Data-mining of time-domain features from neural extracellular field data

Samuel Neymotin[1], Daniel J. Uhlrich[2], Karen A. Manning[2], and William W. Lytton[1]

[1] SUNY Downstate Medical Center Dept. Biomedical Engineering, Brooklyn, NY
samn, billl at neurosim.downstate.edu
[2] University of Wisconsin Dept. of Anatomy, Madison, WI
duhlrich, kamannin at wisc.edu

Summary. Spike-wave and polyspike-wave activity in electroencephalogram are waveforms typical of certain epileptic states. Automated detection of such patterns would be desirable for automated seizure detection in both experimental and clinical venues. We have developed a time-domain algorithm denominated SPUD to facilitate data-mining of large electroencephalogram/electrocorticogram datasets to identify the occurrence of spike-wave or other activity patterns. This algorithm feeds into our enhanced Neural Query System [2]) database application to facilitate data-mining. We have used our algorithm to identify and classify activity from both simulated and experimental seizures.

Key Words: simulation, computer modeling, neural networks, neuroinformatics, query systems, data-mining, SQL, NEURON, seizures, epilepsy, spike and wave, FFT, wavelets, clustering, k-means.

1 Introduction

There are vast databases of existing neural data which are growing in number and size at an ever increasing rate. The data is of clinical and research significance, but without the proper tools to sift through it, it is of limited use [2] . Custom search, interpretation, and quantification methods are needed to optimize the use of the data. These methods and search tools should allow the investigator to visualize the available information in a clear and comprehensible manner, and allow him/her to find patterns that would be difficult to see without the aid of a computer. These types of methods come under the heading *data-mining*, informally defined as user-guided analysis of large datasets using automated techniques. Ideally, data-mining methods should be general enough to work for different types of neural data, *i.e.*, simulated in software, *in vivo*, and *in vitro* recordings, yet allow customizable options to find relevant and particular patterns.

We have developed an algorithm, named SPUD (mnemonic for slice, peak, up, down — explained below), which successfully extracts useful features from disparate types of neural data. It operates on signals in the time-domain, and the features it extracts remain in the time-domain, providing exact timing information for events of interest. This is in contrast to many popular signal analysis algorithms, such as wavelets [1] or FFT (Fast Fourier transform [5]), which operate mainly in the frequency-domain, and therefore, lack precise timing information. For the detection of spike-wave (SW), algorithms operating in the frequency-domain may have difficulty, because the timing of SWs is extremely precise. For example, a population spike, which is very brief, may be followed by a wave, which is considerably longer-lasting. This type of pattern may provide conflicting information in the frequency-domain because spikes and waves are so close together in time.

The SPUD algorithm feeds the extracted features directly into the previously developed Neural Query System (NQS)[12]. We enhanced NQS to allow for standard structured query language (SQL[4]) queries by incorporating the open source database MySQL (http://www.mysql.com). The versatile search capabilities already provided by NQS, together with standard SQL, provides for highly efficient and customizable search directly from the NEURON simulation environment [10, 3]. This makes it convenient for a researcher to run simulations and analyze their results all in one familiar environment. Using the developed techniques, we were able to find SW and polyspike-wave (PSW) patterns in both simulated and *in vivo* recordings from Sprague Dawley rats.

2 Methods

2.1 Data-mining architecture

Database setup

The data of interest may either be read in directly to the database system or run through a feature extraction algorithm. The feature extraction algorithm simplifies the data by pulling out properties deemed to be useful for the particular application. It allows viewing of the data at a higher level of abstraction by ignoring irrelevant details and giving structure to the data. The feature extraction algorithm we used, SPUD, is described below. Other feature extraction algorithms fit into this pipeline and may be used in SPUD's place.

Performing queries

Once the database is set up, the user can perform searches/queries on the data. The query syntax is described below. The results returned by the database system can be displayed visually with NQS's (described below) graphical output system and/or read into data structures of the hoc programming language that is available with the NEURON simulation environment. This allows for a bidirectional flow of information between the user and the database. Fig. 1 shows the data-mining architecture developed.

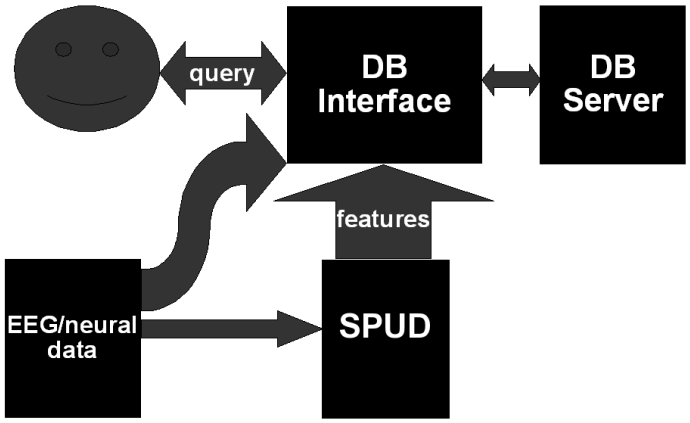

Fig. 1. Schematic of data-mining architecture. Arrows represent flow of information in a given direction. Bi-directional information transfer occurs between the user and the database (DB) system.

2.2 SPUD feature extraction algorithm

SPUD is a general algorithm for extracting information from signals in the time-domain. It works on both noisy and clean data. It was originally intended to work on signals obtained from simulations in the NEURON environment, but performed equally well on experimental data obtained from *in vivo* recordings of rats with electrocorticographic leads. One key design goal was the ability of the algorithm to extract the information characterizing the morphology and precise time properties of an arbitrary signal.

Steps of the algorithm

The SPUD algorithm is used to build a database of *bump* characteristics in the time domain, where a bump is any deviation with return to the baseline.

The algorithm proceeds as follows:

1. Slice the EEG/ECoG trace horizontally at logarithmically spaced voltage levels. These levels are more finely spaced at lower voltage levels to capture small bumps.
2. Iterate over slice positions in the horizontal direction to find matched upward and downward crossing points.
3. Find peaks between crossing points to define slice traces as a triad: upward crossing; peak; downward crossing.
4. Group slices with common peaks to define individual bumps.
5. Extract information about each bump. Information may include *e.g.,* first derivative of upstroke, first derivative of downstroke, amplitude, duration, sharpness (defined below), start time, and others.

Fig. 2 shows bumps extracted from a sample trace.

