Supplementary Material for “Sparse Functional Dynamical Models — a Big Data Approach”

Ela Sienkiewicz¹, Dong Song², F. Jay Breidt¹ and Haonan Wang¹

¹Department of Statistics, Colorado State University

²Center for Neural Engineering, University of Southern California

August 1, 2016

A Pseudo-Code

R Packages implementing the code in this paper are located at

<http://www.stat.colostate.edu/~wanghn/Neuron.html>.

A.1 Maximum-Likelihood Estimation

A.1.1 Iteratively Reweighted Least Square

Iteratively reweighted least squares (IRLS) is a standard algorithm used to compute a maximum likelihood estimate of a generalized linear model (10); see McCullagh and Nelder (1983) for details. The algorithm is easy to implement, and the code is available at numerous sources, e.g., Komarek (2004), Givens and Hoeting (2012) among others. Here, we present the pseudo-code to highlight matrix-vector operations involved in the implementation. Matrix-vector operations is where our approach diverges from the standard implementation.

Listing 1: Iteratively Reweighted Least Squares

```

1  input:   $X_1, y, \beta_0$ 
2  output:  $\hat{\beta}$  which maximizes the likelihood function (8).
3  # initialize mean and deviance
4   $\mu = (y + 0.5)/2$ 
5   $dev_{-1} = \infty$ 
6   $dev_0 = \text{deviance}(\beta_0)$ 
7   $\eta = \Phi^{-1}(\mu)$ 
8
9  for  $s = 1, 2, \dots$ 
10      $w = (\phi(\eta)^2)/(\mu * (1 - \mu))$  # diagonal of matrix of weights W
11      $z = \eta + (y - \mu)/\phi(\eta)$ 
12      $b = -2 [1 \ X]^T (w * z)$ 
13      $\hat{\beta} = \text{argmin}_{\beta} \frac{1}{2} \beta [1 \ X]^T W [1 \ X] \beta - \beta^T b$  # solve with e.g., conjugate gradient
14      $\eta = [1 \ X] \hat{\beta}$ 
15      $\mu = \Phi(\eta)$ 
16      $dev_s = \text{deviance}(\hat{\beta})$ 
17     if  $|dev_s - dev_{s-1}| < \epsilon$  break

```

Here, ϕ and Φ represent the normal density and cumulative distribution respectively. The only directly-called matrix-vector operations are in lines 12 and 14. They are executed in every iteration.

A.1.2 Conjugate Gradient

Conjugate gradient is an optimization method developed for solving linear equations problem $\mathbf{H}x = b$, where $\mathbf{H}$ is a symmetric, positive-definite matrix; see Hestenes and Stiefel (1952) for details. The problem is equivalent to a quadratic optimization of the type: $\min_x (\frac{1}{2}x^T \mathbf{H}x - x^T b)$. The solution is based on conjugate vectors $d_1, \dots, d_p$ such that $d_i^T \mathbf{H}d_j = 0$, for $i \neq j$. The algorithm is thoroughly explained in Komarek (2004) and Antoniou and Lu (2007), among others. Here, $\mathbf{H}$ is a quadratic $p \times p$ matrix, calculated as $\mathbf{H} = 2\mathbf{Z}\mathbf{W}\mathbf{Z}$, where $\mathbf{Z} = [1 \ \mathbf{X}]$. As in the previous case, matrix-vector operations is where our approach

diverges from the standard implementation.

Listing 2: Congugate Gradient

```

1  input:   $X_1, w, b, x_0$ 
2  output:  $x_n$  which minimizes  $\frac{1}{2} x^T [1 \ X]^T W [1 \ X] x - x^T b$  over  $x$ 
3
4  # initialize
5   $k = 0$ 
6   $x_k = x_0$           #  $x_k$  is a solution at step  $k$ 
7   $h_{x_k} = 2 [1 \ X]^T W [1 \ X] x_k$   #  $Hx_k$ 
8   $g_k = (b + Hx_k)$     #  $g_k$  is a gradient at step  $k$ 
9   $d_k = -g_k$           #  $d_k$  is the  $k$ th conjugate direction
10
11 repeat
12      $h_{d_k} = 2 [1 \ X]^T W [1 \ X] d_k$   #  $Hd_k$ 
13      $\alpha_k = (g_k^T g_k) / (d_k^T h_{d_k})$ 
14      $x_n = x_k + \alpha_k * d_k$ 
15      $dev = |\alpha_k|^T |d_k|$ 
16     if ( $dev < \epsilon$ ) break;
17      $h_{x_n} = h_{x_k} + \alpha_k * h_{d_k}$ 
18      $g_n = b + h_{x_n}$ 
19      $\beta_k = (g_n^T g_n) / (g_k^T g_k)$ 
20      $d_n = -g_n + \beta_k * d_k$ 
21     # update counters
22      $k = k + 1; d_k = d_n; g_k = g_n; x_k = x_n; h_{x_k} = h_{x_n}$ 

