REAL-TIME CONJUGATE GRADIENTS FOR ONLINE FMRI CLASSIFICATION

Hao Xu∗, Yongxin Taylor Xi∗, Ray Lee†, Peter J. Ramadge∗

∗Dept. of Electrical Engineering, †Princeton Neuroscience Institute, Princeton University, Princeton NJ

ABSTRACT
Real-time functional magnetic resonance imaging (rtfMRI) enables classification of brain activity during data collection thus making inference results accessible to both the subject and experimenter during the experiment. The major challenge of rtfMRI is the potential loss of inference accuracy due to the resource limitations that rtfMRI imposes. For example, many widely-used analysis methods in off-line neuroimaging are too time-consuming for rtfMRI. We develop an online, real-time, conjugate gradient (rtCG) algorithm that learns to classify brain states as data is being collected. The algorithm is closely connected to partial least squares (PLS), a popular off-line analysis method. We give a theoretical comparison with PLS and show that the algorithm generates identical results to PLS for appropriate initial conditions. However, in practice using an alternative initial condition yields faster convergence. Experimental results show that the online rtCG classifier: is fast (training time < 0.5s), is accurate (prediction accuracy ≈ 90%), can adapt to a varying stimulus, and yields better classification performance than standard PLS applied to a sliding window of recent data.

Index Terms— Online learning, Partial Least Squares, Conjugate Gradient, fMRI classification

1. INTRODUCTION
Functional magnetic resonance imaging (fMRI) is a noninvasive technique to indirectly investigate brain activity based on the blood oxygenation level-dependent (BOLD) signal. Brain activity is recorded as a time sequence of 3-D images allowing exhibited patterns under different tasks or stimuli to be visualized and studied statistically. The problem of interest is to train a classifier to predict the brain activity given a particular 3-D brain image, e.g. is the subject is looking at an image of a face or of a house. Conventionally, this is done off-line after the data is acquired. In contrast, in real-time fMRI (rtfMRI), inference (e.g. classification) on brain states is done concurrently with data acquisition. rtfMRI offers several potential advantages: a) the possibility of directly reading a subject’s cognitive state in real-time; b) monitoring data quality and locating regions of interest as data are collected; c) providing feedback to the subject based on the output of a real-time classifier; d) adaptively changing an experimental stimulus based on real-time feedback to achieve better activation. LaConte et al. [1] have shown that a subject can control the movement of an arrow on a computer screen using brain state classification by rtfMRI. Other applications of rtfMRI are described in [2].

Cox [3] first proposed rtfMRI and implemented a system that computed an activation map from the fMRI data within 500ms of the RF pulse. Goddard [4], Frank [5], and Bagarinao [6] subsequently applied parallel and cluster computing to rtfMRI to speed up the processing tasks. Attention has recently shifted to more effective and efficient algorithms with improved prediction accuracy. LaConte [1] reported a rtfMRI system employing a support vector machine (SVM) algorithm (trained offline) to classify and predict brain states and Esposito [7] developed a real-time independent component analysis (ICA) method to identify brain activity.

Here we consider online algorithms that both learn and classify as data are collected. The main challenges are: a) the algorithm must learn online from high dimensional labelled fMRI data (122,880 voxels per TR or higher), b) it must accurately predict brain states, c) it must do both a) and b) within tight time constraints (one TR = 2-3 secs), it must handle nuisance trends in the data, and it should adapt to changes in the experiment and in the subject. Our contribution is a new online learning algorithm that attains competitive classification accuracy within the constraints required. We demonstrate its effectiveness in actual real-time fMRI tests and give a theoretical connection to PLS.

The paper is organized as follows: §2 gives a detailed description of the rtfMRI problem and specifies the proposed real-time conjugate gradient online learning algorithm for rtfMRI, including implementation details and a theoretical comparison with the widely-used partial least squares algorithm. §3 presents our experimental results.

