Non-parametric Logistic Regression

December 6, 2020

1 Abstract

Two techniques to construct non-parametric logistic regression are described:

1. basis expansion using natural splines
2. regularization penalizing curvature of the resulting function

Straightforward implementation of these techniques leads to technical complication: computations are too slow due to numerical integration in the Newton-Raphson routine. To overcome this issue, we introduce a way to perform closed-form integration instead of numerical integration for both one-dimensional and multidimensional case. Package performing efficient estimations for one-dimensional non-parametric logistic regression is implemented and used for simulations. The package includes *Splines.py* and *NonparametricLogisticRegression.py*

2 Description and Simulations - One-Dimensional Case

2.1 Logistic Regression Using Basis Expansion

Let's start from classical logarithmic setting.

$$\log \frac{P(Y=1)|X=x)}{P(Y=0|X=x)} = f(x) \Rightarrow P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}$$

The simplest form of $f(x)$ is linear, but we lose a lot of flexibility by using the linear form. So let's make $f(x)$ a natural cubic spline and introduce new **basis expantion** $N_i(x)$:

$$f(x) = \sum_{i=1}^{M} \theta_i N_i(x)$$

$$N_1(x) = 1$$

$$N_2(x) = x$$

$$N_{k+2}(x) = d_k(x) - d_{M-1}(x)$$

$$d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_M)_+^3}{\xi_M - \xi_k}$$

We will use notation $p(x_i) = P(Y = 1|X = x_i)$. Coefficients θ_i might be obtained from maximization of the log-likelihood:

$$l(f) = \sum_{i=1}^{N} y_i \log p(x_i) + (1-y_i)\log(1-p(x_i)) =$$

$$= \sum_{i=1}^{N} y_i f(x_i) - \log(1 + e^{f(x_i)}) \to \max_{\theta}$$

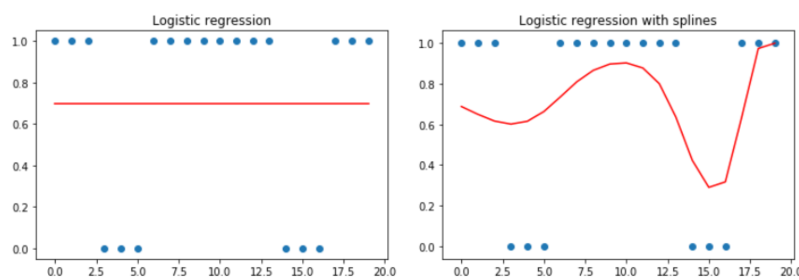
Newton-Raphson method is used to find optimal θ. This method requires knowledge of $\frac{\partial l(\theta)}{\partial \theta}$ and $\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T}$:

$$\frac{\partial l(\theta)}{\partial \theta} = N^T(y-p)$$

$$\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} = -N^T W N,$$

where N is the matrix of $N_j(x_i)$, p is the vector of $p(x_i)$ - predictions at the current iteration, and W is a diagonal matrix of wights $p(x_i)(1-p(x_i))$.

Let's compare simple logistic regression having linear form of $f(x)$ with simple logistic regression using splines as basis expansion for $f(x)$:



Using splines does provide more flexibility. But as one can see from the picture above, there are regions with overfit.

2.2 Logistic Regression Using Basis Expansion and Regularization

So let's introduce **regularization** to make results more robust. We construct penalized log-likelihood:

$$l(f) = \sum_{i=1}^{N} y_i f(x_i) - \log(1 + e^{f(x_i)}) - \frac{1}{2}\lambda \int (f''(t))^2 dt$$

Note that since we know the closed form of $f(x)$, we can penalize curvature of the function f by integrating the second derivative of f.

In this case, $\frac{\partial l(\theta)}{\partial \theta}$ and $\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T}$ for Newton-Raphson method will take the form:

$$\frac{\partial l(\theta)}{\partial \theta} = N^T(y - p) - \lambda \Omega \theta$$
$$\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} = -N^T W N - \lambda \Omega,$$

where matrix Ω is the matrix of integrals of the second derivatives of basis functions: $\Omega_{jk} = \int N_j''(x)^T N_k''(x) dx$.