```

The only matrix-vector operations are in lines 7 and 12. The first operation is executed once, the second one in each iteration. The execution of code in Listings 1 and 2 leads to the computation of the maximum likelihood estimator of model parameters.

A.2 Regularized Estimation

A.2.1 Group Bridge Regularization

Listing 3: Group Bridge Regularization

```

1  input:   $X_1, y, w,$ 
2           $\alpha_0$  - maximum likelihood estimate
3           $A, c, \gamma, \tau_n$  - as required by Group Bridge method (Huang et al., 2009)
4  output:   $\alpha^{(s)}$  which minimizes (9)
5
6  G = number of groups and rows of  $A$ 
7  P = number of parameters and columns of  $A$ 
8
9  for  $s = 1, 2, \dots$ 
10     for  $g$  in  $1..G$ 
11         #  $\theta_g = c_g(\frac{1-\gamma}{\tau_n\gamma})^\gamma * \sum_{i:\alpha_i \in A_g} |\alpha_i^{(s-1)}|^\gamma$ 
12          $\theta[g] = c[g] * (\frac{1-\gamma}{\tau*\gamma})^\gamma * \sum |\alpha^{(s-1)}[A[g, ]]|^\gamma$ 
13
14     for  $p$  in  $1..P$ 
15         #  $\lambda_p = \sum_{g:\alpha_p \in A_g} \theta_g^{1-1/\gamma} c_g^{1/\gamma}$ 
16          $\lambda[p] = \sum \{(\theta[A[, p]])^{1-1/\gamma} * (c[A[, p]])^{1/\gamma}\}$ 
17
18     # (lasso problem) solve with e.g., coordinate_descent
19      $\alpha^{(s)} = \operatorname{argmin}_\alpha ||W^{\frac{1}{2}}(y - X\alpha)||^2 + \sum_p \lambda_p |\alpha_p|$ 
20     if  $||\alpha^{(s)} - \alpha^{(s-1)}||_2^2 < \epsilon$  break

```

A.2.2 Coordinate Descent

Matrix-vector operations is where our approach diverges from the standard implementation of Coordinate Descent algorithm available in Friedman et al. (2007, 2010).

Listing 4: Coordinate Descent

```

1  input:   $X_1, y, w,$ 
2           $\lambda$  - penalty vector
3           $\alpha^0$  - maximum likelihood estimate
4  output:  $\hat{\alpha}$  which minimizes  $\|W^{\frac{1}{2}}(y - X\alpha)\|_2^2 + \sum \lambda_j |\alpha_j|$ 
5
6  N = number of rows of  $X$ 
7  p = number of columns of  $X_1$ 
8  p2 = number of columns of  $X$  #  $p+(p*(p-1))/2$ 
9  covyx =  $[1 \ X]^T(w * y)$ 
10 for  $s = 1, 2, \dots$ 
11      $\alpha^{(s)} = \alpha^{(s-1)}$ 
12     for  $j = 2, \dots, p2$ 
13          $x_j$  = jth column of  $X$ 
14         covxxj =  $[1 \ X]^T(w * x_j)$ 
15          $v = \text{covyx}_j - \text{covxxj}^T \alpha_s$ 
16          $\alpha_j^{(s)} = S(v + \alpha_j^{(s)} \text{covxxj}_j, \lambda_j) / \text{covxxj}_j$ 
17     if  $\|\alpha^{(s)} - \alpha^{(s-1)}\|_2^2 < \epsilon$  break

```

Function $S(\cdot, \cdot)$ used in line 16 is the threshold function introduced in Section 4.5. The only matrix-vector operations are in lines 9 and 14. The first operation is executed once, the second – in each iteration.

