Finding Neural Codes Using Random Projections

Brendan Mumey
Department of Computer Science
Montana State University
Bozeman, MT 59717-3880, USA

Aditi Sarkar
Department of Cell Biology and Neuroscience
Montana State University
Bozeman, MT 59717-3148, USA

Tomáš Gedeon
Department of Mathematical Sciences
Montana State University
Bozeman, MT 59717-0240, USA

Alex Dimitrov
Center for Computational Biology
Montana State University
Bozeman, MT 59717-3148, USA

John Miller
Department of Cell Biology and Neuroscience
Montana State University
Bozeman, MT 59717-3148, USA

February 23, 2003

Abstract

A powerful approach to studying how information is transmitted in basic neural systems is based on finding stimulus-response classes that optimize the mutual information shared between the classes [2, 10]. The problem can be formally described in terms of finding a optimal quantization (X_N, Y_M) of a large discrete joint (X, Y) distribution and various algorithms have been developed for this purpose. Recently it has been proved that finding the optimal such quantization is NP-complete [7], indicating that exact solutions may be computationally infeasible to find in some circumstances. To this end we have developed a new *randomized* algorithm to solve the joint quantization problem. Under assumptions about the underlying (X, Y) distribution, we prove that this algorithm converges to the "true" optimal quantization with high probability that can be increased by repeating the number of random trials used.

1 Introduction

A central question in neuroscience is how is information represented and transmitted in simple neural systems. Our work has focused on studying a sensory system in crickets, although the techniques described are generally applicable to any situation where recordings of neural activity can be correlated with an input stimulus. This work primarily addresses a new method for processing such large neural activity trace data sets (consisting of a time series of input stimulus and output spike events across one or more axons). Existing techniques work

well for the case of a single dimensional spike train recording [3], but such methods may not scale well to high dimensional recordings made on multiple neural signals. We present a new approach to overcome the "curse of dimensionality" to some extent by using a new method of *random projections* to find subtle relations between input stimuli and output pattens potentially encoded over several neural channels. The method of random projections has recently been successfully applied to the problem of finding subtle motifs in protein sequence families [1].

We formulate the neural coding problem in terms of quantizing a joint (X, Y) space where X represents the input stimuli set and Y is the set of possible neural activity patterns. Both of these spaces are high dimensional, especially so if multiple signal channels are recorded simultaneously with the potential for coding to be multiplexed across the channels. Since noise is omnipresent in neural processing, sensory systems must be robust. As such, they must represent similar stimuli in a similar way and then react to similar internal representations of the outside stimuli in a consistent way. This leads to a conclusion that individual input and output patterns are not important for understanding neural function, but rather classes of input stimuli and classes of output patterns and their correspondence is the key. Therefore we are led to a problem of optimal assignment of input and output patterns to classes, where the optimality is judged on how much original mutual information is still present between the collection of classes.

2 Background

Given two discrete random variables X and Y the mutual information $I(X, Y)$ is defined by

$$I(X, Y) = \sum_{x,y} p(x, y) \lg \frac{p(x, y)}{p(x)p(y)}.$$

The mutual information is always nonnegative and achieves its minimal value 0 if the random variables X and Y are independent. The value $I(X, Y)$ tells us, roughly, how much we can learn X by observing Y, and vice versa. In joint quantization [5] we fix sizes M and N of the reproduction spaces $X_M = \{x_1, \ldots, x_m\}$ for X and $Y_N = \{y_1, \ldots, y_n\}$ for Y and define joint quantization using two sets of quantizers, $q(x_m|x)$ and $q(y_n|y)$. Quantizers are conditional probabilities and so $q(x_m|x) \in \Delta_M$, and $q(y_n|y) \in \Delta_N$, where $\Delta_K := \{y \in R^K \mid \sum_i y_i = 1, y_i \geq 0\}$ for $K = M, N$. The function to be minimized is

$$I(X, Y) - I(X_M, Y_N).$$

This is equivalent to the optimization problem

$$\max_{q(x_m|x)\in\Delta_M, q(y_n|y)\in\Delta_N} I(X_M, Y_N), \tag{1}$$

where

$$I(X_M, Y_N) \quad = \quad \sum_{x_m, y_n} p(x_m, y_n) \lg \frac{p(x_m, y_n)}{p(x_m)p(y_n)}$$

A variety of methods have been developed to solve this optimization problem [4, 6, 8, 9]. These methods have been successfully employed for a variety of neural coding problems but they have difficulty scaling to very high dimensional spaces as discussed in the next section.

3 Quantizing Using Random Projections

The high dimensionality of X and Y make it difficult to use traditional methods for finding an optimal quantization of the joint space. With high probability, identical points in either space are never repeated, even in large data sets. For example, if we view the output space Y as a collection of binary vectors (where a 1 indicates a spike) of predetermined length n, with one vector for each of k channels, then the size of Y is 2^{kn}. Hence, it is very unlikely that points in Y are repeated. Unless a metric of some sort is imposed beforehand on Y it is very difficult to say which points might be good candidates for grouping together in the quantization. Another problem with a preimposed metric is that we don't know what features of X and Y are most the important with respect to finding the optimal quantization.