2. REAL-TIME FMRI
A study of rtfMRI is timely both because of the success of off-line regression and classification for fMRI data, e.g. [8], and the recent progress in on-line learning algorithms. Partial least squares (PLS) regression [9, 10], is one successful off-line method for identifying task-dependent changes in brain activity and for examining functional connectivity of brain regions [11]. On-line versions of many successful machine learning methods have been explored and these have potential
application in a rtfMRI system: [6] adapted the general linear model (GLM) for real-time application; [7] applied independent component analysis (ICA) in a sliding data window; [3] implemented an incremental version of the correlation technique; and in a companion paper [12], we examine how to adapt support vector machines (SVM) for rtfMRI.

In this paper we seek an efficient online algorithm with comparable performance to PLS. PLS regression (with a single output variable) assumes that the input data \( X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times k} \) and output data \( y = [l_1, \ldots, l_n]^T \in \mathbb{R}^n \) are both generated by a small number, \( m \), of latent factors:

\[
X = \sum_{i=1}^{m} f_i p_i^T + D = FP^T + D, \quad y = \sum_{i=1}^{m} f_i q_i + e = Fq + e.
\]

In fMRI, \( x_i \) is the 3D image at the \( i^{th} \) TR (rearranged as a column) and \( l_i \) is its class label. The columns of \( F = [f_1, \ldots, f_m] \in \mathbb{R}^{n \times m} \) are the latent factors, the columns of \( P = [p_1, \ldots, p_m] \in \mathbb{R}^{k \times m} \) and entries of \( q = [q_1, \ldots, q_m]^T \in \mathbb{R}^m \) are the factor loadings for \( X \) and \( y \). \( D \in \mathbb{R}^{n \times k} \) and \( e \in \mathbb{R}^n \) are residuals. PLS iteratively finds the latent factors \( f_i = X w_i/\|X w_i\| \), where \( w_i = X_i^T y_i/\|X_i^T y_i\| \in \mathbb{R}^k \), that maximize the covariance between input and output. Then the loading vectors are found by performing univariate regressions \( p_i^T = f_i^T X_i q_i = f_i^T y_i \). After one factor is extracted, a rank one deflation of \( X \) and \( y \) is performed:

\[
X_{i+1} = X_i - f_i p_i^T, \quad y_{i+1} = y_i - f_i q_i.
\]

This procedure is repeated until \( m \) orthonormal factors are extracted. To generate the regression coefficient, PLS uses the latent factor approximations of \( X \) and \( y \). Thus the final regression problem is to construct \( v \) such that \( \arg \min_v \| (FP^T) W v - F q \|^2 \), where \( w_i \) are columns of matrix \( W \in \mathbb{R}^{k \times m} \). Since \( P^T W \) is a upper bidiagonal and nonsingular [13], we have \( \hat{v} = (P^T W)^{-1} q \) and regression coefficient \( b^{PLS} = W \hat{v} = (P^T W)^{-1} q \).

PLS is computationally too expensive for online rtfMRI.

A more efficient algorithm for streamed data, Incremental Sparse Bridge PLS (iS-BPLS) [14], is also computationally expensive on high-dimensional fMRI data. The PLS algorithm is a conjugate gradient (CG) algorithm [15, 16]. This motivates our study of a CG based algorithm for online rtfMRI.

Our online learning algorithm has two steps: 1) at time \( t \), example \( x_t \in \mathbb{R}^k \) arrives and the classifier predicts its label: \( l_t \in \{-1, 1\} \); 2) the actual label \( l_t \in \{-1, 1\} \) of \( x_t \) is then provided and the classifier is updated. In this study, we consider the binary linear classifier \( b_t \in \mathbb{R}^k \) with label prediction

\[
l_t = \text{sign}(x_t^T b_{t-1}).
\]

\( b_{t-1} \) can depend on the labeled data up to time \( t-1 \) and is updated to \( b_t \) when \( x_t \) and \( l_t \) are available. Bring in a window parameter \( h \) and let \( X^h_t = [x_{t-h+1}, \ldots, x_t] \) and \( y^h_t = [l_{t-h+1}, \ldots, l_t] \). \( b_{t-1}, X^h_t, \) and \( y^h_t \) are the inputs to the update step at time \( t \) that yields \( b_t \). The data window \( X^h_t \) limits the amount of high dimensional data used, this mediates the computation resources (time and space) required, and since it uses only the most recent data it enables the classifier to adapt to changes. In this setting, consider the update rule:

\[
b_t = \arg \min_{b \in \mathbb{R}^k} \| X^h_t b - y^h_t \|^2.
\]

