Logistic Regression

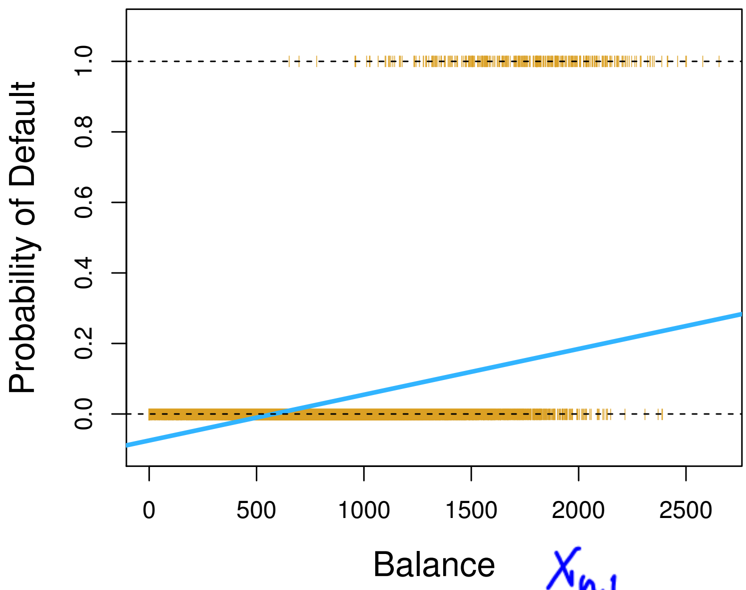
Mohammad Emtiyaz Khan
EPFL

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

Default

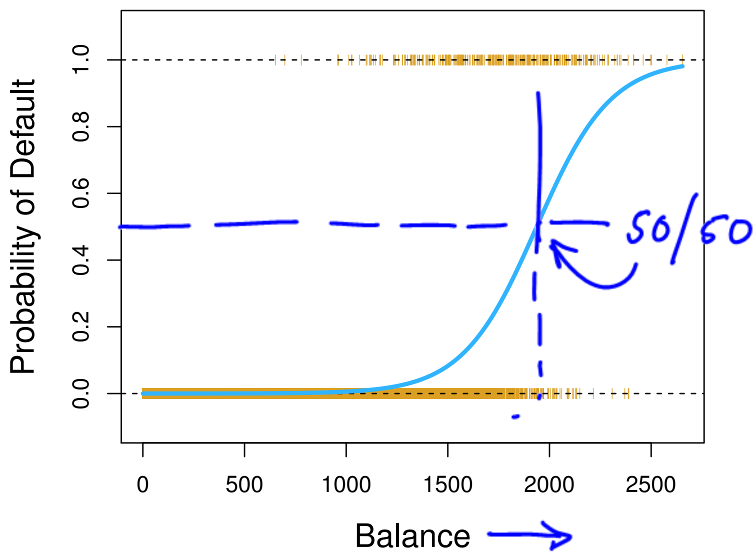


x_{n1}

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

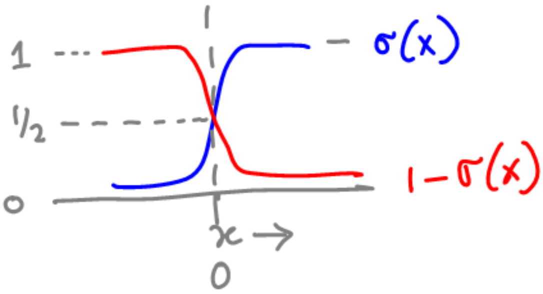
$\beta_0 + \beta_1 x_{n1} \simeq y_n$

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.