At a high level, the method of random projections works by repeating the following procedure: construct random projections of both spaces $f : X \to X'$ and $g : Y \to Y'$ where X' and Y' are much smaller spaces. The functions f and g are referred to as *random hash functions* and are chosen at random from hash function families F and G respectively (we discuss some initial choices of hash function families in the experimental results section). Compute the mutual information I of the data points as projected into (X', Y'); this quantity is a measure of how interesting the random projection is. By the Data Processing Theorem, $I(X'; Y') \leq I(X; Y)$. Suppose that $I = I(f, g)$ is large (perhaps close to $I(X, Y)$). This means that the random hashing preserved most of the mutual information in the original joint distribution. Thus, if data points (x_1, y_1) and (x_2, y_2) hash to the same location, i.e. $f(x_1) = f(x_2)$ and $g(y_1) = g(y_2)$, this is evidence that they should be quantized together and belong to the same joint class in (X_M, Y_N). Of course this could have just happened by chance, so we need to repeat this process with many random projections. We can summarize the results of all the projections efficiently in a weighted *joint similarity graph* constructed on the joint data points. A vertex v_i represents the joint data point (x_i, y_i). If data points (x_i, y_i) and (x_j, y_j) *collide* (meaning that they map to the same point in (X', Y') on applying hash functions f and g), and $I(f, g)$ is greater than a prespecified cut-off I_T, then we add 1 to the weight of the edge between v_i and v_j (inserting the edge first if didn't previously exist). Currently we chose I_T so that approximately 10% of all hash functions are kept, although this parameter can obviously be tuned. If a sufficient number of hash functions are employed, the resulting joint similarity graph can be used to find the natural clusters in joint clusters in the (X, Y) distribution and in so doing construct effective quantizers X_M and Y_N. If an edge weight $w(v_i, v_j)$ is relatively high, this provides evidence that the underlying data points (x_i, y_i) and (x_j, y_j) should belong to the same class in the joint quantization. We can computationally determine the quantized classes by partitioning the vertices in the graph into a set of clusters. These clusters will represent the joint classes. We expect the intra-cluster edge weights to be much higher on average than the the inter-cluster edge weights and use this fact to discover the clusters. A number of potential graph algorithms could be employed at this point to find the clusters (multi-way minimum cuts, approximate cliques); we have chosen a simple greedy strategy based on picking vertices that appear to cluster centers and then assigning the remaining vertices to their nearest center.

Further details on the method will appear in the full version of the paper.

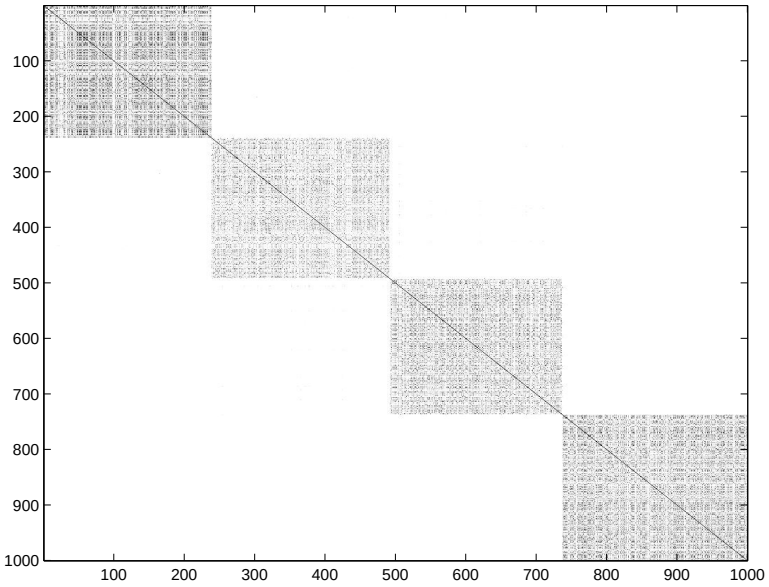

Figure 1: **The joint similarity matrix for the 4-blob data set.** This matrix shows the similarity between each pair of data points as determined by the edge weights in the joint similarity graph. Similarity strength is indicated using a grey scale (dark = high similarity). The data points have been permuted into the four clusters found by the algorithm. The clusters corresponding exactly to the underlying "true" clusters; no false predictions were made.

4 Experimental Results

In this section we describe our initial results with the random projection method for finding neural codewords. We have tested a simple Matlab implementation of the method on two data sets. The first data set was very simple data set, called *4-blob*, which as the name suggests, is a joint distribution formed by four disjoint Gaussian distributions in R^2 space ($X = Y = R$). One thousand joint data points were sampled at random from this distribution and used as input for the random projection algorithm. Random hash functions f were constructed by picking a fixed number k (we chose $k = 10$) of random "centers" then hashing each point x to its nearest center $\{1, \ldots, k\}$. G was constructed in a similar fashion. The results are shown in Figure 1.

