33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

An Information-Theoretic Approach to Motor Action Decoding with a Reconfigurable Parallel Architecture

Stefan Craciun, *Student Member, IEEE*, Austin J. Brockmeier, *Student Member, IEEE*,
Alan D. George, *Senior Member, IEEE*, Herman Lam, *Member, IEEE*, and José C. Príncipe, *Fellow, IEEE*

***Abstract*— Methods for decoding movements from neural spike counts using adaptive filters often rely on minimizing the mean-squared error. However, for non-Gaussian distribution of errors, this approach is not optimal for performance. Therefore, rather than using probabilistic modeling, we propose an alternate non-parametric approach. In order to extract more structure from the input signal (neuronal spike counts) we propose using minimum error entropy (MEE), an information-theoretic approach that minimizes the error entropy as part of an iterative cost function. However, the disadvantage of using MEE as the cost function for adaptive filters is the increase in computational complexity. In this paper we present a comparison between the decoding performance of the analytic Wiener filter and a linear filter trained with MEE, which is then mapped to a parallel architecture in reconfigurable hardware tailored to the computational needs of the MEE filter. We observe considerable speedup from the hardware design. The adaptation of filter weights for the multiple-input, multiple-output linear filters, necessary in motor decoding, is a highly parallelizable algorithm. It can be decomposed into many independent computational blocks with a parallel architecture readily mapped to a field-programmable gate array (FPGA) and scales to large numbers of neurons. By pipelining and parallelizing independent computations in the algorithm, the proposed parallel architecture has sublinear increases in execution time with respect to both window size and filter order.**

I. Introduction

In the area of brain-machine interfaces, adaptive filters (AFs) are well suited for neural decoding. Generally, because the statistics of the input signal are unknown (as is often the case), we can use AFs to estimate these required signal statistics through an iterative adaptation process. AFs adjust their internal parameters or weights dynamically to match the changes in the input, usually in a robust way in order to meet a specific performance criterion. This performance criterion, also known as the cost function, will set the rules of adaptation at each iteration. In this paper we compare the performance of two cost functions: the Wiener filter that aims at minimizing the mean squared error (MSE) and an AF that adapts its weights by minimizing the entropy of the error (MEE). The gain in performance of the MEE cost function comes at the cost of increased computational complexity.

The tradeoff between performance and complexity is an important factor that has impeded MEE-based algorithms from being implemented in real-time. We address this problem by designing a parallel architecture for a reconfigurable hardware solution. We show that the learning time for decoding motor action can be greatly reduced, allowing us to take advantage of the superior performance of the MEE cost function without incurring the time penalties encountered in software. By parallelizing the algorithm, we can more efficiently search the solution space for the global minima, and adapt all the weights of our system in parallel. With each iteration, the weights move in the direction of the estimated gradient in small increments, reaching a stable solution in a finite number of steps. Our experiments prove that no precision is lost, as both hardware and software adapt in the same number of iterations.

The least mean squares (LMS) filter, a stochastic gradient descent AF, and the Wiener filter, which is a one step analytical solution, base their cost function criterion on minimizing the mean squared error (MSE) [1]. Both methods are computationally efficient, but MSE-based cost functions are only an optimum solution when applied to Gaussian signals with linear filters [1]. The central limit theorem states that a summation of sufficiently large numbers of random variables is Gaussian, but if the input consists of a small number of counts, as in the case of binned neural action potentials, the distribution of the errors is not Gaussian. Thus, we propose using a cost function inspired by Information Theory and based on Rényi's quadratic entropy measure [2]. This cost function is optimal for nonlinear signal processing and will achieve better performance when compared to the Wiener solution, as we will show in the Methods and Results section of this paper. However, the estimation of the error density requires a kernel evaluation between all sample pairs, resulting in a substantial increase in computational complexity. Consequently, the MEE cost function has not been suited to real-time implementation. Reconfigurable computing (RC) can provide an efficient solution to this computational problem by mapping the algorithm onto hardware, taking advantage of its unique dataflow and inherent data dependencies. Furthermore, by creating a custom parallel architecture fine-tuned to the computational needs of the MEE cost function, we can exploit the parallelism inherent in this algorithm with a hardware design that achieves considerable speedup.

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422 and DARPA Contract N66001-10-C-2008.

