

Logistic Regression

Mohammad Emtiyaz Khan
EPFL

Oct 8, 2015



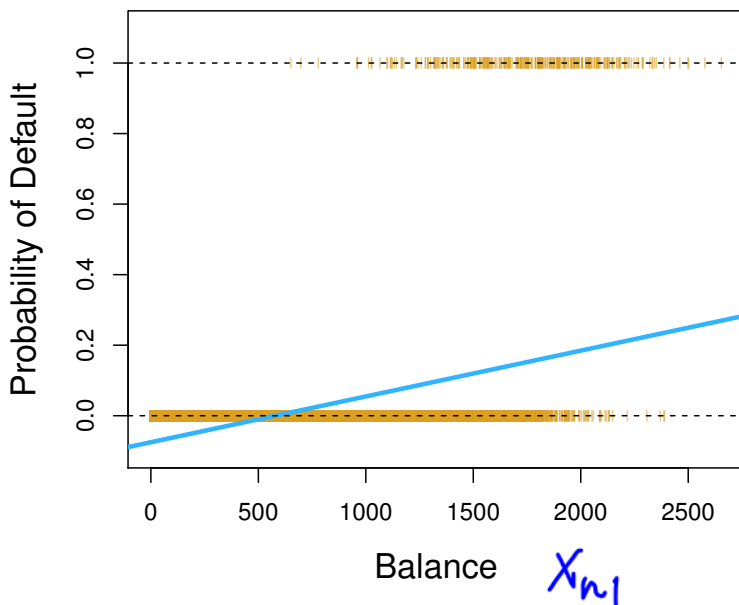
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

©Mohammad Emtiyaz Khan 2015

Classification with linear regression

We can use $y = 0$ for \mathcal{C}_1 and $y = 1$ for \mathcal{C}_2 (or vice-versa), and simply use least-squares to predict \hat{y}_* given \mathbf{x}_* . We can predict \mathcal{C}_1 when $\hat{y}_* < 0.5$ and \mathcal{C}_2 when $\hat{y}_* > 0.5$.

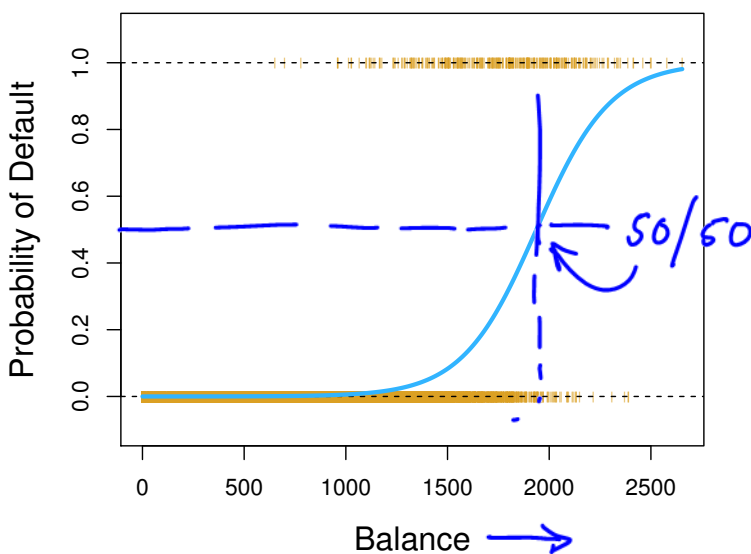
Default



$$p(\text{Default} = 1 \mid x)$$

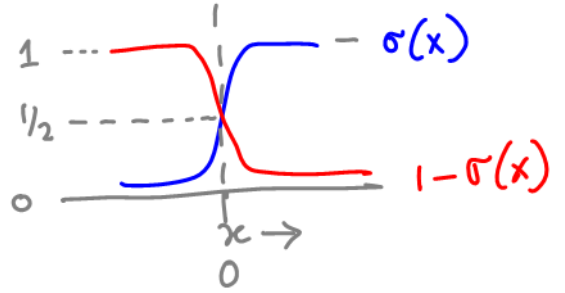
$$\beta_0 + \beta_1 x_{n1} \approx y_n$$

Any problems with this approach?



Logistic regression

We need to model $p(y = \mathcal{C}_1 | \mathbf{x})$ and $p(y = \mathcal{C}_2 | \mathbf{x})$ such that they both are > 0 and also sum to 1. For a new input \mathbf{x}_* , we can classify to \mathcal{C}_1 when $p(\hat{y}_* | \mathbf{x}_*) < 0.5$.



We will use the logistic function.

