

CS545: Classification with Logistic Regression

Chuck Anderson

Department of Computer Science
Colorado State University

Fall, 2009

Outline

CS545:
Classification with
Logistic Regression

Chuck Anderson

Logistic Regression
Derivation Recap
Implementation in R

Logistic Regression

Derivation Recap

Implementation in R

Derivation Recap

- $P(C = k|\mathbf{x}_n)$ and the data likelihood we want to maximize:

$$g(\mathbf{x}_n, \beta_k) = P(C = k|\mathbf{x}_n) = \frac{f(\mathbf{x}_n, \beta_k)}{1 + \sum_{m=1}^{K-1} f(\mathbf{x}_n, \beta_m)}$$

$$f(\mathbf{x}_n, \beta_k) = e^{\beta_k^T \mathbf{x}_n}$$

$$\begin{aligned} L(\beta) &= \prod_{n=1}^N \prod_{k=1}^{K-1} p(C = k|\mathbf{x}_n)^{t_{n,k}} \\ &= \prod_{n=1}^N \prod_{k=1}^{K-1} g(\mathbf{x}_n, \beta_k)^{t_{n,k}} \end{aligned}$$

- Gradient of log likelihood with respect to β_j :

$$\begin{aligned}\nabla_{\beta_j} l(\beta) &= \sum_{n=1}^N \sum_{k=1}^{K-1} \frac{t_{n,k}}{g(\mathbf{x}_n, \beta_k)} \nabla_{\beta_j} g(\mathbf{x}_n, \beta_k) \\ &= \sum_{n=1}^N \left(\sum_{k=1}^{K-1} t_{n,k} \delta_{jk} - g(\mathbf{x}_n, \beta_j) \sum_{k=1}^{K-1} t_{n,k} \right) \\ &= \sum_{n=1}^N \mathbf{x}_n (t_{n,j} - g(\mathbf{x}_n, \beta_j))\end{aligned}$$

- which results in this update rule for β_j

$$\beta_j \leftarrow \beta_j + \alpha \sum_{n=1}^N (t_{n,j} - g(\mathbf{x}_n, \beta_j)) \mathbf{x}_n$$

- How do we do this in R?

- Update rule for β_j

$$\beta_j \leftarrow \beta_j + \alpha \sum_{n=1}^N (t_{n,j} - g(\mathbf{x}_n, \beta_j)) \mathbf{x}_n$$

- What are shapes of each piece?
 - \mathbf{x}_n is $D \times 1$ (D includes the constant 1 input)
 - β_j is $D \times 1$
 - $t_{n,j} - g(\mathbf{x}_n, \beta_j)$ is a scalar
- So, this all works. But, notice the sum is over n , and each term in the product as n components, so we can do this as a dot product.

- Let's remove the sum and replace subscript n with $*$.
(This is not standard math notation!)

$$\beta_j \leftarrow \beta_j + \alpha \sum_{n=1}^N (t_{n,j} - g(\mathbf{x}_n, \beta_j)) \mathbf{x}_n$$

$$\beta_j \leftarrow \beta_j + \alpha (t_{*,j} - g(\mathbf{x}_*, \beta_j)) \mathbf{x}_*$$

- What are shapes of each piece?
 - $(t_{*,j} - g(\mathbf{x}_*, \beta_j))$ is $N \times 1$
 - $\mathbf{x}_* = X$ is $N \times D$
 - β_j is $D \times 1$
- So, this will work if we transpose X and premultiply it and define g as a function that accepts \mathbf{X} .

$$\beta_j \leftarrow \beta_j + \alpha \mathbf{X}^T (t_{*,j} - g(\mathbf{X}, \beta_j))$$

- Let's keep going...and try to make this expression work for all of the β 's.

- Playing with the subscripts again, replace j with $*$

$$\beta_j \leftarrow \beta_j + \alpha \mathbf{X}^T (t_{*,j} - g(\mathbf{X}, \beta_j))$$
$$\beta_* \leftarrow \beta_* + \alpha \mathbf{X}^T (t_{*,*} - g(\mathbf{X}, \beta_*))$$

- Now what are shapes?
 - $\beta_* = \beta$ is $D \times K - 1$
 - $t_{*,*} = T$ is $N \times K - 1$
 - $g(\mathbf{X}, \beta)$ is $N \times K - 1$
 - $t_{*,*} - g(\mathbf{X}, \beta)$ is $N \times K - 1$
 - So, $\mathbf{X}^T (T - g(\mathbf{X}, \beta))$ is $D \times K - 1$
- Now our update equation for all β 's is

$$\beta \leftarrow \beta + \alpha \mathbf{X}^T (T - g(\mathbf{X}, \beta))$$

- Update is

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha \mathbf{X}^T (T - g(\mathbf{X}, \boldsymbol{\beta}))$$

and we had defined

$$f(\mathbf{x}_n, \boldsymbol{\beta}_k) = e^{\boldsymbol{\beta}_k^T \mathbf{x}_n}$$
$$g(\mathbf{x}_n, \boldsymbol{\beta}_k) = \frac{f(\mathbf{x}_n, \boldsymbol{\beta}_k)}{1 + \sum_{m=1}^{K-1} f(\mathbf{x}_n, \boldsymbol{\beta}_m)}$$

- Changing these to handle all samples \mathbf{X} and all parameters $\boldsymbol{\beta}$ we have

$$f(\mathbf{X}, \boldsymbol{\beta}) = e^{\mathbf{X}\boldsymbol{\beta}}$$
$$g(\mathbf{X}, \boldsymbol{\beta}) = \frac{f(\mathbf{X}, \boldsymbol{\beta})}{1 + \sum_{m=1}^{K-1} f(\mathbf{X}, \boldsymbol{\beta}_m)}$$

- To summarize, given training data \mathbf{X} ($N \times D$) and class indicator variables T ($N \times K - 1$), the expressions

$$f(\mathbf{X}, \beta) = e^{\mathbf{X}\beta}$$

$$g(\mathbf{X}, \beta) = \frac{f(\mathbf{X}, \beta)}{1 + \sum_{m=1}^{K-1} f_m(\mathbf{X}, \beta)}$$

$$\beta \leftarrow \beta + \alpha \mathbf{X}^T (T - g(\mathbf{X}, \beta))$$

can be performed with

```
g <- function(X,beta) {
  fs <- exp(X %*% beta) # N x K-1
  denom <- 1 + rowSums(fs)
  N <- nrow(X)
  K1 <- ncol(beta)
  gs <- fs / matrix(rep(denom,K1),N,K1)
  cbind(gs,1/denom)
}
### Xtrain is standardized and has a 1 column.
### Ttrain is matrix of indicator variables .
for (steps in 1:1000) {
  gs <- g(Xtrain,beta)
  beta <- beta + alpha * t(Xtrain) %*% (Ttrain[,-K] - gs[,-K])
  likelihoodPerSample <- exp( sum(Ttrain * log(gs)) / nrow(Xtrain))
}
```