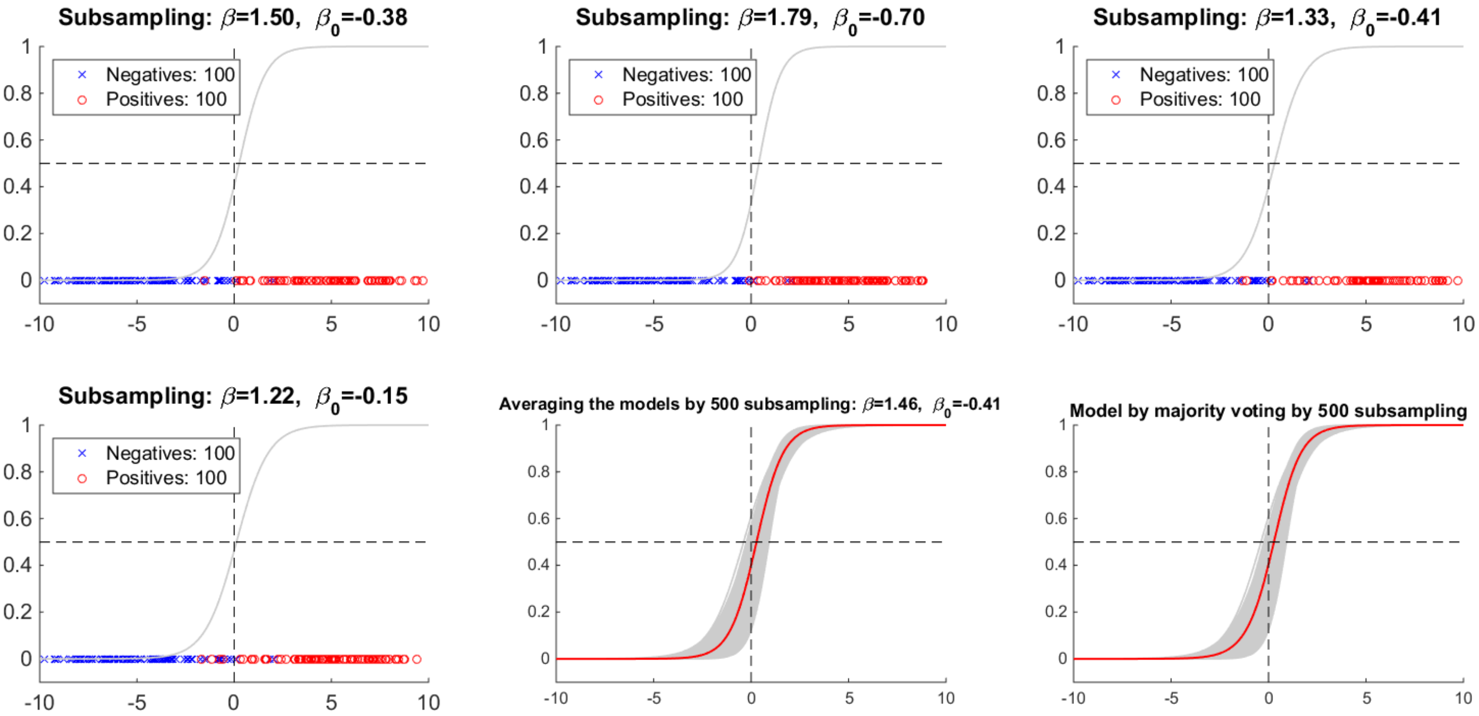
Imbalanced Data



Fitted Models on imbalanced data.

Undersampling and Ensembling

- We undersample the data such that the number of the data samples of two classes are roughly the same.



Fitted models on the undersampled data. The fifth figure plots the model obtained by averaing the parameters. The last one plots the model by majority voting.

Resources

- Books
  - An Introduction to Statistical Learning with applications in R, Section 4.3.
  - Bayesian Reasoning and Machine Learning, Section 17.4.

- Software
  - SLEP: Sparse Learning with Efficient Projections. http://yelab.net/software/SLEP/