A.3 Matrix-Vector Operations

There are three types of matrix-vector operations referenced in the preceding code. They require special handling due to the size of matrix $\mathbf{X}$. All other code is safely executed in memory of a standard computer. The matrix-vector operations are as follows: $[1 \ \mathbf{X}]^T \mathbf{v}$, $[1 \ \mathbf{X}] \mathbf{v}$ and $[1 \ \mathbf{X}]^T \mathbf{W} [1 \ \mathbf{X}] \mathbf{v}$, for a matrix X , a vector $\mathbf{v}$ and a diagonal matrix of weights W , represented by its diagonal vector $\mathbf{w}$. A flexible implementation of algorithms in all above listings would consider using matrix-vector operations as three functional parameters. That way one can plug-in a matrix-handling code that is appropriate for the size and the

location of the data. In our case, these three methods operate on a partial (first-order) matrix available in the memory of a single multi-core computer. The mappers and reducers for these operations are symbolically described in Equations (17). Mappers execute the following operations respectively:

$$\begin{aligned}
\mathbf{x}_k^T \mathbf{w}_k \mathbf{x}_k \boldsymbol{\beta} &= [(\mathbf{x}_k^{(1)})^T \mathbf{w}_k \mathbf{a}_k, (\mathbf{x}_k^{(1)} \otimes' \mathbf{x}_k^{(1)})^T \mathbf{w}_k \mathbf{a}_k] \\
\mathbf{a}_k &= (\mathbf{x}_k^{(1)})^T \boldsymbol{\beta}^{(1)} + (\mathbf{x}_k^{(1)} \otimes' \mathbf{x}_k^{(1)})^T \boldsymbol{\beta}^{(2)} \\
\mathbf{x}_k \mathbf{w}_k \mathbf{y}_k &= [\mathbf{x}_k^{(1)}, (\mathbf{x}_k^{(1)} \otimes' \mathbf{x}_k^{(1)})] \mathbf{w}_k \mathbf{y}_k \\
\mathbf{x}_k^T \mathbf{w}_k \boldsymbol{\beta} &= (\mathbf{x}_k^{(1)})^T \mathbf{w}_k \boldsymbol{\beta}^{(1)} + (\mathbf{x}_k^{(1)} \otimes' \mathbf{x}_k^{(1)})^T \mathbf{w}_k \boldsymbol{\beta}^{(2)}
\end{aligned}$$

Reducers execute summation (case one and two) and concatenation (case three). Symbol $\otimes'$ indicates a modified Kronecker product. In our application, not all components participate in the interactions, the input x_{d+1} (the delayed output signal) is excluded. Most applications don't need this sort of distinction and the Kronecker product of all inputs is even simpler to compute.

B Additional Figures

The context for Figure A is provided in Section 5.3. Figure A shows the estimated interaction components, i.e., the second-order kernels, of the second-order model. Only four second-order kernels are shown, φ_{11} , φ_{37} , φ_{57} and φ_{67} . The remaining kernels exhibit global sparsity, which suggests no interaction exists between corresponding input neurons or within the same input neurons at different time lags.

The context for Figure B is provided in Section 5.4. The discrete time rescaling theorem proposed by (Haslinger et al., 2010) allows us to transform estimated probabilities $\theta(t)$ to the uniform distribution. The resulting KS plot compares the transformed data to the actual uniform distribution. A small bias is visible, as described in details in Section 5.4.

Figure A: Penalized estimation of interaction components in the second-order model. Only non-zero components are displayed.

References

- Antonioni, A. and Lu, W. (2007). *Practical Optimization*. Springer.
- Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.
- Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1.
- Givens, G. H. and Hoeting, J. A. (2012). *Computational Statistics*, volume 708. John Wiley & Sons.
- Haslinger, R., Pipa, G., and Brown, E. (2010). Discrete time rescaling theorem: determining goodness of fit for discrete time statistical models of neural spiking. *Neural Computation*, 22(10):2477–2506.

Figure B: KS plot for rescaled firing probabilities against a reference distribution (uniform). Rescaled firing probabilities are depicted with a thick dashed line, reference uniform distribution with a dotted line, and a 95% confidence interval with a thin solid line. Small bias is visible for low probabilities, more evident in the zoom-in subplot.

Hestenes, M. R. and Stiefel, E. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS.

Huang, J., Ma, S., Xie, H., and Zhang, C. (2009). A group bridge approach for variable selection. *Biometrika*, 2:339–355.

Komarek, P. (2004). Logistic regression for data mining and high-dimensional classification. *Robotics Institute*, pages 1–222.

McCullagh, P. and Nelder, J. A. (1983). *Generalized linear models*, volume 37. London: Chapman and Hall.