Finding Neural Codes Using Random Projections

Brendan Mumey [a,*] Aditi Sarkar [b] Tomáš Gedeon [c] Alex Dimitrov [b]
John Miller [b]

[a] *Department of Computer Science*
Montana State University
Bozeman, MT 59717-3880, USA

[b] *Department of Cell Biology and Neuroscience*
Montana State University
Bozeman, MT 59717-3148, USA

[c] *Department of Mathematical Sciences*
Montana State University
Bozeman, MT 59717-0240, USA

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 [2,6]. The problem can be formally described in terms of finding a optimal quantization (A, B) 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 [3], indicating that exact solutions may be computationally infeasible to find in some circumstances. 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 performing additional random trials.

Key words: neural coding, random projections, information quantization

1 Introduction

How is information represented and transmitted in simple neural systems? Our work on this central question in neuroscience 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 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" using a method of *random projections* to find subtle relations between input stimuli and output pattens potentially encoded over several neural channels.

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

*

Email address: mumey@coe.montana.edu (Brendan Mumey).

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

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

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. For the application to neural coding both X and Y are very high-dimensional spaces, so we are motivated to find a joint quantization of the underlying distribution into reproduction spaces $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ and a joint quantization using two sets of quantizers, $q(a_i|x)$ and $q(b_j|y)$. Quantizers are conditional probabilities and so $q(a_i|x) \in \Delta_m$, and $q(b_j|y) \in \Delta_n$, where $\Delta_k = \{z \in R^k | \sum_i z_i = 1, z_i \geq 0\}$. We would like to minimize that amount of distortion introduced by the quantizer (for fixed m and n). Distortion is defined as

$$D(A,B) = I(X,Y) - I(A,B),$$

where

$$\begin{aligned} I(A,B) &= \sum_{a_i,b_j} \Pr(a_i,b_j) \lg \frac{\Pr(a_i,b_j)}{\Pr(a_i)\Pr(b_j)} \\ \Pr(a_i,b_j) &= \sum_{x,y} q(a_i|x)q(b_j|y)\Pr(x,y) \\ \Pr(a_i) &= \sum_j \Pr(a_i,b_j) \\ \Pr(b_j) &= \sum_i \Pr(a_i,b_j). \end{aligned}$$

This is equivalent to the optimization problem

$$\max_{q(a_i|x)\in\Delta_m, q(b_j|y)\in\Delta_n} I(A,B). \tag{1}$$

A variety of methods have been developed to solve this optimization problem but all 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 in stochastically generated data sets. 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.

One way to circumvent some of these difficulties is offered by the method of random projections. 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 [1], $I(X^*, Y^*) \leq I(X, Y)$. We say two data points (x_i, y_i) and (x_j, y_j) *collide* if they hash to the same location, i.e. $f(x_i) = f(x_j)$ and $g(y_i) = g(y_j)$. This is some evidence that this pair of data points should be quantized together and belong to the same joint class in (A, B). 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 i-th joint data point (x_i, y_i). If data points (x_i, y_i) and (x_j, y_j) collide 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 the best 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. 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. We currently use an existing clustering software package, CLICK [4], to find clusters in the joint similarity graph.

3.1 Choosing random projections

The hash function families F and G are constructed differently as the underlying spaces X and Y are different. In the case of cricket sensory data sets, the input space X consists of waveform of length δ milliseconds (e.g. $\delta = 65$). The output space Y represents the spike train recording (typically one or more binary vectors depending on the number of channels recorded).

For hashing the input space, we construct $f \in F$ as follows: First, chose k stimulus waveforms to be "centers". Then, map each waveform x to its nearest (L_2 metric) center $\{1, \ldots, k\}$. The output space Y consists of binary vectors representing the spike occurrence pattern y recorded over a fixed time after the occurrence of each stimulus waveform x. A simple random hash function family G is based on counting the number of spikes in y and mapping this to a smaller set of integers $Y^* = \{1, \ldots, p\}$. The mapping is 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 is determined and y is mapped to the partition number that its spike count falls into. We view F and G as test cases and are further refining both families.

We present preliminary theoretical results on the number of random trails necessary to successfully recover the correct joint quantization. We make the assumption that the underlying joint space (X, Y) is clustered into k clusters $C_1, \ldots, C_k$ and that each cluster C_i is sampled with probability $p_i \geq 0$, $\sum_i p_i \leq 1$. The random projection method is sensitive to the discrimination ability of the hash functions used. We have analyzed the method under the assumption of a *good projection family*. Let $(x, y) \in C_i$ and $(x', y') \in C_j$. We say (F, G) is good if there exists constants $1 \geq q > r > 0$ such that for all $1 \leq i, j \leq k$:

$$\begin{array}{l} i = j \Rightarrow \Pr[(x,y) \text{ and } (x',y') \text{ collide}] = q \\ i \neq j \Rightarrow \Pr[(x,y) \text{ and } (x',y') \text{ collide}] = r \end{array}$$

We assume collision probabilities are independent between all pairs of non-identical points in the joint space.