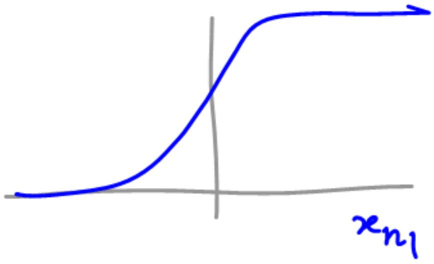
$\sigma(x) + (1 - \sigma(x)) = 1$

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

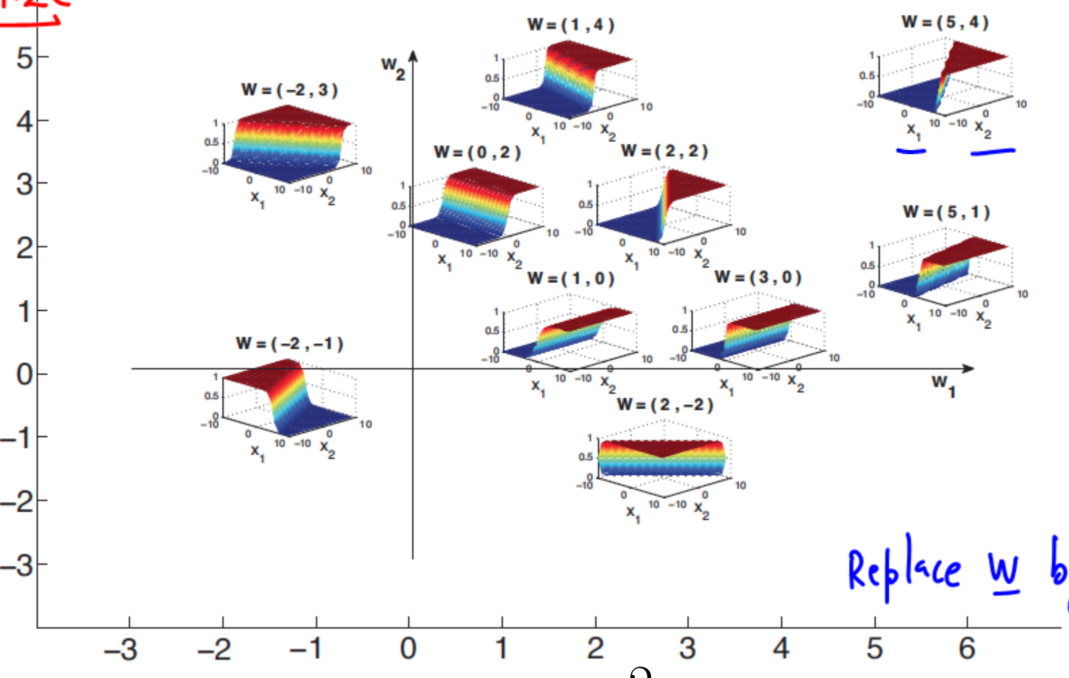
We pass the linear-regression model $\eta_n = \widetilde{\mathbf{x}}^T \boldsymbol{\beta}$ through the logistic function to get the probabilities.

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

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



Visualize



$\underline{\beta}^T \underline{\tilde{X}}_n$

Replace $\underline{w}$ by $\underline{\beta}$

The probabilistic model

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

$\sigma(\underline{\beta}^T\underline{\tilde{X}}_n)$ or $[1-\sigma(\underline{\beta}^T\underline{\tilde{X}}_n)]$

$$\text{A} \quad p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n)$$

$$\text{B} \quad = \prod_{n:y_n=\mathcal{C}_1} p(y_n = \overset{1}{\mathcal{C}_1}|\mathbf{x}_n) \prod_{n:y_n=\mathcal{C}_2} p(y_n = \overset{0}{\mathcal{C}_2}|\mathbf{x}_n)$$

$\sigma(\underline{\tilde{X}}_n^T\underline{\beta})$ $\quad$ $(1-\sigma(\underline{\tilde{X}}_n^T\underline{\beta}))$

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

$$\begin{cases} p(y_n=C_1/\underline{X}_n), \text{ when } y_n = C_1 \\ p(y_n=C_2/\underline{X}_n), \text{ when } y_n = C_2 \end{cases}$$

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

The log-likelihood is given as follows:

$\log p(\underline{y}/X, \underline{\beta})$

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

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

Gives us D

Proof: C $\rightarrow$ E

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

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

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

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

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

Maximum likelihood

We will use the following fact to derive the gradient.