S. Craciun*†, A. J. Brockmeier*, A. D. George†, H. Lam†, and J. C. Príncipe* are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA. (email: craciuns@ufl.edu, ajbrockmeier@cnel.ufl.edu, george@chrec.org, hlam@chrec.org, principe@cnel.ufl.edu)

*Computational Neuro-Engineering Laboratory (CNEL)

†NSF Center for High-Performance Reconfigurable Computing (CHREC)

II. Methods and Results

A. Adaptive Filter

The structure of adaptive filters can be divided into two fundamental blocks as shown in Fig. 1. The first is the adaptive FIR, which contains the input delay line along with the filter weights. The second block contains the cost function and the learning algorithm. The new weights are computed based upon the current and past input and error values. A batch of input samples will generate a sequence of errors, which are stored in a delay line. The number of past errors used to update the weights within every iteration is defined as the *window size*.

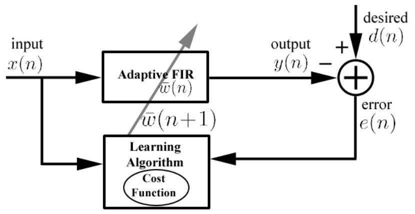

Fig. 1. General-adaptive filter structure

The output of a general finite-impulse response (FIR) filter is defined as the dot product of input $\bar{x}$ and the L weights $\bar{w}(n)$. For neural decoding we consider the case of M inputs denoted $\{\bar{x}^k\}_{k=1}^M$ with weights $\{\bar{w}^k\}_{k=1}^M$.

$$y(n) = \sum_{k=1}^{M}\sum_{i=1}^{L} x^k(n-i+1)\cdot w_i^k(n). \tag{1}$$

The error at time instant n is $e(n)$ and is simply the difference between the filter output and the desired value $e(n) = d(n) - y(n)$. The MEE cost function provides an update equation for every weight of the FIR based on the current value of the weights $\bar{w}(n)$, a step size μ, and the gradient of the information potential ∇V.

$$\bar{w}(n+1) = \bar{w}(n) + \mu\nabla V \tag{2}$$

The information potential (IP) is defined [2] as the argument of the logarithm in Rényi's quadratic entropy of the distribution of errors. Let the probability density of errors be $p(e)$ then the quadratic entropy of an error distribution $\mathcal{E}$ is $H_2(\mathcal{E}) = -\log\int p^2(e)\mathrm{d}e$, and a nonparametric estimator of the information potential (IP) [2] using the Gaussian kernel $\kappa_\sigma(.)$ over the last N errors (using subscripts for compactness) is

$$V = \frac{1}{n^2}\sum_{i=n-N+1}^{n}\sum_{j=n-N+1}^{n}\kappa_\sigma(e_i - e_j) \tag{3}$$

where $\kappa_\sigma(e_i - e_j) = \left(\sqrt{2\pi}\sigma\right)^{-1}\exp\left((e_i-e_j)^2/(2\sigma^2)\right)$ is the kernel function for density estimation. By taking the derivative of the IP with respect to weight w_l^k, we can compute the gradient at step n

$$\nabla V_l^k \propto \sum_{i=n-N+1}^{n}\sum_{j=n-N+1}^{n}\kappa_\sigma(e_i-e_j)\left(e_i-e_j\right)\left(x_{i-l+1}^k - x_{j-l+1}^k\right) \tag{4}$$

where a constant multiplicative factor is absorbed in μ in (2).

The three most important parameters influencing the computational complexity of the overall filter adaptation are the FIR order and the window size used to estimate IP. The computational complexity of the algorithm is $\mathrm{O}(n^2)$ with respect to window size N but only linear $\mathrm{O}(n)$ with respect to filter order and number of inputs $L\cdot M$. By varying the size of the filter order delay line, we will influence the performance of the prediction. A large window size N is needed to correctly estimate the IP gradient (4), but the $\mathrm{O}(n^2)$ complexity weighs heavily on the sequential execution time of the algorithm.

B. Motor Decoding from Neural Subsets

We use linear models to learn motor action decoding from the firing rates of small subsets of neurons (8 neurons). The motivation is to identify subsets of neurons whose activity provides good predictors of the movement, as neurons widely vary in their prediction performance for different dimensions of the movement. Thus, we choose random subsets of neurons and train linear models for decoding movement from each subset. The neurons that appear in multiple subsets, or the best performing subsets, could be selected for use in a larger more complicated model. This process is in essence a brute force feature selector, normally infeasible in standard hardware, but can be optimized in reconfigurable computing.