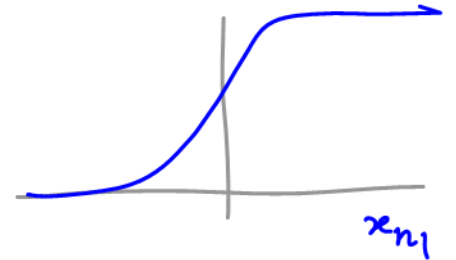
$$\sigma(x) + (1 - \sigma(x)) = 1$$

$$\sigma(x) = \frac{\exp(x)}{1 + \exp(x)}, \quad 1 - \sigma(x) = \frac{1}{1 + \exp(x)}$$

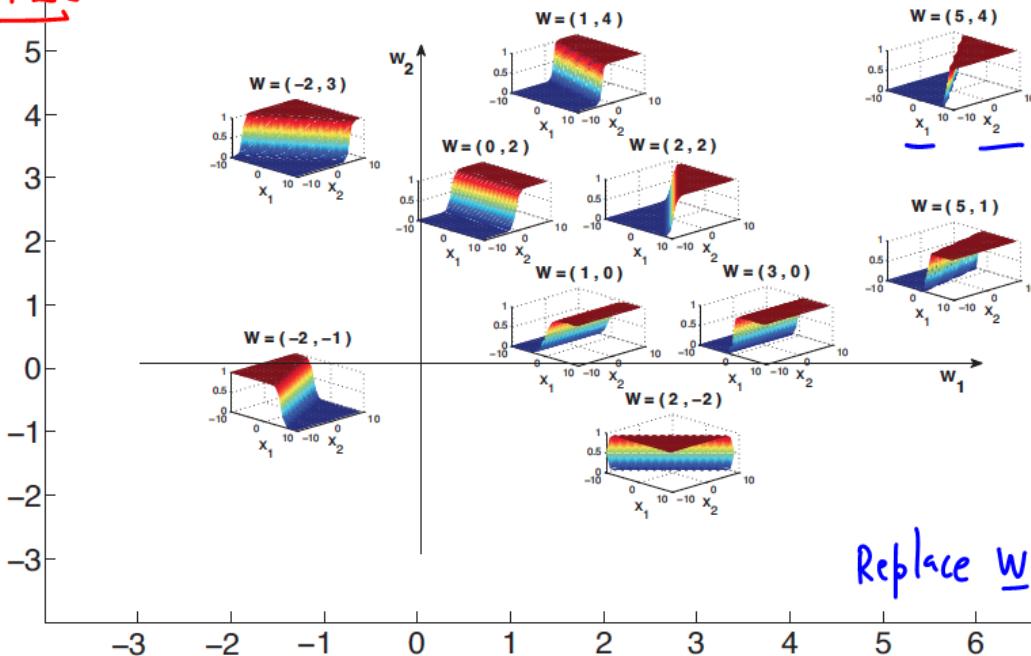
We pass the linear-regression model $\eta_n = \tilde{\mathbf{x}}^T \boldsymbol{\beta}$ through the logistic function to get the probabilities.

$$p(y_n = \mathcal{C}_1 | \mathbf{x}_n) = \sigma(\eta_n), \quad p(y_n = \mathcal{C}_2 | \mathbf{x}_n) = 1 - \sigma(\eta_n)$$

This figure visualizes the probabilities obtained for a 2-D problem (taken from KPM Chapter 7).



Visualize



$$\boldsymbol{\beta} = \frac{\sum \tilde{\mathbf{x}}_n^T}{n}$$

Replace w by beta

The probabilistic model

Assuming that each y_n is independent of others, we can define the probability of \mathbf{y} given \mathbf{X} and β :

(A)
$$p(\mathbf{y}|\mathbf{X}, \beta) = \prod_{n=1}^N p(y_n|\mathbf{x}_n)$$
← $\sigma(\beta^T \tilde{\mathbf{x}}_n)$ or $[1 - \sigma(\beta^T \tilde{\mathbf{x}}_n)]$

(B)
$$= \prod_{n:y_n=C_1} p(y_n = C_1|\mathbf{x}_n) \prod_{n:y_n=C_2} p(y_n = C_2|\mathbf{x}_n)$$
↙ $\sigma(\tilde{\mathbf{x}}_n^T \beta)$ ↘ $(1 - \sigma(\tilde{\mathbf{x}}_n^T \beta))$

A better way to write this is to use the coding $y_n \in \{0, 1\}$.

