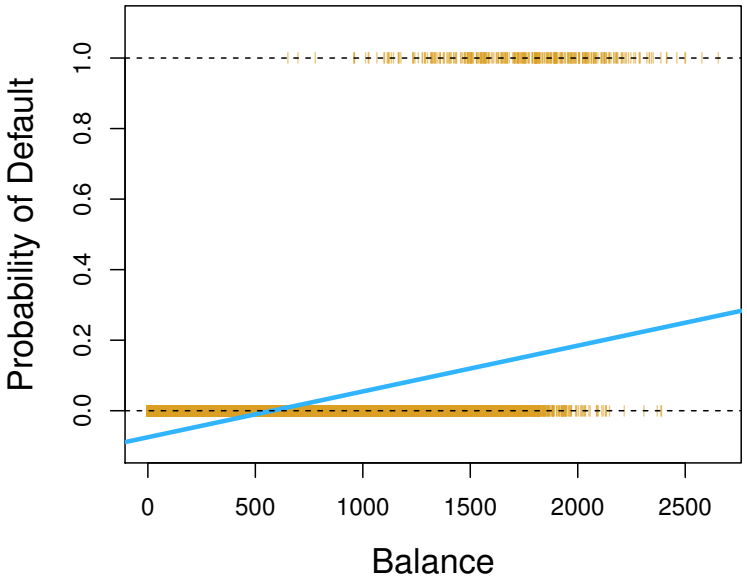
Logistic Regression

Mohammad Emtiyaz Khan
EPFL

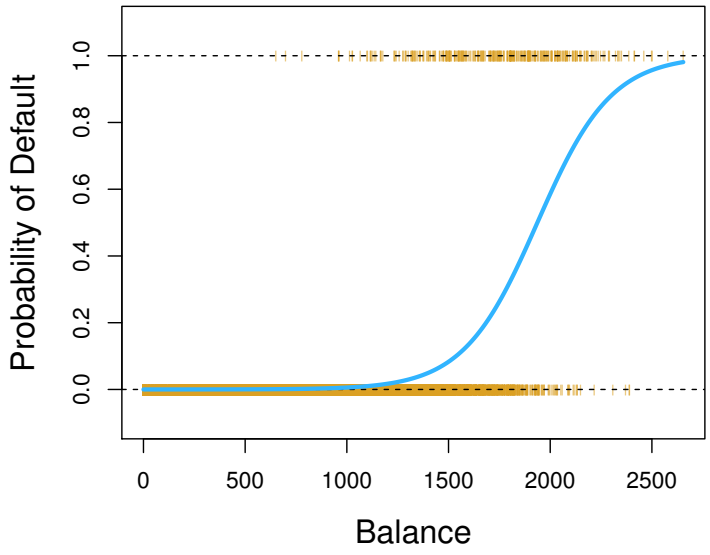
Oct 8, 2015

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 $\hat{y}_* > 0.5$.



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 $p(\hat{y}_*|\mathbf{x}_*) < 0.5$.