We propose using MEE for the adaptive filter methodology because for binned spike data, the distribution of values is far from Gaussian, thus a small combination of these values will also be non-Gaussian (see Fig. 2 for the distribution of errors from a Wiener solution with 8 neurons).

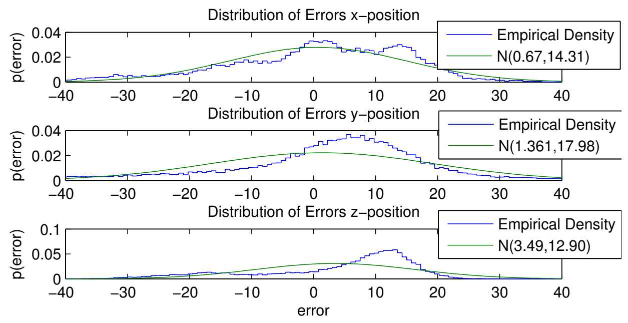

Fig. 2. Error distributions for Wiener solution compared to Gaussians.

MEE is contrasted to the Wiener filter that assumes uses only second-order information about the error. The higher-order information that MEE can extract is useful when the error distribution is far from Gaussian.

C. Dataset Description and Results

The neural data we use was collected in Dr. Nicolelis's primate laboratory at Duke University. Specifics can be found in [3]. The data is recorded from an owl monkey's cortex while the animal was performing a food-reaching task with a single arm. Multiple micro-wire arrays record from 104 neural cells in multiple cortical areas: posterior parietal cortex; left and right primary motor cortex; and dorsal premotor cortex. Synchronous recordings provide the reaching hand's position in three dimensions. The spikes are binned at 100 ms and the three-dimensions (x, y, and z) of hand position are also at 10 Hz. In this dataset, there are over 38 minutes of data during which the animal would reach and eat food sitting on two different trays, while in-between reaches, the animal returns its hand to a resting location. In our experiments we used 200 seconds of training data and 35 minutes of testing data.

The results use 160 random subsets of 8 neurons that were selected and tested with the condition that all neurons have at least a 1Hz firing rate. Both models used 3 taps (300ms) for each neuron as input, and a decoder for each dimension of hand position is trained independently. The Wiener solution was regularized by adding a scaled identity vector to the autocorrelation matrix $\bar{w} = (\mathbf{R}+10^{-4}\mathbf{I})^{-1}\bar{P}$, where $\mathbf{R}$ is the auto-correlation matrix and $\bar{P}$ is the cross-correlation vector [1]. For the MEE filter we used a kernel size of $\sigma = 49.5$ (the largest 10-90 percentile range) , a window length $N = 75$ errors (chosen to cover the time between movements), and a step size $\mu = 0.05$ (for stability). The performance is gauged in terms of cross-correlation between the true movement and the predicted,

$$CC(x, y) = \frac{\mathrm{E}[xy]}{\sqrt{\mathrm{E}[x^2]\mathrm{E}[y^2]}}. \tag{5}$$

Fig. 3 compares the true movement to the Wiener and MEE prediction using a single random subset of 8 neurons over a 1 minute window. Fig. 4 compares the cross-correlation for the Wiener and the MEE solution across all subsets and output dimensions. For subsets with poor performance there is no difference in the MEE versus Wiener performance; however, for the best performing subsets the MEE solution does better in nearly all of the subsets. It should be noted that neither filter does well in predicting the x direction ($CC < .5$) in this particular dataset as noted for a wide range of filters in Kim et al. [4].

III. Parallel Architecture

In this section we describe how the decoding algorithm's MEE cost function is mapped to a parallel architecture in reconfigurable hardware. Fig. 5 shows the overall architecture of the adaptive filter, which consists of M FIR filters, the error pairwise distance block, the Gaussian kernel block, M input pairwise distance blocks, and parallel pipelined accumulators. The input from each of the M neurons enters a FIR of length L such that, at every clock cycle a new sample enters the delay line and the oldest sample is forgotten. In addition, the last N input values for each neuron and