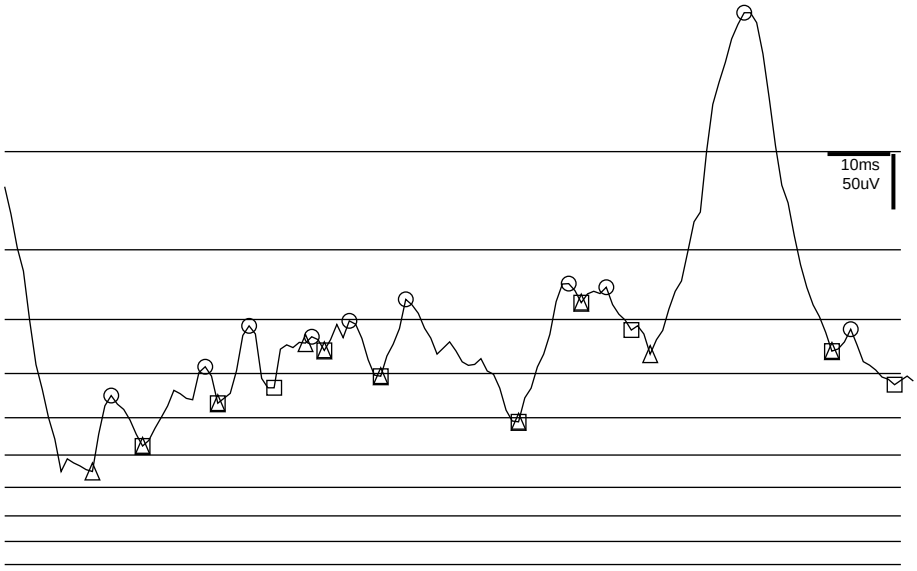

Fig. 2. Schematic showing different steps of SPUD feature extraction algorithm on recording from rat right occipital cortex. Horizontal lines are threshold locations. Open triangles, circles, and squares represent left/upward crossing position, peak position, and right/downward crossing position of bumps respectively. Note: some bumps starting and ending points are at the same time location and therefore a square and triangle marks them.

Bumps

The information extracted from the signal, which is defined as the sequence of sampled time-ordered pairs (t_i, s_i), takes the form of *bumps*. A bump, b, is a section of the signal containing a deviation from the baseline value,

$L_m \in L$ (set of threshold lines), followed by a return to that value. Given a series of samples in the signal, $s_i, s_{i+1}, s_{i+2}, \ldots s_{i+n}$, occurring at times $t_i, t_{i+1}, t_{i+2}, \ldots t_{i+n}$, and a fixed horizontal baseline, L_m, a bump begins at the first time signal pair, (t_f, s_f), such that $s_f \geq L_m$. The bump consists of the time-ordered set of pairs (t_k, s_k) such that $s_k \geq L_m$. The last pair, (t_l, s_l), where $s_l \geq L_m$ but $s_{l+1} < L_m$, defines the end of the bump. The next bump in the set may begin at the next time the signal crosses L_m. For a bump b, $\forall (t_k, s_k) \in b, t_k \geq t_f, t_k \leq t_l, s_k \geq L_m$. For notational convenience (t_f, s_f) , (t_l, s_l) , and (t_p, s_p) denote the first, last, and peak time/signal pairs in a bump. A given bump $B_i \in B$'s start, end, and peak time will be denoted $B_{i_{tf}}$, $B_{i_{tl}}$, and $B_{i_{tp}}$ respectively.

Bumps have properties associated with them which may include width $(t_l - t_f)$, height from baseline $(\max(s_i - L)$), absolute peak value $(\max(s_i)$, first derivative $(\frac{s_i - s_{i-1}}{t_i - t_{i-1}})$ at arbitrary positions within the bump, number of nested bumps, etc. The properties may be easily extended and determined algorithmically once the extents of the bump have been determined with the default algorithm. The shape of the bumps can be anything from a single action potential to the complex spiking patterns from extracellular recordings.

Thresholding

In order to extract only the *significant* bumps, a set of horizontal threshold lines, L, is used to define the baselines where bumps deviate from. This set of horizontal lines may be selected interactively by the user as it becomes apparent where significant bumps occur. Initially it can be set to either linear spacing, or logarithmic spacing. Further work needs to be done to algorithmically space the threshold lines in an optimal way depending on a given signal. The linear and logarithmic spacings allow for more control of bump extraction for different types of signals. For example, use of logarithmic spacing allows finer spacing at lower values of the signal, allowing for the extraction of smaller bumps. In general, the minimum value of the signal determines a value above which the first horizontal threshold line is placed and the maximum value determines a location below which the last threshold line is placed. These threshold lines must be monotonically increasing.

Varying threshold lines

There are several options for specifying how to space the threshold lines. Depending on the option chosen, the extracted bumps can be somewhat different. The default option is to space the lines logarithmically from 0.05 of the maximum amplitude in the trace to 0.95 of the maximum amplitude. This generally picks out the most significant bumps and smaller bumps that are placed near the bottom of the trace due to the finer spacing there. Another option which can yield better performance when the user's focus is on the middle of a trace, allows a *fan-out* logarithmic spacing whereby the start of a logarithmically spaced set of threshold lines extends upward and downward

from 0.5 of the maximum amplitude of the trace. The most simple option, which does not allow for focusing in on any particular region is to space the threshold lines linearly. In general, logarithmic spacing starting at a certain vertical position will extract finer details in the surrounding region.The user is also given the option of specifying his own threshold lines to narrow in on activity of interest. Figure 3(A,B) shows the differences in resulting bumps using 2 different sets of threshold slices.

Bump finding

Once the threshold lines, L, have been determined, they are traversed from the lowest threshold line to the highest threshold line. Each threshold line, L_m, is traversed in the horizontal direction from the start of the signal, (t_0, s_0), to the end of the signal, (t_n, s_n), in a search for matched upward, peak, and downward crossing points. To start the search for upward crossing points, a point in the signal below the current threshold line ($s_i < L_m$) is found. This point is the starting position for the search. The horizontal position is incremented until the signal crosses the threshold ($s_i \geq L_m$) line in an upward direction. This horizontal point is then noted as the left-most point of the first bump. The search then continues until a maxima is reached. This point is noted. Once again, the search continues until the signal passes the current threshold line in a downward direction ($s_i \leq L_m$) and the point noted as the right-most position of the current bump. In this fashion, each threshold line is traversed with a triad of upward, peak, and downward crossing points being extracted as the initial bumps.

Overlapping bumps

Since multiple threshold lines are used, the set of bumps found from the previous step, B, may be overlapping. This occurs with two bumps, B_i and B_j, when $B_{i_{tf}} \geq B_{j_{tf}}$ and $B_{i_{tl}} \leq B_{j_{tl}}$. This means that B_i is entirely contained within the time bounds of B_j. Overlapping also occurs when $B_{i_{tf}} \geq B_{j_{tf}}$, $B_{i_{tf}} \leq B_{j_{tl}}$, and $B_{i_{tl}} > B_{j_{tl}}$. There are two options for dealing with these types of inevitabilities. The first, and simplest, is to allow overlapping bumps. Though this is not the default behavior of the algorithm, for certain situations, this is the desired result, *i.e.*, determining the # of nested bumps in a signal/time interval. Overlapping bumps may also help determine the bounds of low frequency, long duration bumps in the presence of high frequency components. For example, in fig. 2, the first 6 low amplitude bumps are high frequency fluctuations on a longer low frequency deflection. In other situations, overlapping bumps can have their starting and ending times adjusted. This is done by grouping threshold slices with common peaks to define individual bumps. First, the bumps are traversed in increasing time order and checked for overlaps on the left or right sides. A left overlap is defined as the start of the current bump, being between the start and the peak time of the

previous bump ($B_{c_{tf}} > B_{c-1_{tf}}$ and $B_{c_{tf}} < B_{c-1_{tp}}$). In such a case, the threshold lines are traversed from $\min(L)$ to $\max(L)$ to find the lowest threshold containing a bump with the same peak time as the current bump's peak time and containing a starting time greater than the previous bump's ending time (all threshold lines have their associated bumps stored during initial bump extraction). The current bump's starting time is set as this threshold line's peak time. The same idea is used to check for and correct overlaps on the right side.