At this point, we have all means to implement techniques of basis expansion with splines and regularization for one-dimensional case, but we face a **technical complication**: to find matrix Ω, we should find integrals of functions $N_j''(x)^T N_k''(x)$. If these calculations are performed numerically, then it will take **significant amount of time** even for modest amount of training data (remember that functions $N_j(x)$ are defined using training data). Additionally, for multidimensional case there will be necessity to calculate integrals over multidimensional space of arguments, which further complicates calculations.

This complication can be overcome by the fact that we know exactly the closed form of $N_j(x)$, and thus we can find analytic expression for $\int N_j''(x)^T N_k''(x) dx$. Let's first write out expression for $N_j''(x)^T N_k''(x)$, where $j, k > 2$:

$$N_j''(x)^T N_k''(x) =$$

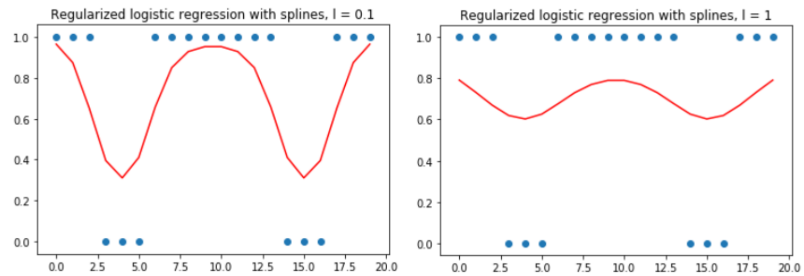
$$36\Big[\frac{(x-\xi_{j-2})_+(x-\xi_{k-2})_+ - (x-\xi_{j-2})_+(x-\xi_M)_+ - (x-\xi_M)_+(x-\xi_{k-2})_+ + (x-\xi_M)_+^2}{(\xi_M - \xi_{j-2})(\xi_M - \xi_{k-2})} -$$
$$- \frac{(x-\xi_{M-1})_+(x-\xi_{k-2})_+ - (x-\xi_{M-1})_+(x-\xi_M)_+ - (x-\xi_M)_+(x-\xi_{k-2})_+ + (x-\xi_M)_+^2}{(\xi_M - \xi_{M-1})(\xi_M - \xi_{k-2})} -$$
$$- \frac{(x-\xi_{j-2})_+(x-\xi_{M-1})_+ - (x-\xi_{j-2})_+(x-\xi_M)_+ - (x-\xi_M)_+(x-\xi_{M-1})_+ + (x-\xi_M)_+^2}{(\xi_M - \xi_{j-2})(\xi_M - \xi_{M-1})} +$$
$$+ \frac{(x-\xi_{M-1})_+ - 2(x-\xi_{M-1})_+(x-\xi_M)_+ + (x-\xi_M)_+^2}{(\xi_M - \xi_{M-1})^2}\Big]$$

Now it is easy to find the integral $\int N_j''(x)^T N_k''(x) dx$, since ξ_j are known observations and resulting expression is the sum of integrals of the form

$$\int \frac{(x-a)_+(x-b)_+}{c} dx$$

This approach is applicable for the case of multidimensional X as well - see the next section

Now let's see how this model performs on initial dataset with various levels of λ:



We see that regularized logistic regression with splines performs much better and computations are fast with integration performed in the closed form.

3 Conceptual Description - Multidimensional Case

We can apply the closed-form-integration approach from the previous section for the case of multidimensional X. For simplicity, we describe two-dimensional case only, but this approach can be generalized for arbitrary amount of dimensions.

We are now interested in the efficient calculations for the matrix Ω. In the two-dimensional case basis expansion for the function $f(x_1, x_2)$ takes the form

$$f(x_1, x_2) = \sum_i \sum_j \theta_{ij} N_i^{(1)}(x_1) N_j^{(2)}(x_2) = \sum_\alpha \theta_\alpha N_{i_\alpha}^{(1)}(x_1) N_{j_\alpha}^{(2)}(x_2)$$

Note that α is used to just reindex expression in a convenient way.