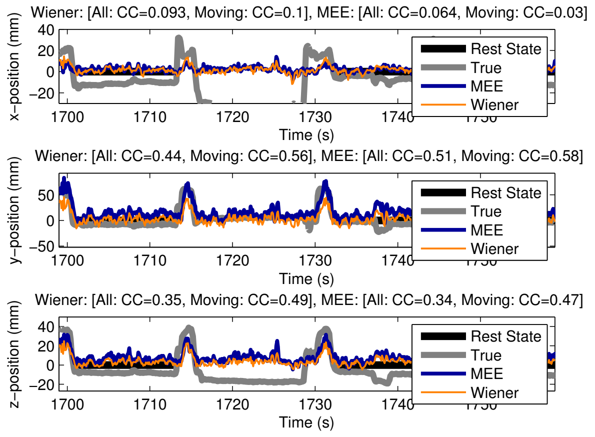

Fig. 3. Snapshot of 3 reaches with a comparison of the true movement, Wiener prediction, and MEE prediction.

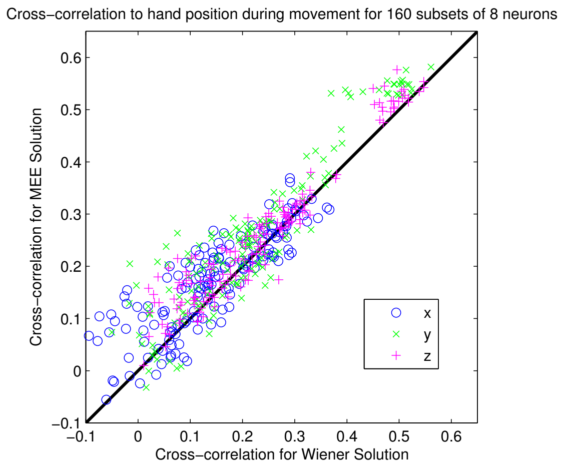

Fig. 4. Cross-correlation to hand position for 160 subsets of 8 neurons.

the N errors are stored to estimate the IP gradient (4) and thus the new weights. The parallel architecture and hardware design balances precision and resource utilization. The FIRs are constructed from 32-bit fixed-point registers, adders, and multipliers. For this design, the pipeline can process a new sample from each of the M neurons every clock cycle without stalling.

Computing all pairwise distances (4) takes a considerable amount of time in software because each error requires $N-1$ pairwise error distance evaluations. In a pipelined and parallel design, all these independent computations can be parallelized. Fig. 6(A) shows the pipelined design of the pairwise error distance block, which allows for the computation of pairwise distances between the current error and all past errors in just one clock cycle. Similarly, each input pairwise distance employs the same pipelined design. By computing pairwise distances in parallel, this architecture only utilizes 1 clock cycle to compute $N-1$ subtractions for errors and $M \cdot (N-1)$ subtractions for inputs.

The second term in (4) is the Gaussian kernel evaluation

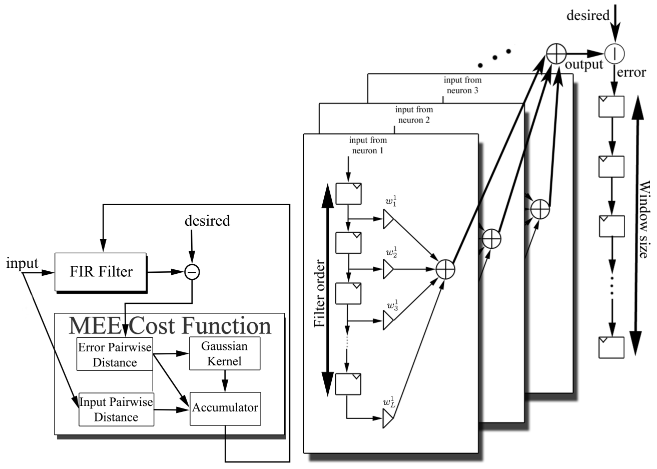

Fig. 5. (Left) MEE cost-function subcomponents. (Right) Dataflow.