Creeping

At this point there is a non-default option allowing a bump $B_i \in B$'s $B_{i_{tf}}$ and $B_{i_{tl}}$ and associated signal values to *creep* to a local minima. This means decreasing the starting time until it is a local minima, *i.e.*, having a signal value less than the surrounding signal values ($s_i < s_{i-1}, s_i < s_{i+1}$), and increasing the starting time in a similar fashion. In certain situations, this helps get a more accurate estimate of a bump's time bounds, i.e., if a threshold line is much higher than the local minima found by creeping.

Bump properties

Once the time bounds of the bumps have been set, the other properties of a bump are extracted. The values we had the algorithm extract were: start time, peak voltage amplitude, peak time location, time from start to finish (*width*), sharpness, and when overlapping is allowed, the number of other bumps nested within each bump's start and end times. Others features can be readily added. The sharpness was defined as $(s_i - s_{i-4}) - (s_{i+4} - s_i)$, where s_i is the sample corresponding to the given bump's peak time location. This measure approximates the discretized 2nd derivative, but takes into account more information from the surrounding values. It can easily be modified depending on the intended usage. We also extracted the base voltage level of the bump on its left and/or right side and as a result we were able to query for the height of the bump, which is defined as peak voltage level minus base voltage level.

Extracting upside-down bumps

Since extracellular field recordings have different properties when they are displayed upside-down vs. right-side up, it is desirable to allow the user to extract bumps from both the original trace and the inverted trace. This allows for the extraction of *dips*, or inverted bumps, which will yield additional information. Figure 3 shows the two possible sets of bumps extracted using the normal, and the upside down version of the trace. Note that SPUD can automatically extract both sets of bumps by setting a parameter.

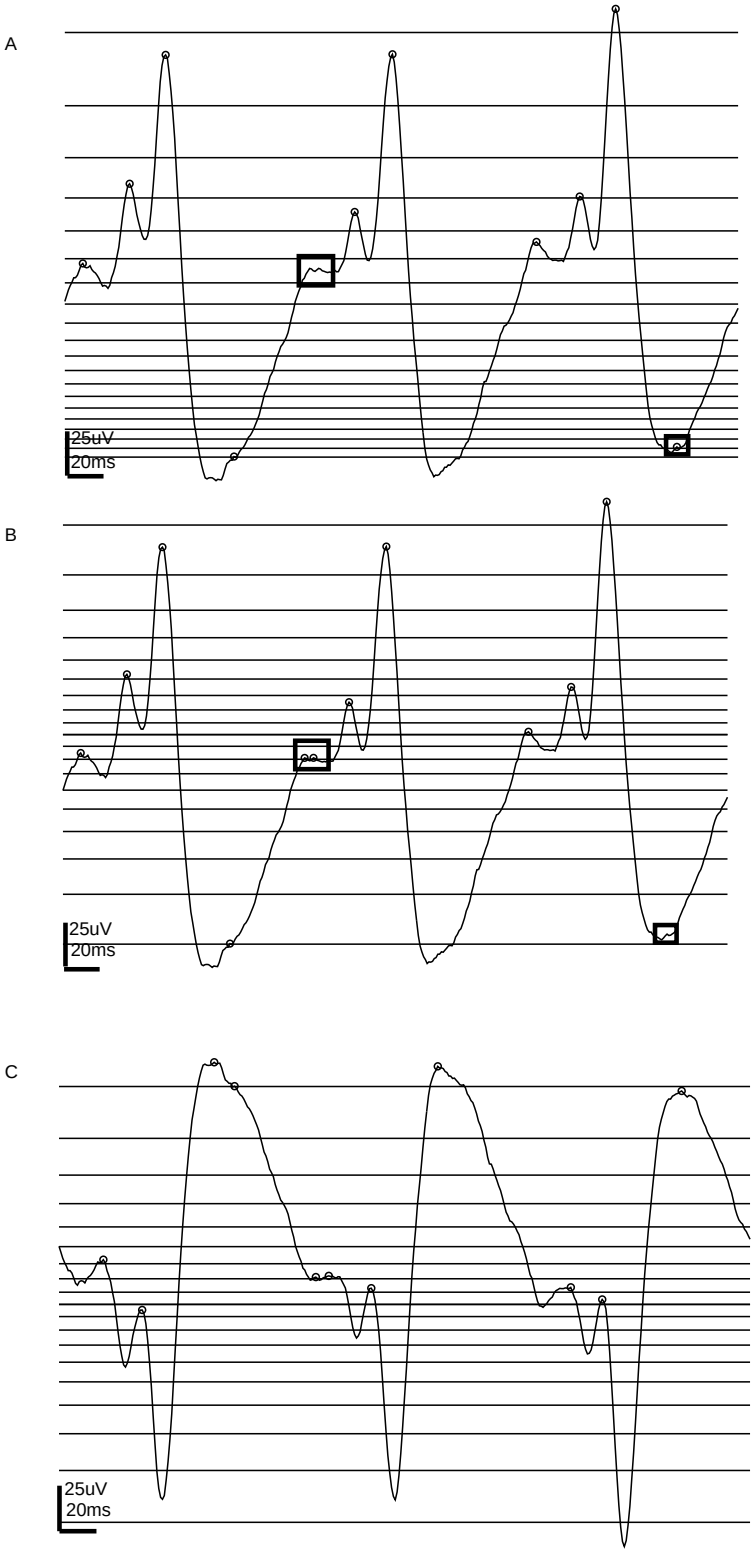

Fig. 3. Traces obtained by averaging all triplets of photic responses across 25 trials. Spacing the threshold slices (horizontal lines) differently may result in different bumps being extracted. Bump peaks represented as circles. Differences in extracted bumps between sets in A and B slices highlighted by rectangles. (A) Logarithmic spacing starting near bottom of trace. (B) Logarithmic spacing centered at middle (fan-out). (C) Dips extracted as bumps by flipping trace over x-axis.

Run-time complexity

A typical trace of neural data in our data set was sampled with a frequency on the order of milliseconds and was recorded for at least a few seconds, typically 9. This results in a vector of 9000 time points and corresponding voltage levels. The main portion of the SPUD algorithm is the traversal through each point in a given trace, once for each threshold line, in the search for bumps. There are typically 10 or so threshold lines, a number significantly smaller than the number of points in a trace.

The extraction of a bump feature is taken to be approximately constant. The number of bumps in a trace is also significantly less than the number of points in a trace, usually by a 30-fold reduction or more (*e.g.,* 300 bumps in a 9000 millisecond trace).

The main steps of the algorithm consist of iterating over the threshold lines for each time point and extracting bumps. Then for each bump, features are extracted which has a constant cost. The time complexity of the algorithm can therefore be approximated as $|L| \times |B| \times |S|$ where $|L|$ is the number of threshold lines, $|B|$ is the number of bumps found, and $|S|$ is the number of sampled points in a trace. When removing overlapping bumps, additional terms proportional to $|B|^2$ may be added to the run-time complexity. This is because, for a single given bump, $b \in B$, overlapping with another bump, the search for the modified starting/ending time is linear in $|B|$ in the worst case (the average will be far less). Since there are $|B|$ bumps, this results in an additional cost of $|B|^2$. The total run-time complexity will then be $|L| \times |S| \times |B| + |B|^2$, with the dominating term being $|S|$. As a consequence the algorithm will have an $o(|S|)$ run-time. The results of a signal may be analyzed many times in a data-mining framework and consequently, the run time cost of bump extraction can be viewed as an amortized payment which has returns each time a query is performed by the user.