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

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

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

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

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

Least-squares: $-\tilde{X}^T[\tilde{X}\beta - y]$

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

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.

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

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

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

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

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

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

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

Q

Hessian of the Log-Likelihood

We will use the following fact:

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

$$\frac{\partial}{\partial t} \frac{e^t}{1+e^t} = -\frac{e^{2t}}{(1+e^t)^2} + \frac{e^t}{1+e^t}$$
$$= -[\sigma(t)]^2 + \sigma(t)$$
$$= \sigma(t)[1-\sigma(t)]$$

Taking the derivative of the gradient we get the Hessian,

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

$\widetilde{X}^T \widetilde{X}$

$$\underbrace{\frac{\partial^2 \mathcal{L}}{\partial \underline{\beta}\, \partial \underline{\beta}^T}}_{(D+1)\times(D+1)} = \sum_{n=1}^{N} \frac{\partial}{\partial \underline{\beta}^T} \underbrace{\sigma(\underline{\widetilde{x}}_n^T \underline{\beta})\, \underline{\widetilde{x}}_n}_{(D+1)}$$

$$= \sum_{n=1}^{N} \underbrace{\underline{\widetilde{x}}_n}_{(D+1)\times 1} \underbrace{\sigma(\underline{\widetilde{x}}_n^T \underline{\beta}) [1 - \sigma(\underline{\widetilde{x}}_n^T \underline{\beta})]}_{1\times 1} \underbrace{\underline{\widetilde{x}}_n^T}_{1\times(D+1)}$$

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

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

S_{nn}

$$S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & \ddots & \\ & & & S_{NN} \end{bmatrix}_{N\times N}$$

Is the negative of the log-likelihood *strictly* convex? No

$$\underline{a}^T H \underline{a} > 0$$
$$\Rightarrow \underline{a}^T \widetilde{X}^T S \widetilde{X} \underline{a} > 0$$
$$\Rightarrow \underbrace{\underline{a}^T \widetilde{X}^T S^{1/2}}_{t^T} S^{1/2} \widetilde{X} \underline{a} > 0$$
$$\Rightarrow \underline{t}^T \underline{t} > 0$$

but when $\exists\, \underline{a} \neq 0$ s.t. $\widetilde{X} \underline{a} = 0$
then $\underline{t}^T \underline{t} = 0$

else

$$\frac{\partial^2 \mathcal{L}}{\partial \underline{\beta}\, \partial \underline{\beta}^T} = \widetilde{X}^T S \widetilde{X}$$

Newton's Method

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

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

gradient descent: $\underline{\beta}^{(k+1)} = \underline{\beta}^{(k)} - \alpha_k \underline{g}_k$

Newton: $\underline{\beta}^{(k+1)} = \underline{\beta}^{(k)} - \alpha_k H_k^{-1} \underline{g}_k$

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

where $\underline{g}_k = \frac{\partial}{\partial \underline{\beta}}[-\mathcal{L}(\beta)]\Big|_{\beta=\beta^{(k)}}$

$H_k = \frac{\partial^2}{\partial \underline{\beta}\, \partial \underline{\beta}^T}\Big|_{\beta=\beta^{(k)}}$

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

Newton minimizes the quadratic approx. at β^k

where $\mathbf{g}_k$ is the gradient.

$\mathcal{L}(\underline{\beta}) \simeq \mathcal{L}(\underline{\beta}^{(k)}) + \underline{g}_k^T(\underline{\beta} - \underline{\beta}^{(k)}) + (\underline{\beta} - \underline{\beta}^{(k)})^T H_k (\underline{\beta} - \underline{\beta}^{(k)}) := Q(\underline{\beta}, \underline{\beta}^{(k)})$

$\beta^* = \arg\min_{\underline{\beta}} Q(\beta, \beta^{(k)}) = H_k^{-1} \underline{g}_k$

Computational complexity

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

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

only applies to linear reg.