The second data set we are currently looking at is real cricket neural trace data collected by the Center of Computational Biology and Montana State Univesity. For this experiment, the set of stimulus waveforms consisted of a set of simple uni-directional air current "puffs" of identical shape and duration, presented sequentially from a variety of directions around the animal's body. Stimulus angle can be represented as a one-dimensional variable. The neuron's response in this simple case was represented by the number of elicited spikes in a 50ms window. The particular cell studied here shows pronounced directional selectivity, i.e., the number of spikes it fires depends on the direction from which the air puff originates. We have found fairly strong evidence for four joint clusters in the (X, Y) space for this data. The hash function family F chosen was similar to the 4-blob case. This is because the input stimulus waveform can modeled as being sampled from a fixed size collection of template waveforms with some Gaussian noise. Each random hash function f

was constructed by chosing k stimulus waveforms to be "centers" and then mapping each remaining waveform x to its nearest (L_2 metric) center. The G random hash family was somewhat different. In this case the output space Y consists of binary vectors representing the spike occurence pattern y recorded over a fixed time after the occurence of each stimulus waveform x. We chose a fairly simple random hash function based on counting the number of spikes in y and mapping this to a smaller set of integers $Y' = \{1, \ldots, p\}$. The mapping was constructed by creating a random p-partition of the set $\{1, \ldots, n\}$, where n was the maximum observed spike count in Y. The spike count of each y was determined and y was mapped to the partition number that its spike count fell into.

Details of this and other experiments with real neural recoding data sets will appear in the full version of the paper.

5 Conclusions and Future Work

Neural code determination is an important problem in neurobiology. Analysis methods must be able to scale with larger data sets, both in the time dimension and the spatial dimension of how many neural signals are simultaneously recorded. The random projection method is a promising new technique for studying neural coding through optimal mutual information joint quantization.

Our initial experimental results are promising. On the 4-blob data set, the algorithm successfully categorized each data point into the correct "blob" class. The real data results while preliminary, are also quite encouraging. Further validation studies are ongoing and will be reported in the full version of the paper.

We are actively investigating both theoretical and practical improvements to the method. In particular the family of random hash functions can be improved and refined. The hash functions we used for the spike vectors above are quite basic (capturing only rate encodings); one can imagine more complicated projections based on the number of doublets, triplets, etc. or other patterns spanning multiple channels. We are in the process of testing a variety of hash functions on multiple channel data sets. At the moment, we propose weighting edges in the joint similarity graph by counting the number of "interesting" hash functions for which their endpoints collide. It is possible that other edge weighting functions may be more informative (e.g. just adding $I(f, g)$ to the weight of the edge instead of using a cut-off threshold). Another interesting direction is to use a traditional quantization algorithm on the (X', Y') space after each random hash application. This would have the effect of mapping the points to an even smaller joint space (X'', Y'') (points of which represent the joint classes found) and perhaps add more useful edges to the joint similarity graph at the cost of increased computational effort per random step.

References

[1] Jeremy Buhler and Martin Tompa. Finding motifs using random projections. In *Proceedings of the fifth annual international conference on Computational biology*, pages 69–76. ACM Press, 2001.

[2] Alexander G. Dimitrov and John P. Miller. Neural coding and decoding: communication channels and quantization. *Network: Computation in Neural Systems*, 12(4):441–472, 2001.

[3] Alexander G. Dimitrov, John P. Miller, Zane Aldworth, and Tomáš Gedeon. Non-uniform quantization of neural spike sequences through an information distortion measure. In James Bower, editor, *Computational Neuroscience: Trends in Research 2001*. Elsevier, 2001.

[4] Alexander G Dimitrov, John P Miller, Tomáš Gedeon, Zane Aldworth, and Albert E Parker. Analysis of neural coding using quantization with an information-based distortion measure. *Network: Computation in Neural Systems*, 2003. *(at press)*.

[5] Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *Proc. Seventeenth Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[6] Tomáš Gedeon, Albert E. Parker, and Alexander G. Dimitrov. Information distortion and neural coding. *Can. Math. Q.*, 2002. *(at press)*.

[7] Brendan Mumey and Tomáš Gedeon. Optimal mutual information quantization is NP-complete. Initial version to appear at the Neural Information Coding (NIC) workshop, Snowbird UT, 2003.

[8] Albert E. Parker, Tomáš Gedeon, Alexander G. Dimitrov, and Brian Roosien. Annealing and the rate distortion problem. In S. Becker, S. Thrun, and K Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. 2003. *(to appear)*.

[9] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 617–623. MIT Press, 2000.

[10] Naftali Tishby, Fernando Pereira, and William Bialek. The information bottleneck method. In *Proceedings of The 37th annual Allerton conference on communication, control and computing*. University of Illinios, 1999.