In runtime complexity, SPUD is competitive with FFT which is $\theta(|S| \times \log(|S|))$ [11], and with the the fast wavelet transform using the lifting scheme, which is $o(|S|)$ when using finite filters [15].

2.3 NQS and MySQL

Overview

The next step in our data-mining pipeline was storing the extracted bumps and their associated information into the Neural Query System database as well as a MySQL table to allow data-mining of features.

NQS

The Neural Query System (NQS) is a relational database system built into the NEURON simulation environment. It allows storage of structured records and a flexible search syntax on these records. It works seamlessly with NEURON's

data types, including Vectors, Lists, strings, scalars and neural or network models. It also allows storage of sub-tables within tables. This is very handy for storage of complex data types. NQS also provides for graphical display of data, aiding in finding patterns in the data. Using NEURON with NQS, it is possible to run large simulations and store partially analyzed data quickly and easily from one environment.

MySQL

The MySQL database system (http://www.mysql.com/) has been in use for over a decade and is open-source. It has a large user base familiar with Structured Query Language (SQL). Though NQS has an intuitive syntax and much in common with SQL, there are differences in the details of the two query languages. Requiring users who already know SQL to learn NQS syntax is a possible hindrance to fast learning of the data-mining system. Although NQS has a convenient front-end and runs quickly on fast machines, it does not have the benefit of more than a decade of open source contributions from a wide community of programmers. For these reasons, we decided to create an interface between NQS and MySQL that would maintain the strengths of both systems — maintaining the ease of use of NQS and SQL, as well as the optimizations available in MySQL.

NQS/MySQL Interface

The interface we created enhances NQS with the ability to perform standard SQL queries using MySQL's version of SQL. The interface was written with the MySQL C application programmer interface (API) being translated into blocks in NEURON's NMODL language [9]. NMODL allows C code to be compiled into the NEURON environment and then called once it is running. The API we wrote connects directly to a MySQL database server from within NEURON. This allows conversion of NQS databases to MySQL databases and vice versa. It also allows for SQL queries to be performed on MySQL databases and then reading the returned data into NEURON data types. Exchanging data and queries between NQS and a MySQL server becomes quite easy. Once the queries are done, the user may retrieve their results into an NQS table.

Function overview

The main interface consists of functions for connecting to the MySQL server as well as performing queries. Once the queries have been made, their results may be obtained with helper functions. These functions read the data directly into NEURON data structures. There are other helper functions made to help search tables in the database. A typical session in NEURON will consist of connecting to the MySQL server and exchanging queries and data back and forth. We also added some functions written in NEURON's scripting language, hoc, to allow for the conversion between NQS tables and MySQL tables easily, as well as a function to help create MySQL tables programmatically. These functions make it very easy to use MySQL and NQS together efficiently.

2.4 Simulation setup

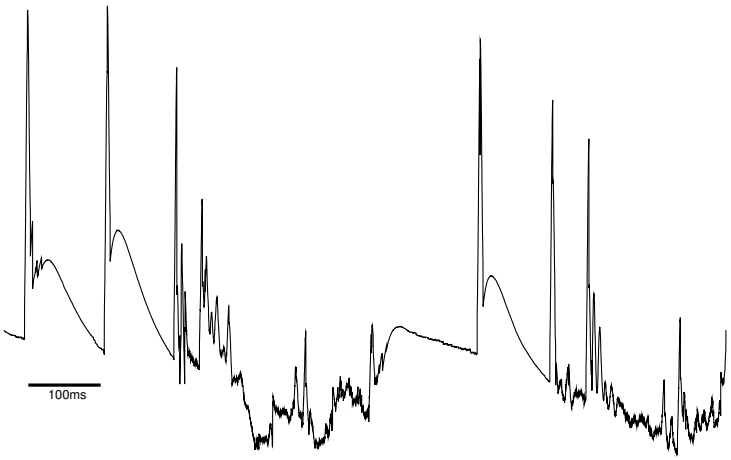

Fig. 4. Single trace from simulation. In this and following traces derived from simulation, voltage is in arbitrary units.

The simulations were run in NEURON using *tonic-clonic* simulation networks previously described [13]. The networks consisted of 1350 neurons comprising three types of cells: spontaneously active cells (drivers), inhibitory cells, and expressors (main excitatory cell population). 138,240 traces of extracellular field potentials were produced. Realistic patterns of activity, such as SW, were present (a sample trace with SW activity is shown in figure 4). The traces were then stored in the NQS system along with the 8,205,557 bumps and corresponding features extracted with the SPUD algorithm. The enhanced NQS system also converted the resulting database into a MySQL database and indexed each column to allow for real-time/interactive searching of the full dataset. Having to sift through such a massive set of data without the aid of a flexible data-mining tool would have prohibitive time costs and most likely introduce errors.

2.5 Rat data recording setup

Materials

Data was acquired from normal adult Sprague Dawley rats using BrainWare (TDT), which employs an electrode channel window for electrophysiological recording. We recorded the electrocorticogram (ECoG) at a sampling rate of 25kHz from a stainless steel wire placed on the dural surface over right occipital cortex; a second lead located over the cerebellum served as the differential reference [16].

Fig. 5. ECoG response from photically-sensitized rat in response to a train of stroboscopic flashes presented at 8 Hz. Vertical lines represent occurrence of strobe flashes. The first flashes in the strobe train elicited a photic driving response that morphed into PSW activity.

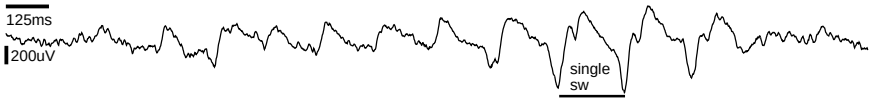

Fig. 6. ECoG recording from right occipital cortex during a PTZ-induced seizure with SW discharges.

Experimental conditions

There were two experimental conditions used for recording. In all recordings, the rat was awake and moving freely.

Epileptiform behavior via photic-induced sensitization The first experimental condition involved photic-induced sensitization following repeated exposure to trains of stroboscopic flashes [16]. One strobe train consisted of 17 flashes presented at 8Hz. Trains were presented every 30 seconds and 30-40 trains were presented during a daily recording session. On initial train presentations, the responses were small, showing at most an entrained sinusoidal driving response. However, over the course of three sessions of strobe exposure, the response grew in magnitude and acquired epileptiform characteristics like SW morphology (Fig. 5).

Chemically-induced seizures The rat was injected with 24 mg/kg of pentylenetetrazol (PTZ), a convulsant agent [7] . We started recording as soon as the rat was injected. SW seizures appeared within 2-7 min of PTZ injection (Fig. 6).

Aritifact rejection In some of the recorded traces there were artifacts introduced by the recording equipment or sudden animal movement. These artifacts were generally a voltage value of 0 or amplifier maximal value. Therefore, before running our algorithms on these traces, we preprocessed them by setting outliers/artifacts to an average of the surrounding voltage levels programmatically. Outliers were considered as any value that differed from the median voltage level of that trace by a predetermined threshold. Our algorithms and tools were then tested on several hundred of these preprocessed traces.