(C)
$$p(\mathbf{y}|\mathbf{X}, \beta) = \prod_{n=1}^N \sigma(\eta_n)^{y_n} [1 - \sigma(\eta_n)]^{1-y_n}$$
{ $\begin{cases} \phi(y_n=C_1/\tilde{\mathbf{x}}_n) & \text{when } y_n=C_1 \\ \phi(y_n=C_2/\tilde{\mathbf{x}}_n) & \text{when } y_n=C_2 \end{cases}$

The log-likelihood is given as follows:

(D)
$$\mathcal{L}_{mle}(\beta) = \sum_{n=1}^N y_n \log \sigma(\tilde{\mathbf{x}}_n^T \beta) + (1 - y_n) \log [1 - \sigma(\tilde{\mathbf{x}}_n^T \beta)]$$
log p(y/x, beta)

(E)
$$= \sum_{n=1}^N y_n \tilde{\mathbf{x}}_n^T \beta - \log [1 + \exp(\tilde{\mathbf{x}}_n^T \beta)]$$
↑ Gives us (D)

Proof: (C) → (E)

$$\log \sigma(\tilde{\mathbf{x}}_n^T \beta) = \log \frac{e^{\tilde{\mathbf{x}}_n^T \beta}}{1 + e^{\tilde{\mathbf{x}}_n^T \beta}} = \tilde{\mathbf{x}}_n^T \beta - \log [1 + e^{\tilde{\mathbf{x}}_n^T \beta}]$$

$$\log [1 - \sigma(\cdot)] = \log \frac{1}{1 + e^{\tilde{\mathbf{x}}_n^T \beta}} = -\log [1 + e^{\tilde{\mathbf{x}}_n^T \beta}]$$

To get (E), Add these for all n (y_n + 1 - y_n) = 1

$$= \sum_{n=1}^N y_n \tilde{\mathbf{x}}_n^T \beta - y_n \log [1 + \exp(\tilde{\mathbf{x}}_n^T \beta)] - (1 - y_n) \log [1 + \exp(\tilde{\mathbf{x}}_n^T \beta)]$$

$$\mathcal{L}(\beta) := \sum_n y_n \tilde{x}_n^T \beta - \log [1 + \exp(\tilde{x}_n^T \beta)]$$

Maximum likelihood

We will use the following fact to derive the gradient.

$$\frac{\partial}{\partial t} \log [1 + e^t] = \frac{e^t}{1 + e^t} = \sigma(t)$$

$$\frac{\partial}{\partial x} \log [1 + \exp(x)] = \sigma(x)$$

Taking the gradient of the log-likelihood, we get the following:

$$\mathbf{g} := \frac{\partial \mathcal{L}}{\partial \beta} = \tilde{\mathbf{X}}^T [\sigma(\tilde{\mathbf{X}}\beta) - \mathbf{y}] = 0$$

This is similar to the normal equation for least-squares.

There are no closed-form solutions, but we can use gradient descent.

Convexity

The negative of the log-likelihood $-\mathcal{L}_{mle}(\beta)$ is convex.

Proof I: The sum of a linear function and a (strictly) convex function is (strictly) convex.

Proof II: The Hessian of a convex function is positive semi-definite and for a strictly-convex function it is positive definite.

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_n \tilde{x}_n y_n - \tilde{x}_n \sigma(\tilde{x}_n^T \beta) = \sum_n \tilde{x}_n [y_n - \sigma(\tilde{x}_n^T \beta)]$$

$$\sigma(\tilde{x}_n^T \beta) := \begin{bmatrix} \sigma(\tilde{x}_1^T \beta) \\ \sigma(\tilde{x}_2^T \beta) \\ \vdots \\ \sigma(\tilde{x}_N^T \beta) \end{bmatrix}$$

$$\begin{aligned} \max_{\beta} \mathcal{L}(\beta) \\ = \min_{\beta} -\mathcal{L}(\beta) \end{aligned}$$

$$\begin{aligned} \mathcal{L}(\beta) &= \sum_{n=1}^N (\tilde{x}_n^T \beta) y_n - \log(1 + e^{\tilde{x}_n^T \beta}) \\ -\mathcal{L}(\beta) &= \sum_{n=1}^N -(\tilde{x}_n^T \beta) + \log(1 + e^{\tilde{x}_n^T \beta}) \end{aligned}$$

$$-\frac{\partial \mathcal{L}}{\partial \beta} = \sum_n -y_n \tilde{x}_n + \sigma(\tilde{x}_n^T \beta) \tilde{x}_n$$

$$-\frac{\partial^2 \mathcal{L}}{\partial \beta \partial \beta^T} = \frac{\partial}{\partial \beta \partial \beta^T} \sigma(\tilde{x}_n^T \beta) \tilde{x}_n$$