Log-likelihood with regularization will take the form:

$$l(f) = \sum_{i=1}^{N} y_i f(x_1^i, x_2^i) - \log(1 + e^{f(x_1^i, x_2^i)}) - \frac{1}{2}\lambda \int\int \Big[\Big(\frac{\partial^2 f}{\partial t^2}\Big)^2 + 2\Big(\frac{\partial^2 f}{\partial t \partial s}\Big)^2 + \Big(\frac{\partial^2 f}{\partial s^2}\Big)^2\Big] dtds$$

$$\Big(\frac{\partial^2 f(t,s)}{\partial t^2}\Big)^2 = \Big(\sum_\alpha \theta_\alpha N_{i_\alpha}^{(1)\prime\prime}(t) N_{j_\alpha}^{(2)}(s)\Big)^2$$

$$\Big(\frac{\partial^2 f(t,s)}{\partial t \partial s}\Big)^2 = \Big(\sum_\alpha \theta_\alpha N_{i_\alpha}^{(1)\prime}(t) N_{j_\alpha}^{(2)\prime}(s)\Big)^2$$

$$\Big(\frac{\partial^2 f(t,s)}{\partial s^2}\Big)^2 = \Big(\sum_\alpha \theta_\alpha N_{i_\alpha}^{(1)}(t) N_{j_\alpha}^{(2)\prime\prime}(s)\Big)^2$$

In multidimensional case, $\frac{\partial l(\theta)}{\partial \theta}$ and $\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T}$ for Newton-Raphson method will take the form:

$$\frac{\partial l(\theta)}{\partial \theta} = N^T(y - p) - \lambda \Omega \theta$$

$$\frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} = -N^T W N - \lambda \Omega,$$

where N is the matrix of $N_{km} = N^{(1)}_{i_m}(x^k_1)N^{(2)}_{j_m}(x^k_2)$, p is the vector of $p(x^i_1, x^i_2)$ - predictions at the current iteration, and W is a diagonal matrix of wights $p(x^i_1, x^i_2)(1 - p(x^i_1, x^i_2))$. θ is a vector of θ_α. Bottleneck of this routine is finding integrals of the matrix Ω over multidimensional space, but this task can be reduced to integration over one-dimensional space:

$$\Omega_{km} = \Big[\frac{\partial^2}{\partial\theta\partial\theta^T}\int\int\frac{1}{2}\Big[\Big(\frac{\partial^2 f}{\partial t^2}\Big)^2 + 2\Big(\frac{\partial^2 f}{\partial t\partial s}\Big)^2 + \Big(\frac{\partial^2 f}{\partial s^2}\Big)^2\Big]dtds\Big]_{km} =$$

$$= \int\int\big(N^{(1)}_{i_k}{''}(t)N^{(2)}_{j_k}(s)\big)\big(N^{(1)}_{i_m}{''}(t)N^{(2)}_{j_m}(s)\big)dtds +$$

$$+ \int\int\big(N^{(1)}_{i_k}{'}(t)N^{(2)}_{j_k}{'}(s)\big)\big(N^{(1)}_{i_m}{'}(t)N^{(2)}_{j_m}{'}(s)\big)dtds +$$

$$+ \int\int\big(N^{(1)}_{i_k}(t)N^{(2)}_{j_k}{''}(s)\big)\big(N^{(1)}_{i_m}(t)N^{(2)}_{j_m}{''}(s)\big)dtds =$$

$$= \Big(\int N^{(1)}_{i_k}{''}(t)N^{(1)}_{i_m}{''}(t)dt\Big)\Big(\int N^{(2)}_{j_k}(s)N^{(2)}_{j_m}(s)ds\Big) +$$

$$+ \Big(\int N^{(1)}_{i_k}{'}(t)N^{(1)}_{i_m}{'}(t)dt\Big)\Big(\int N^{(2)}_{j_k}{'}(s)N^{(2)}_{j_m}{'}(s)ds\Big) +$$

$$+ \Big(\int N^{(1)}_{i_k}(t)N^{(1)}_{i_m}(t)dt\Big)\Big(\int N^{(2)}_{j_k}{''}(s)N^{(2)}_{j_m}{''}(s)ds\Big)$$

To find one-dimensional integrals we use the same method from the previous section, i.e. we find closed-form expression of all integrals involved. Generalization to arbitrary amount of dimensions can be performed in the same manner as in the two-dimensional case.

Implementing this routine for arbitrary amount of dimensions takes significant amount of tensor algebra. But since finding closed-form expressions for integrals calculations involve only elementary operations, this implementation would take reasonable time for execution.