3 Results

3.1 Simulated data

Data-mining for SWs

Once the bumps and their corresponding information were stored in the database, we were able to data-mine for patterns of interest. We were able to use bump properties to classify the simulated EEG traces shown in figure 7A. A simple search looked for all traces with bumps having height above the 95th percentile, and sharpness above the 95th percentile. This was done by first creating a table in NQS storing the percentile values of several properties of interest such as height, sharpness, and number of bumps in a trace. Percentile tables were then created with a few function calls. Then the following NQS query was used to select all traces having more than 3 bumps above the 95th percentile of height, and 3 bumps above the 30th percentile of sharpness:

```
sq.select("HE95" , ">" ,3, "SH30", ">" ,3, "NUM", "<" ,10)
```

This query was returned traces with spike-wave activity. Some results are shown in figure 7. If it is most important to identify all seizures, a liberal criteria is used in the search which will result in a potentially higher number of false positives. On the other hand, a more conservative select is used in order to collect a few clear seizures,

SQL SW select

An alternative approach to finding SW activity was to produce an enhanced NQS/SQL query that returned times of spikes and times of waves. This was done by selecting all bumps from a given trace having two consecutive bumps, a and b, where a is a sharp, high amplitude spike, and b is a blunt, wide, and relatively low amplitude bump. The following SQL query was used to find all the spike and wave bumps satisfying these properties from a particular trace:

```
select b1.id, b2.id from bumps as b1, bumps as b2
where b1.peak > 1.33 * b2.peak and
b2.sharp < 200 and b2.width >= 15 and
b1.trace=6 and b2.trace=6 and
b1.id < b2.id and
abs(b1.id-b2.id) < 2 and
b1.sharp > 3312*1.5 and
b2.sharp < 3312/2 and
b1.sharp > 30*b2.sharp;
```

This query returns all 6 of the SWs from the given trace shown above in figure 8.

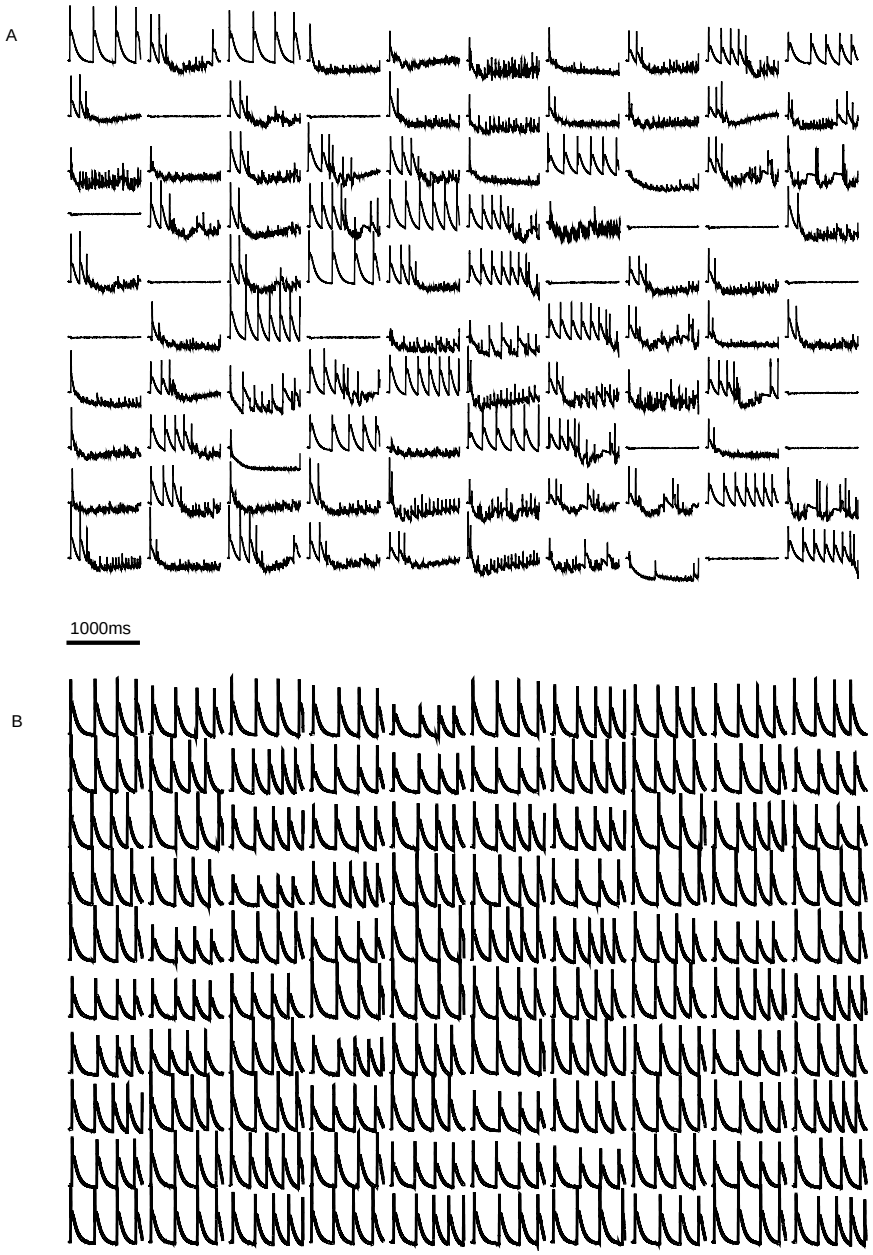

Fig. 7. Simulation traces. (A) 100 traces of 1000ms duration each, from tonic-clonic simulation described in methods section. There are many different types of patterns in these traces, including SW activity. (B) Some of the traces returned with the NQS *percentile-based* select described below. These traces were selected from the data set of 138,240 simulation traces, some of which are shown in A.

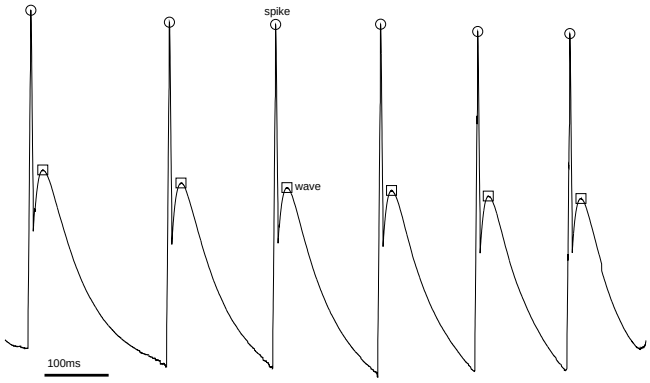

Fig. 8. This figure shows that all 6 SWs in the simulated EEG trace were returned with the SQL select mentioned below. All spike peaks returned by the SQL select are marked with open circles. All wave peaks are marked with open squares. As shown, all of the SWs in the trace are found.

System performance

Using the SQL query above, the data-mining system was able to search through the entire bump database, which is on the order of 8e6 bumps, and return results in less than 1ms. This performance level is useful for a fast back and forth between investigator and the data-mining system.

3.2 Rat data

Classification algorithms

To help find different types of activity patterns in the *in vivo* electrocorticogram recordings of rat epileptiform activity several types of classification algorithms were used on the bumps. One such application was in determining whether a PSW occurred during a PPR response. This involved clustering the time intervals based on their bump content. We also developed an algorithm to determine the time bounds of the PTZ induced seizures by analyzing the bump properties in a trace.