Q

Hessian of the Log-Likelihood

We will use the following fact:

$$\frac{\partial \sigma(t)}{\partial t} = \sigma(t)[1 - \sigma(t)]$$

Taking the derivative of the gradient we get the Hessian,

$$\mathbf{H}(\boldsymbol{\beta}) := -\frac{\partial \mathbf{g}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}^T} = \tilde{\mathbf{X}}^T \mathbf{S} \tilde{\mathbf{X}}$$

where \mathbf{S} is a $N \times N$ diagonal matrix with diagonals

$$S_{nn} = \sigma(\tilde{\mathbf{x}}_n^T \boldsymbol{\beta})[1 - \sigma(\tilde{\mathbf{x}}_n^T \boldsymbol{\beta})].$$

Is the negative of the log-likelihood *strictly* convex?

Newton's Method

Gradient descent uses only first-order information and takes steps in the direction of the gradient.

Newton's method uses second-order information and takes steps in the direction that minimizes a quadratic approximation.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}_k$$

where \mathbf{g}_k is the gradient.

Computational complexity

Compare the computational complexity of least-squares and Newton's method.

Newton's method is equivalent to solving many least-squares problems.

Penalized Logistic Regression

The cost-function can be unbounded when the data is [linearly separable](#).

For a well-defined problem, we will regularize.

$$\min_{\beta} - \sum_{n=1}^N \log p(y_n | \mathbf{x}_n^T, \beta) + \lambda \sum_{d=1}^D \beta_d^2$$

Additional notes

Derivation of Newton's method

The second-order approximation of a function is given as follows:

$$\mathcal{L}_Q(\boldsymbol{\beta}) := \mathcal{L}(\boldsymbol{\beta}^{(k)}) + \mathbf{g}_k^T(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}) + \frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)})^T \mathbf{H}_k(\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)})$$

The minimum of \mathcal{L}_Q is at $\boldsymbol{\beta}^{(k)} - \mathbf{H}_k^{-1} \mathbf{g}_k$. A conservative option is to take a small step in this direction using step-size α_k , which is the step used in Newton's method.

Set α_k using line search, e.g. the [Armijo rule](#). See Section 8.3.2 of Kevin Murphy's book. A good implementation can be found on page 29 of Bertsekas book "Non-linear programming".

Iterative Recursive Least-Squares (IRLS)

(IRLS) expresses Newton's method with $\alpha_k = 1$ as a sequence of least-squares problems. Below is the derivation and pseudo code.

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}_k \tag{1}$$

$$= \boldsymbol{\beta}^{(k)} - (\tilde{\mathbf{X}}^T \mathbf{S}_k \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T (\boldsymbol{\sigma}_k - \mathbf{y}) \tag{2}$$

$$= (\tilde{\mathbf{X}}^T \mathbf{S}_k \tilde{\mathbf{X}})^{-1} [(\tilde{\mathbf{X}}^T \mathbf{S}_k \tilde{\mathbf{X}}) \boldsymbol{\beta}^{(k)} - \tilde{\mathbf{X}}^T (\boldsymbol{\sigma}_k - \mathbf{y})]$$

$$= (\tilde{\mathbf{X}}^T \mathbf{S}_k \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{X}_k [\tilde{\mathbf{X}} \boldsymbol{\beta}^{(k)} + \mathbf{S}_k^{-1} (\mathbf{y} - \boldsymbol{\sigma}_k)]$$

$$= (\tilde{\mathbf{X}}^T \mathbf{S}_k \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{X}_k \mathbf{z}_k \tag{3}$$

where $\mathbf{z}_k = \tilde{\mathbf{X}} \boldsymbol{\beta}^{(k)} + \mathbf{S}_k^{-1} (\mathbf{y} - \boldsymbol{\sigma}_k)$.

```
1 for k = 1:maxIters
2     sig = sigmoid(tX*beta);
3     s = sig.*(1-sig);
4     z = tX*beta + (y-sig)./s;
5     beta = weightedLeastSquares(z,tX,s);
6 end
```

Quasi-Newton

Read about [L-BFGS](#) in Section 8.3.5 of Kevin Murphy's book. The key idea is to approximate \mathbf{H} using a diagonal and a low-rank matrix.

To do

1. Practice to derive the cost function using maximum likelihood estimation.
2. Understand the normal equation.
3. Understand the interpretation of log-odds (JWHT Chapter 3).
4. Learn to prove convexity using the positive-definite property of the Hessian.
5. Implement Newton's method (part of next week's lab).
6. Understand the relationship of Newton's Method with IRLS.