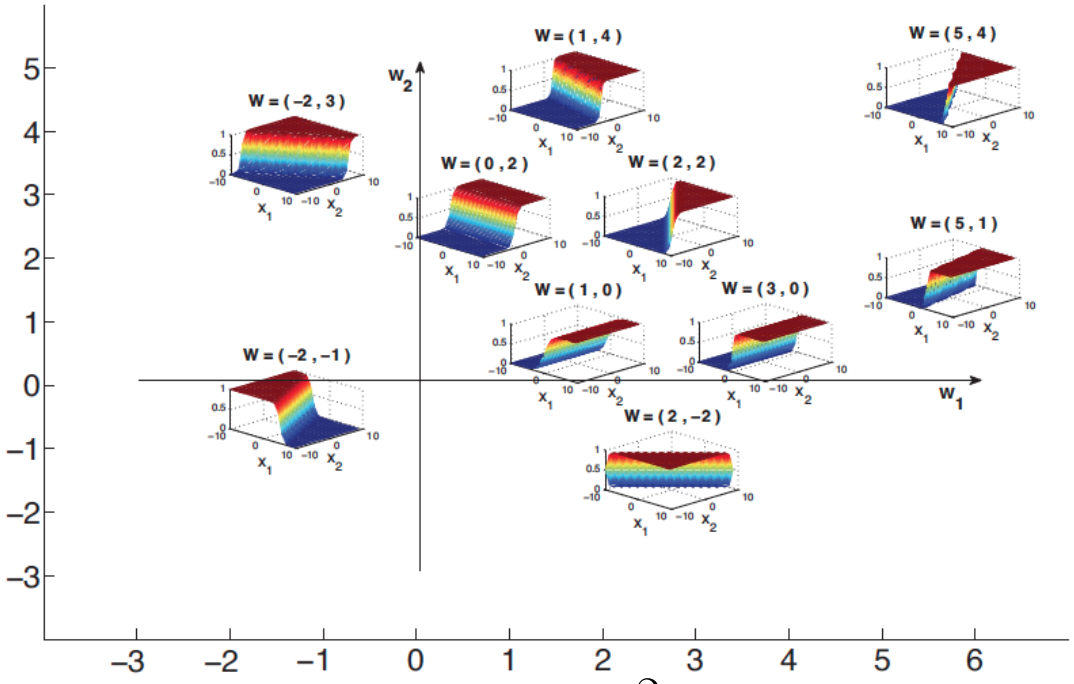
We will use the logistic function.

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

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

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

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



The probabilistic model

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

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n)$$
$$= \prod_{n:y_n=\mathcal{C}_1} p(y_n = \mathcal{C}_1|\mathbf{x}_n) \prod_{n:y_n=\mathcal{C}_2} p(y_n = \mathcal{C}_2|\mathbf{x}_n)$$

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

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\beta}) = \prod_{n=1}^{N} \sigma(\eta_n)^{y_n}[1 - \sigma(\eta_n)]^{1-y_n}$$

The log-likelihood is given as follows:

$$\mathcal{L}_{mle}(\boldsymbol{\beta}) = \sum_{n=1}^{N} y_n \log \sigma(\widetilde{\mathbf{x}}_n^T \boldsymbol{\beta}) + (1 - y_n) \log[1 - \sigma(\widetilde{\mathbf{x}}_n^T \boldsymbol{\beta})]$$
$$= \sum_{n=1}^{N} y_n \widetilde{\mathbf{x}}_n^T \boldsymbol{\beta} - \log[1 + \exp(\widetilde{\mathbf{x}}_n^T \boldsymbol{\beta})]$$

Maximum likelihood

We will use the following fact to derive the gradient.

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

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

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

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 $-\mathcal{L}_{mle}(\boldsymbol{\beta})$ 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.

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,

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

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

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

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.

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

Additional notes

Derivation of Newton's method

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

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

The minimum of $\mathcal{L}_Q$ is at $\boldsymbol{\beta}^{(k)} - \mathbf{H}_k^T\mathbf{g}_k$. 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.

$$\begin{aligned}
\boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(k)} - \alpha_k\mathbf{H}_k^{-1}\mathbf{g}_k && (1)\\
&= \boldsymbol{\beta}^{(k)} - (\widetilde{\mathbf{X}}^T\mathbf{S}_k\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T(\boldsymbol{\sigma}_k - \mathbf{y}) && (2)\\
&= (\widetilde{\mathbf{X}}^T\mathbf{S}_k\widetilde{\mathbf{X}})^{-1}[(\widetilde{\mathbf{X}}^T\mathbf{S}_k\widetilde{\mathbf{X}})\boldsymbol{\beta}^{(k)} - \widetilde{\mathbf{X}}^T(\boldsymbol{\sigma}_k - \mathbf{y})]\\
&= (\widetilde{\mathbf{X}}^T\mathbf{S}_k\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T\mathbf{X}_k[\widetilde{\mathbf{X}}\boldsymbol{\beta}^{(k)} + \mathbf{S}_k^{-1}(\mathbf{y} - \boldsymbol{\sigma}_k)]\\
&= (\widetilde{\mathbf{X}}^T\mathbf{S}_k\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T\mathbf{X}_k\mathbf{z}_k && (3)
\end{aligned}$$

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

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

Quasi-Newton

Read about L-BFGS in Section 8.3.5 of Kevin Murphy's book. The key idea is to approximate $\mathbf{H}$ usign 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.