We note that (2) is a quadratic program. Standard CG solves (2) iteratively in a finite number of steps by finding conjugate search directions \( d_i \) with \( d_i^T (X^h_t)^T X^h_t d_j = 0, i \neq j \). In each iteration, CG first updates the current estimate \( g_i \) by minimizing the objective in direction \( d_i \), \( g_{i+1} = g_i + \alpha_i d_i \), where \( \alpha_i = r_i^T r_i / d_i^T (X^h_t)^T X^h_t d_i \) is the step size. Then CG updates the residual \( r_{i+1} = r_i - \alpha_i (X^h_t)^T X^h_t d_i \) and finds a new conjugate search direction \( d_{i+1} = r_{i+1} + \theta_i d_i \), where \( \theta_i = r_{i+1}^T r_{i+1} / r_i^T r_i \). After \( k \) iterations, CG solves (2) with \( b_t = g_k \). The change from \( X^h_{t-1} \) to \( X^h_t \) involves removing a column and adding a column. So we expect \( b_{t-1} \) to be close to \( b_t \). Hence \( b_{t-1} \) is used to initialize the CG iterations at time \( t \), i.e. \( g_0 = b_{t-1} \) at time \( t \). Note that the windowed data represents the recent information and \( b_{t-1} \) encodes past information. CG solves (2) in \( k \) steps but for fMRI data \( k \) is very large (\( \approx 10^5 \)). Hence we limit the number of iterations to \( I_{max} \ll k \). \( I_{max} = 0 \), yields \( b_t = b_{t-1} \) and the recent information is ignored; \( I_{max} = k \), solves (2) using only the recent information. Hence \( I_{max} \) balances past memory with learning from recent data.

**Algorithm 1 Real-time CG (rtCG) Algorithm**

1. Initialize: \( b_0^{rtCG} = 0 \)
2. at time \( t \): Let \( \beta = X^h_t y_t^h, A = X^h_t X^h_t \) and \( r_0 = d_0 = \beta - A g_0 \), \( g_0 = b_0^{rtCG} \)
   while \( i \leq I_{max} \) & residual \( \geq \) tolerance do
   \( g_{i+1} = g_i + \alpha_i d_i \), where \( \alpha_i = r_i^T r_i / d_i^T A d_i \)
   \( r_{i+1} = r_i - \alpha_i A d_i \)
   \( d_{i+1} = r_{i+1} + \theta_i d_i \), where \( \theta_i = r_{i+1}^T r_{i+1} / r_i^T r_i \)
   end while
3. Output: \( b_t^{rtCG} = g_t \)

We now show that if \( g_0 = 0 \), then the iterations of rtCG are identical to the steps of PLS applied to \( X^h_t \) and \( y^h_t \).

**Theorem 1.** Assuming \( g_0 = 0 \), rtCG and PLS yield identical steps applied when applied to \( X^h_t \) and \( y^h_t \).

**Proof.** Let \( X \) and \( y \) denote the training examples and labels at time \( t \) available to both rtCG and PLS. For rtCG with \( g_0 = 0 \), by induction of the while loop in Algorithm 1 we have \( g_i = \sum_{l=0}^{i-1} d_l (d_l^T A d_l)^{-1} d_l \beta \), where \( A = X^T X, \beta = X^T y \). Since \( d_l \)'s are conjugate directions, i.e., \( d_l^T A d_j = 0 \) for \( i \neq j \), we can rewrite this in a matrix form as:

\[
g_i = D (D^T A D)^{-1} D^T \beta
\]

where \( D = [d_0, \ldots, d_{i-1}] \). By a property of CG [16], the conjugate directions span a Krylov subspace:

\[
\text{span} \{d_0, d_1, \ldots, d_{i-1}\} = \text{span} \{\beta, A \beta, \ldots, A^{i-1} \beta\}
\]
For PLS, [13] gives:

\[ b_i^{PLS} = \tilde{W}(\tilde{W}^T A \tilde{W})^{-1} \tilde{W}^T \beta, \]