Clustering with k-means

Classifying time windows in traces into background activity vs. PSW is useful for detecting the onset and duration of seizures and/or epileptiform activity. We were able to use k-means clustering [14, 8] to classify intervals into two classes. K-means used 4-D vectors consisting of the un-normalized maximal bump width, height, peak, and sharpness from the bumps of a given interval. The time intervals were cut to be 125ms, which is the time between successive strobe flashes. Although PSWs typically occur in between strobe flashes, we

did not limit the clustering solely to those intervals so that we would obtain accurate background vs. PSW activity. When we only used the time intervals between strobe flashes, the false positive/negative rates increased. Once the k-means clustering was performed, there were two classes - one with a high amplitude width, height, peak, and sharpness, and one with a significantly lower amplitude of these values. Though k-means has an element of randomness in it, running it thousands of times helps ensure an accurate result. This is done by choosing the cluster centroids that occur the majority of the time as the final cluster centroids. All time intervals are classified accordingly. The final centroids of the two classes came out with these ratios for the PSW/non-PSW values : peak 1.67, width 2.75, sharpness 1.77, height 3.3. It is clear from these ratios that the PSW intervals have significantly higher values of all the properties used for classification. This difference in magnitude of the properties allowed the algorithm to determine which class had PSWs and which did not.

The resultant clusters, shown in fig. 9, separating between PSW and non-PSW intervals correlated well with the results of the algorithm described below, providing a measure of cross-validation. The results were also visually inspected for accuracy. Though certain intervals are mis-classified, overall the method does well. Since the distinction between PSW and driving activity is sometimes blurry, errors may be reduced by thresholding the results of a fuzzy k-means algorithm [6]. However, there are certain sharp borders in this classification scheme: based on visual inspection, PSWs detected outside of the strobing windows are always false positives. An NQS select shows that there are 21 PSW cycles detected outside of this time interval (before 3000ms or after 5000ms) out of a total of 243 total PSW cycles detected. That is an approximately 8.6% false positive rate. False negative rates are more difficult to obtain due to the blurry border between PSW and non-PSW responses (see discussion). More generally, the clustering of the time intervals may be viewed as another step in the data-mining pipeline, where the researcher filters the data sets to find ever more particular information. Though this method has the additional information of optimal time window length, it may be possible to dynamically determine this based on bump properties for other types of traces.

Detecting time bounds of seizures

To detect the time bounds of the PTZ-induced seizures, we could not rely on a clustering algorithm like k-means, because there were no predefined time windows that would be guaranteed to work. Therefore we used a strategy based on the fact that in general, the seizure regions of the trace consisted of *tall* and *wide* bumps with other small bumps in intermediate positions. The presence of seizures could then be indicated by positive deviations from the mean width and height of the bumps in a trace by a certain factor. The first step in detecting the seizure was iterating over the set of bumps and searching

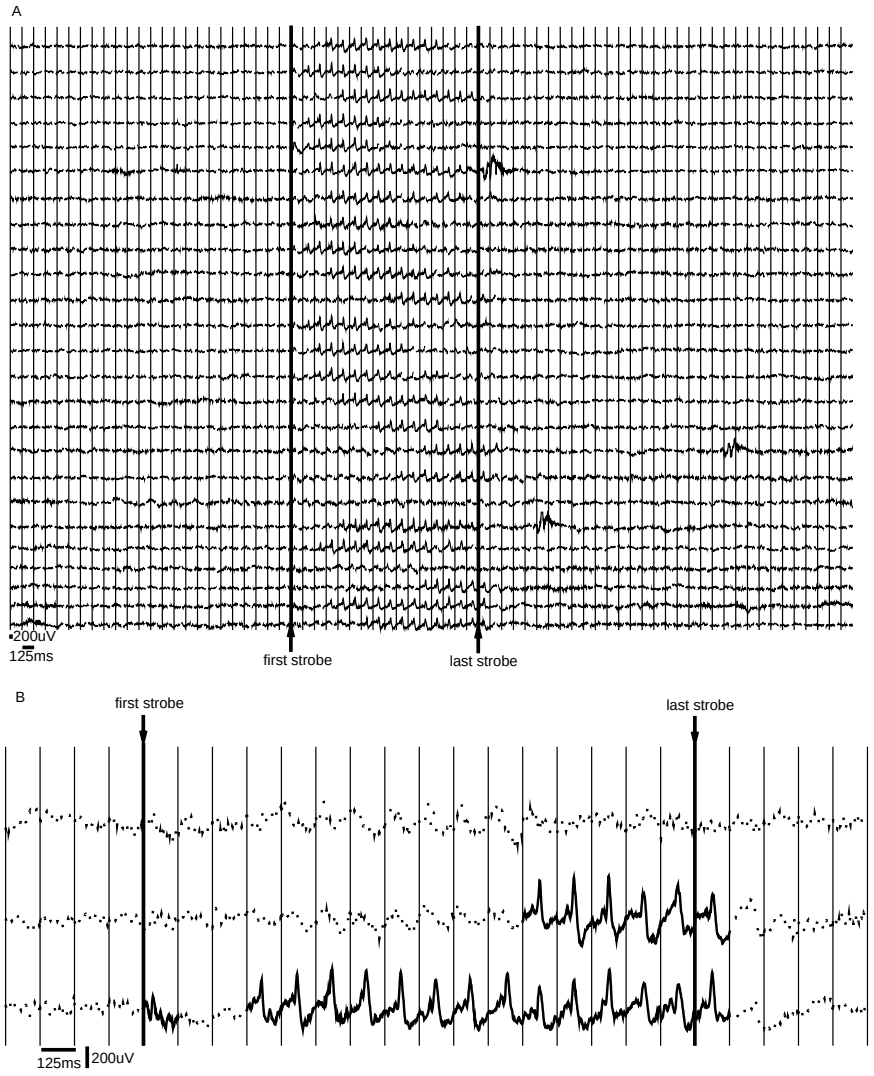

Fig. 9. K-means clustering on strobe data traces. Vertical lines indicate time windows for bump analysis. Each time window is 125ms. Thick, dark lines represent PSW type activity. Thinner lines indicate absence of PSWs. Length of each trace is 9000ms. PSW activity begins shortly after onset of strobing. Vertical solid lines represent time interval borders for clustering and 8Hz strobe times (between first and last strobe only). (A) Full set of results. (B) Zoom in of portion of traces.

for tall and wide bumps. These bumps were designated as *seed* bumps. The qualifications for a seed bump that worked well for the PTZ seizures were that the height be greater than 4.5× the mean height and the width be greater than 3× the mean width. Once the seed bumps were found, we had a rough estimate of where seizure activity was likely to occur. We next had to find the extent of the seizure. To find where the seizure started and ended, we determined positions to the left and right of the seed bump where the bumps returned to baseline size for a selected time duration (50 ms in the current analysis). This prevents misclassification due to small bumps occurring within a seizure. To avoid false positives, we also had an option of excluding very short seizures. This method was able to extract all of the significant seizures from our initial data set. A few sample results are shown in fig. 10.