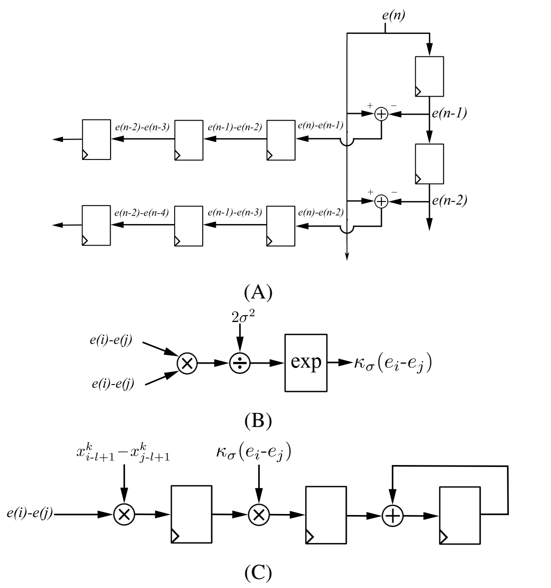

Fig. 6. (A) Pipelined pairwise distance computation. (B) Gaussian kernel block. (C) Pipelined accumulator.

$\kappa_\sigma(e_i - e_j)$. As the argument of the exponential function has a wide dynamic range, we concluded that pre-computed lookup tables estimating the exponential function would require considerable FPGA resources, thus preventing us from fully exploiting the wide parallelism of the algorithm. We used the first 6 terms of the Taylor series to compute the exponential function as needed in the Gaussian kernel, as shown in Fig. 6(B). As soon as the error pairwise distances become available from the pairwise distance block, their Gaussians can be computed in parallel with the latency of one kernel computation. In software all Gaussian kernels are computed sequentially. Finally, as suggested by the double summations in (4), the product between the error distances, Gaussian kernels, and input distances needs to be summed. Fig. 6(C) shows the three-step accumulator required to sum all pairwise interaction. In total, there are N accumulators to parallelize the double summation in (4).

We tested the performance of our parallel architecture at the NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida [5]. Our design for an information-theoretic adaptive filter was tested on an Altera Stratix-III E260 FPGA featuring 256K logic elements with 4.25GB of dedicated memory in three parallel banks and operating at 150 MHz. The software baseline used for comparison is written in C with optimization O2 and executed on a 2.4 GHz processor with 8GB of RAM. For less than 20 neurons the entire design fits on one FPGA. A weight update for 8 neurons with a window of 1000 samples takes 40 μs in software compared to 6 ms in hardware, but the hardware achieves sublinear increases in execution time for increasing window sizes and number of neurons as shown in Table I.

TABLE I

EXECUTION TIME SCALING, RELATIVE TO $N = 100, M = 8$ FOR HARDWARE(SOFTWARE)

N	M # of neurons		
window size	8	20	40
100	1(1)	2.1(2.1)	3.7(4.1)
10000	1.2(73)	1.7(195)	3.8(390)

IV. CONCLUSION

This paper proposes an information-theoretic approach to identify subsets of neurons that can be used to decode motor intent. We compare an error-entropy minimization cost function to the conventional mean-squared-error cost function in the classic Wiener filter. When using lower numbers of binned spike counts from neurons to decode motor intent, the training errors are non-Gaussian, suggesting that a typical MSE approach is not optimal. In contrast, an MEE type of cost function is able to extract higher-order statistics, as opposed to only the power of the error, and can optimally adapt to non-Gaussian errors. The penalty we pay for this cost function is higher computational complexity that translates to a substantial increase in training time. The solution to this problem is a pipelined and parallel hardware architecture that can exploit algorithm parallelism (both wide and deep), essentially decreasing training time, and converging to the optimal weights much faster. Execution time experiments demonstrate the ability of the hardware to efficiently scale to larger window sizes. This allows MEE-based algorithms to be used in real-time applications.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*. Prentice Hall, 2002.

[2] D. Erdogmus and J. Principe, "Generalized information potential criterion for adaptive system training," *Neural Networks, IEEE Transactions on*, vol. 13, no. 5, pp. 1035–1044, 2002.

[3] J. Wessberg, C. Stambaugh, J. Kralik, P. Beck, M. Laubach, J. Chapin, J. Kim, S. Biggs, M. Srinivasan, and M. Nicolelis, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, no. 6810, pp. 361–365, 2000.

[4] S. Kim, J. Sanchez, Y. Rao, D. Erdogmus, J. Carmena, M. Lebedev, M. Nicolelis, and J. Principe, "A comparison of optimal MIMO linear and nonlinear models for brain–machine interfaces," *Journal of Neural Engineering*, vol. 3, p. 145, 2006.

[5] "Novo-G Overview." [Online]. Available: http://www.chrec.org/~george/Novo-G.pdf