(5)

where \( \tilde{W} \) is any matrix with range \( W \)-range \( \tilde{W} \). One such matrix is \([\beta, A\beta, \ldots, A^{r-1}\beta]\). Given this, and comparing (3) and (4) with (5), yields \( b_i^{PLS} = g_i, \ i = 0, \ldots, k - 1 \).

Since rtCG and PLS solve the same problem and give the same intermediate results, we expect rtCG to have PLS’s robust and stable performance when applied to fMRI data. In practice, rtCG is faster at updating the intermediate results and when we set \( g_0 = b_i^{ICCG} \), it reaches the convergence threshold very quickly.

3. EXPERIMENT RESULTS

We compare the performance of three algorithms: 1) Real-time CG (rtCG); 2) real-time PLS (rtPLS), i.e., PLS applied to \( X^k_i \) and \( y^k_i \); and 3) real-time BPLS (rtBPLS), i.e., the isBPLS algorithm, with sparse setting removed, applied to \( X^k_i \) and \( y^k_i \). We tested rtCG, rtPLS and rtBPLS on three synthetic and two real fMRI data sets. The synthetic data sets consist of \( m \) examples, each of dimension 2000. From the 2000 features, we select 2 disjoint subsets of 100 features. When the label of the example is \( 1 \) (resp. \(-1\)), the first (resp. second) set of features are set to 1 and all other features are zero. In the first synthetic data set \( m = 200 \) and the sequence of labels is a repeating pattern of ten “1’s” and ten “-1’s”. In the second, \( m = 200 \) and the sequence of labels is random. In the third, \( m = 600 \), random labels are used, and every 150 examples, we select two new nonoverlapping sets of features, i.e., after every 150 examples, the model generating the data changes. Finally, \( N(0, 1) \) noise is added to all examples.

The first fMRI data set is a visual perception test. At each time the subject views a flashing checker board. The checker board is presented for 15 secs on the right side of the visual field, then for 15 secs of the left side, and so on. Each TR is 3 secs, so each presentation yields 5 consecutive (labelled) examples. Each example has dimension \( 30 \times 64 \times 64 = 122,880 \). The second fMRI experiment is identical, except that the location of the checker board (right/left) is selected randomly every 6 secs and the meantime between change is 12 secs. The third fMRI data set is one subject’s data from Haxby et al.’s 8-category study [17] in which subjects viewed images from 8 categories. We use 10 runs of one subject’s fMRI data captured while the subject is viewing either face or house images (from [18]). Each example has dimension \( 40 \times 64 \times 64 = 163,840 \).

For rtCG on synthetic data, \( h = 20 \), \( I_{\text{max}} = 10 \) and the convergence tolerance is \( 10^{-2} \). For rtPLS and rtBPLS, \( m = 10 \), \( h = 20 \), and for rtBPLS the ridge parameter is \( 10^{-4} \). All algorithms initiate learning after the first 10 examples.

| Table 1. Results on rtCG, rtPLS and rtBPLS |
|----------------|----------------|----------------|
| Synth 1         | Synth 2         | Synth 3         |
| rtCG             | rtPLS           | rtBPLS          |
| accuracy(%) / average time(ms) | 100 / 2.3       | 100 / 2.3       | 97.97 / 2.4 |
|                  | 100 / 10        | 100 / 10        | 98.81 / 9.9  |
|                  | 99.15 / 7.1     | 99.15 / 7.1     | 98.81 / 9.9  |
| fMRI 1           | fMRI 2          | fMRI 3          |
| rtCG             | rtPLS           | rtBPLS          |
| accuracy(%) / average time(ms) | 92.67 / 414.9   | 91.05 / 408.2   | 90.23 / 548.1 |
|                  | 74 / 1490.3     | 75.26 / 1472.7  | 89.55 / 2129.1 |
|                  | 73.33 / 867.5   | 73.16 / 904.5   | 88.06 / 1354.8 |