This algorithm performed well on the photic-induced epileptiform activity as well (results shown in fig. 11), but we had to specify different parameters to the algorithm, such as the deviation from mean bump width and height required for a bump to qualify as a *seed* bump, as well as the minimum time span required for baseline activity, which was reduced to 12.5 ms. The fact that the algorithm performed well on various types of traces shows it has usefulness as a general detection algorithm for epileptiform activity. The results of this algorithm on the photic-induced epileptiform activity also correlated highly with the results of the k-means clustering. This lends support to both methods and to the feature extraction algorithm's robustness. The run-time complexity of this algorithm is essentially linear in the number of bumps in the time interval being analyzed, since it essentially traverses the bumps in order.

Data-mining seizure properties

Once we had the PTZ seizures extracted we used data-mining to look for different types of correlations between properties of bumps and properties of seizures. For example, we found that the maximum bump height and width of a PTZ-induced seizure had a positive correlation with the duration of the seizure, with values of 0.856 and 0.819 respectively. This is shown in a scatter plot in fig. 12.

4 Discussion

Data-mining system

The combination of the feature extraction algorithm and the NQS search software, provides a powerful data-mining tool for finding patterns of interest. The SPUD algorithm we developed is able to efficiently extract features of interest in the time-domain. We have shown that more complicated frequency-domain methods are not always needed. We have also shown that the data-mining framework can be used with various *ad hoc* techniques to interact

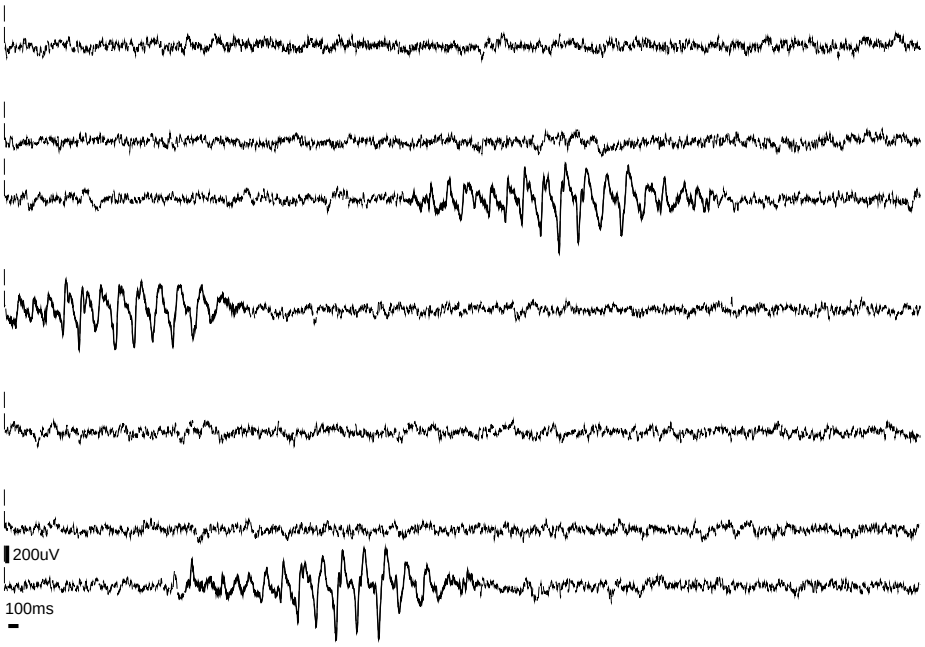

Fig. 10. Recordings from the rat's right occipital cortex after PTZ administration. Dark black lines indicate extents of seizures detected algorithmically. Each trace is 9000ms.

with the feature database and pull out interesting and physiologically complex patterns. This was done with simple algorithms as well as with intuitive, easy to use NQS/SQL selects. Once patterns were pulled out, it was possible to find interesting correlations between different properties of the data sets.

Quantification of neural data

One of the goals of our quantitative analysis is to provide measures that could be used to define what is to be considered a seizure and what is not. There is no gold standard for seizure definition since there is a continuum of electrographic activity appearance resulting in a large grey area where abnormal activity might or might not be defined as epileptiform depending on the criteria used. In order to estimate classification accuracy and error rates in the future we would like to develop several criteria-sets that operate along different dimensions and use these for cross validation.

Applications using extracted features

Using the features extracted with the SPUD algorithm it was also possible to determine the time bounds of several different classes of activity including PTZ-induced seizures and photic-induced epileptiform activity. This will have

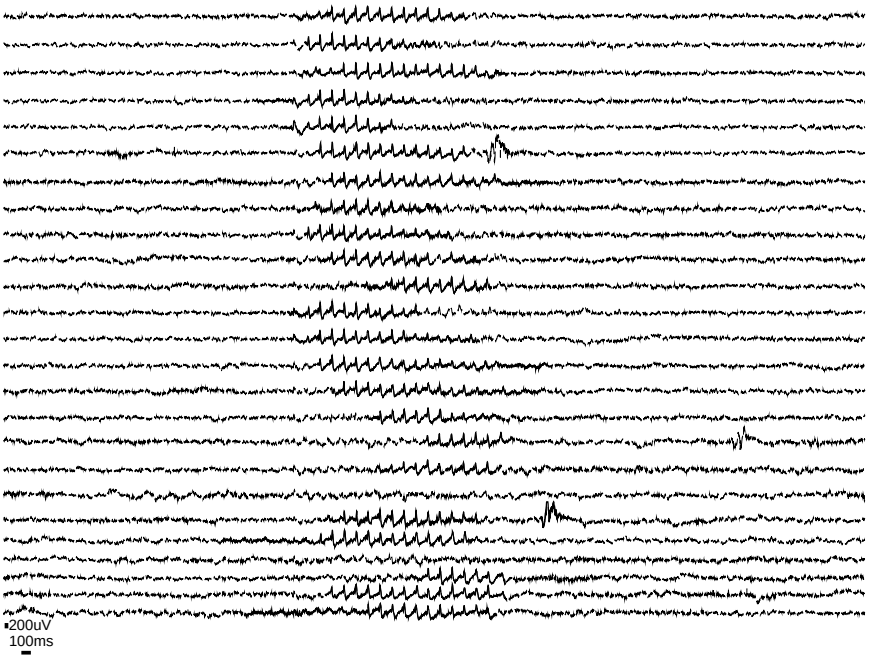

Fig. 11. Recordings from the rat's right occipital cortex during strobing trials. Dark black regions indicate extents of epileptiform activity detected algorithmically. Each trace is 9000ms. Note high correlation between epileptiform activity detected here and PSWs detected with k-means clustering (same data as Fig. 9).

both research and clinical usefulness. Clinical applications may involve a real-time seizure detector that extracts features from recordings of an epileptic patient and detects when they begin with algorithms similar to those here developed. Research applications include searching for the underlying features of bumps occurring pre, post, and during seizures. This may shed light on network and neuronal dynamics leading to seizure genesis. Future work will involve the comparison of different classes of epileptiform activity, *i.e.,* PTZ vs. photic-induced, by quantifying differences in the bumps and their associated properties. We also plan on quantifying and tracking seizure genesis by watching how bump properties change spatio-temporally.

Another future direction is extending the seizure detection algorithms developed to cover a wide class of seizures. Other types of neural activity may also be analyzed using the feature extraction algorithms developed. Future work may entail analyzing the correlations between activity properties in different brain regions as well as spatio-temporal patterns during specific behaviors. This may help uncover the interplay between different brain regions. More generally, the feature extraction methods developed will help organize the vast amounts of neuronal information currently available.