Gradient descent	$O(NDI)$
least-squares	$O(ND^2 + D^3)$
Newton method	$O[(ND^2 + D^3)I]$

$\underbrace{(\tilde{X}^T S^{(k)} \tilde{X})^{-1}}_{H_k} \underbrace{\tilde{X}^T(\sigma(\tilde{X}\underline{\beta}^{(k)}) - \underline{y})}_{g_k}$

$D \times N \quad N \times D$

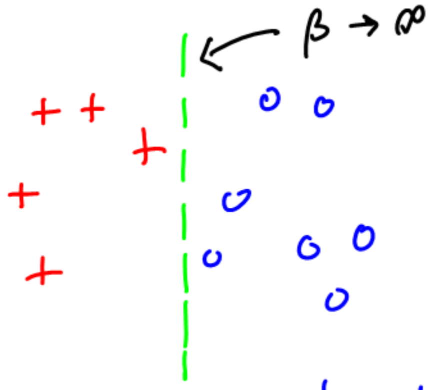
$D^2N \qquad D^3$

Least-Squares: $(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T(\tilde{X}\beta - \underline{y}) = 0$

Penalized Logistic Regression

In "linearly separable" case, as $\beta \to \infty$, $\mathcal{L}(\beta) \to \infty$.
Therefore the global minimum does not exist!

The cost-function can be unbounded when the data is linearly separable.



For a well-defined problem, we will regularize.

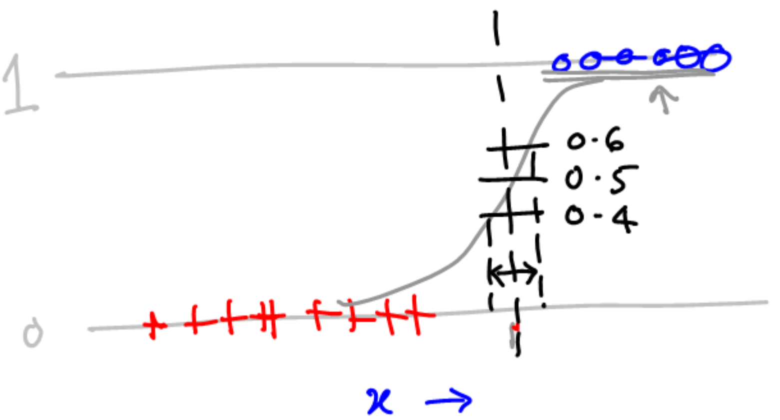
$$x^T S x + \lambda \begin{bmatrix} 0 & \\ & I_D \end{bmatrix}$$

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

[Regularization solves this problem]

[Typically, a linearly separable problem has many solutions.]

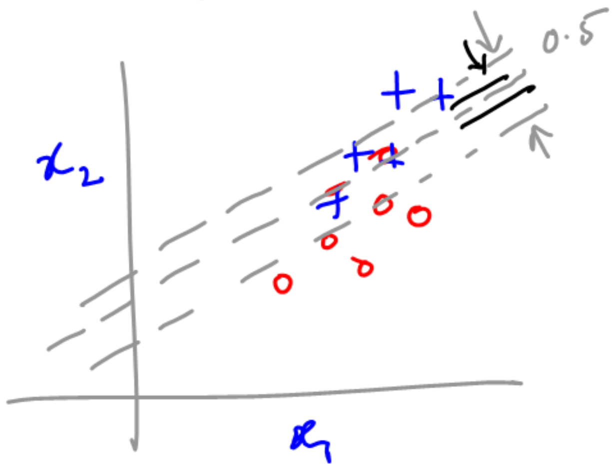
Linearly separable in 1-D



$x < \tau \Rightarrow y = \text{'+'}$

$x > \tau \Rightarrow y = \text{'0'}$

"Margin" in 2-D



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.

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

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.
7. Do exercise 8.3 to 8.7 in KPM book. (Chapter on logistic Regression)