The results in Table 1 show that rtCG on high dimensional fMRI data (122,880~163,840 voxels per TR) reaches an accuracy of about 90% using about 0.5 sec of computation per example. In Synth 3 test, rtCG successfully adapts to the change in the model with all of the prediction errors arising from the transition between the two models. Every time the model changes, it takes rtCG less than 10 examples to adapt and reach 100% accuracy. On fMRI data, rtCG outperforms rtPLS and rtBPLS, in both accuracy and computation time.

Using \( b_{t-1}^{rtCG} \) as the initialization for \( g_0 \) at time \( t \) in rtCG helps reduce the magnitude of the initial residual \( \|r_0\| \) at the beginning of step 2 in the rtCG algorithm. In the Synth 1 test, the \( \|r_0\| \) residual is reduced from 80 to 10 within the first 20 examples. The minimum of \( \|r_0\| \) reaches 0.0731 or 0.1% of the \( \|r_0\| \) at time \( t = 1 \). For fMRI 1, the \( \|r_0\| \) residual shown in Fig.1(e) is reduced from around \( 3 \times 10^5 \) to around \( 3 \times 10^4 \). The minimum of \( \|r_0\| \) reaches 20.1888, or 0.0071% of the \( \|r_0\| \) at the beginning. When the algorithm begins learning, the initial residual \( \|r_0\| \) is almost the same as the norm of the example. As the number of examples increases, the regression coefficient is more accurate and the initialization \( g_0 = b_{t-1}^{rtCG} \) leads to less initial residual. The average time and accuracy of rtCG with zero initialization (i.e. \( g_0 = 0 \)) are 442.1 ms and 74% in fMRI 1 test 434.5ms and 74.21% in fMRI 2 test. Comparing these with Table 1, we see that the proposed initialization makes rtCG faster and more accurate.

We note that using \( g_0 = b_{t-1}^{rtCG} \) gives rtCG memory. In the synthetic data tests (Fig.1(a) and (b)) within the same model, rtCG shows a learning curve in the classification results. In the fMRI data tests, the learning curve is less obvious. But in the fMRI test 1 (Fig.1(d)) after the first 40 examples, the prediction accuracy reaches 97.5%. So the memory is helping rtCG learn when the model is constant. However, there is a tradeoff between memory and adaptiveness. In synthetic data test 3, rtPLS and rtBPLS beat rtCG in accuracy since it takes rtCG with \( g_0 = b_{t-1}^{rtCG} \) longer to forget the past model. Fig.1(c) shows the initial residual plot of synthetic data test 3 with the changing model. As we can see, when the model changes (at example 151, example 301, example 451), the initial residuals increase. By detecting the large increase in
initial residuals, it may be possible to locate the time when
the model changes and reset rtCG.

![Graphs](image)

(a) Prediction (Synthetic 2)  (b) Prediction (Synthetic 3)

(c) $||r_0||$ Residual (Synthetic 3)  (d) Prediction (fMRI 1)

(e) $||r_0||$ Residual (fMRI 1)  (f) Prediction (fMRI 2)

Fig. 1. Prediction results and $||r_0||$ residuals

4. CONCLUSION

We have proposed an online conjugate gradient algorithm as an efficient and adaptive solution for an online rtfMRI sys-

![Diagram](image)

tem. Our experiments show that rtCG can process high di-

![Diagram](image)

mensional fMRI data using about 0.5 sec and reach a predic-
tion accuracy around 90%. We have also provided a theore-
tical connection between rtCG and PLS when applied to the

![Diagram](image)

quadratic program of interest. More complex experiments are

![Diagram](image)

planned to test rtCG and evaluate its performance. rtfMRI is

still a new technique and more work is needed to explore its

![Diagram](image)

potential applications. For example, a successful online rtfMRI system will raise new questions, such as how
to best take advantage of the real-time classification results
to aid understanding of cognitive function.

5. REFERENCES


568