Lemma 1 *If $q - r = \Omega(1/n)$, we can recover the true joint clusters with high probability with $O(n)$ random projections.*

PROOF. We observe that intracluster and intercluster edge counts in the joint similarity graph are binomially distributed with parameters q and r respectively. Suppose m random hashes are performed on the joint space. We use the following rule to decide if an edge in the joint similarity graph is an intracluster edge: if the edge count is greater than or equal to a threshold τ, we keep the edge, otherwise we discard it. Let Q be the probability that a true intracluster edge is kept and let R be the probability that a false intercluster edge is kept. It can be verified that $Q - R$ is maximized when

$$\tau = \left[1 + \frac{\log q - \log r}{\log(1-r) - \log(1-q)}\right]^{-1} \cdot m$$

and that if $m = O(n)$ then $Q - R = \Omega(1)$. At this point, recent random graph clustering results [5] can be applied to find the clusters with high probability. We have omitted full details due to space constraints.

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 several data sets. We report on a test data set of cricket sensory neural trace data collected by the Center of Computational Biology and Montana State University. In the experiment, a “white noise” air current stimulus was applied to the cricket. A single neural channel was recorded. Each time an *onset spike* (no activity for several hundred milliseconds previously) occured an input-output pair was generated as follows: The input x was formed by taking a (-30ms, +25ms) window of the input stimulus wave form, where 0 indicates the time of the onset spike. The corresponding output y recorded the spike occurences in the (0,+25ms) window (encoded as a binary vector). We have found strong evidence for four joint clusters in the (X, Y) space for this data set, as shown in Figure 1.

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

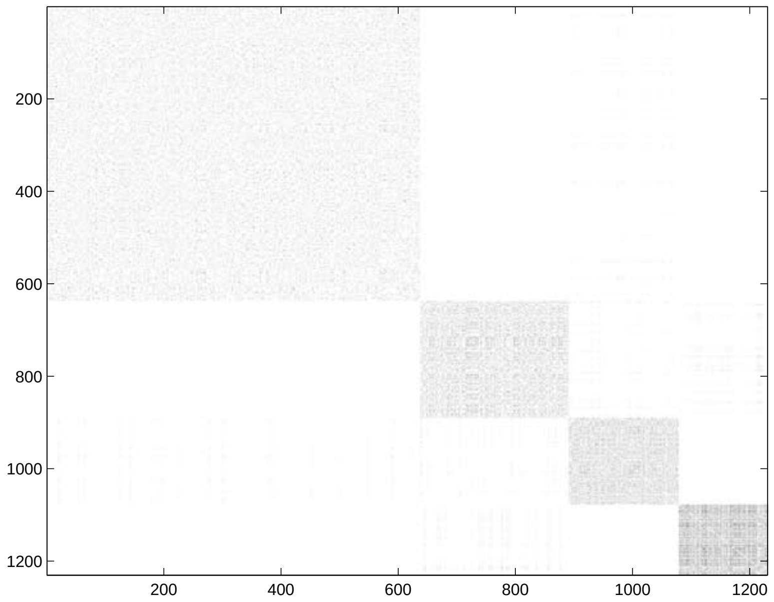

Fig. 1. **The joint similarity matrix for cricket sensory neural data set.** This matrix shows the similarity between each pair of data points as determined by the edge weights in the joint similarity graph. Similarity strength is indicated using a gray scale (dark = high similarity). The data points have been permuted into the four clusters found by the randomized projection algorithm. The clusters agree closely with previous quantizations found on the same data set. The mutual information between the quantized classes was 1.4921 bits.

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.

We are investigating both theoretical and practical improvements to the method. In particular the choice of random hash function families can likely be refined. The functions we used for the spike vectors above are basic; one can imagine more complicated projections based on the number of doublets, triplets, etc. or other patterns spanning multiple channels. 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 (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 iteration.

References

[1] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[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] Brendan Mumey and Tomáš Gedeon. Optimal mutual information quantization is NP-complete. Neural Information Coding (NIC) workshop poster, Snowbird UT, 2003.

[4] Ron Shamir and Rodid Sharan. Click: A clustering algorithm with applications to gene expression analysis. In *Proceedings of Intelligent Systems for Molecular Biology (ISMB)*, pages 307–316, 2000.

[5] Ron Shamir and Dekel Tsur. Improved algorithms for the random cluster graph model. In *Proceedings of Scandanavian Workshop on Algorithms Theory (SWAT)*, pages 230–239, 2002.

[6] 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.