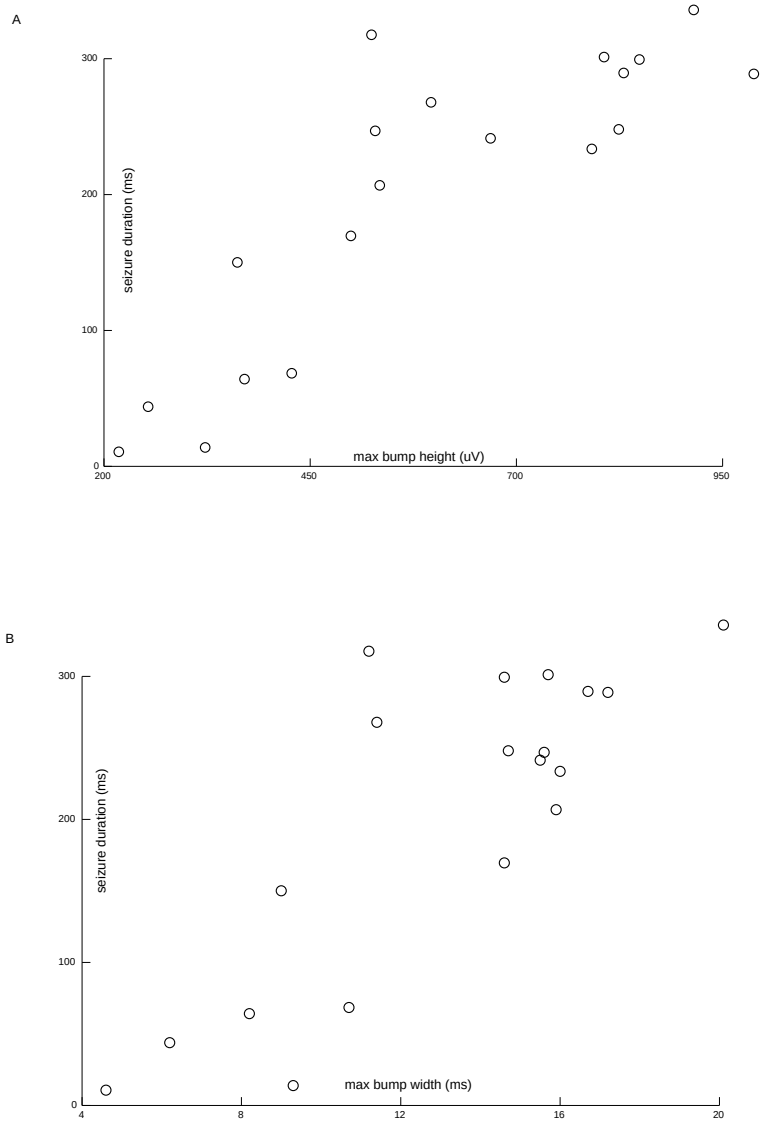

Fig. 12. Correlations between bump properties during a PTZ-induced seizure and the seizure properties. (A) Correlation between maximum bump height in a PTZ-induced seizure vs. that seizure's duration. Computed correlation coefficient was 0.856. (B) Correlation between maximum bump width in a PTZ-induced seizure vs. that seizure's duration. Computed correlation coefficient was 0.819.

Frequency-domain measures

Frequency-domain measures from wavelet or Fourier analysis can be readily added to the databases to provide a still richer information matrix for pattern extraction. This added information may help in understanding neuronal data. It may also reduce false positive/negative rates in pattern extraction algorithms. Note that some of the attributes that we pull out in the time-domain, particularly bump duration and sharpness, are correlates of frequency. This is not a problem in data-mining: we have no need to eliminate redundancy and simply seek to include as many features as may be useful.

Compression

Due to the large size of raw trace databases, and potentially limited storage space, it may be desirable to store extracted features in place of full traces for some applications. Corresponding segmental traces could be stored with bump features to minimize loss of information. This can result in an order of magnitude compression in addition to permitting arbitrary activity-pattern searches to be done across large amounts of physiological data. Researchers will be more likely to store their data in central repositories if they can be readily stored and easily accessed.

5 Download

The software discussed is available at ModelDB:
http://senselab.med.yale.edu/ModelDB

6 Acknowledgments

The authors wish to thank Michael Hines and Ted Carnevale for continuing assistance with NEURON. The simulations and data analysis were performed at the Neurosim lab at SUNY Downstate and the authors would like to thank the Neurosim staff, particularly Larry Eberle, for long hours of advice and assistance. Simulation and data-mining were performed under NIH NS045612. Physiological recording experiments were conducted at the U. of Wisconsin and were supported by NSF grant IOB-0445606.

References

1. Adeli H, Zhou Z, Dadmehr N (2003)
Analysis of EEG records in an epileptic patient using wavelet transform.
J Neurosci Methods. 2003 Feb 15;123(1):69-87.

2. Ascoli GA, De Schutter E, Kennedy DN (2003)
An information science infrastructure for neuroscience
Neuroinformatics 1 (1): 1-2 SPR 2003
3. Carnevale NT, Hines ML (2006)
The NEURON Book.
Cambridge, UK: Cambridge University Press, 2006.
4. Chamberlin DD, Boyce RF (1974).
SEQUEL: A structured English query language
International Conference on Management of Data, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control, Ann Arbor, Michigan, pp. 249-264.
5. Cooley JW, Tukey JW (1965)
An algorithm for the machine calculation of complex Fourier series
Math. Comput. 19: 297-301.
6. Dunn JC (1973)
A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters
Journal of Cybernetics 3: 32-57
7. Golarai G, Cavazos JE, Sutula TP (1992)
Activation of the dentate gyrus by pentylenetetrazol evoked seizures induces mossy fiber synaptic reorganization.
Brain Res. 593:257-64.
8. Hartigan JA, Wong MA (1979)
A K-Means Clustering Algorithm
Applied Statistics 1979 28:100-108
9. Hines ML, Carnevale NT (2000).
Expanding NEURON's repertoire of mechanisms with NMODL.
Neural Comput. 2000 May;12(5):995-1007. Review.
10. Hines ML, Carnevale NT (1997)
The NEURON simulation environment.
Neural Comput. 1997 Aug 15;9(6):1179-209. Review.
11. Johnson SG, Frigo M (2007)
A Modified Split-Radix FFT With Fewer Arithmetic Operations
IEEE Transactions on Signal Processing, 55:111-119
12. Lytton WW (2006)
Neural query system: data-mining from within the neuron simulator
Neuroinformatics, 4:163-176
13. Lytton WW, Omurtag A (2007)
Tonic-clonic transitions in computer simulation
Journal of Clinical Neurophysiology, 24:175-181
14. MacQueen JB (1967)
Some Methods for classification and Analysis of Multivariate Observations
Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability
Berkeley, University of California Press, 1:281-297
15. Sweldens W (1997)
The Lifting Scheme: A Construction of Second Generation Wavelets
SIAM Journal on Mathematical Analysis, 29:511-546
16. Uhlrich DJ, Manning KA, O'Laughlin ML, Lytton WW (2005)
Photic-Induced Sensitization: Acquisition of an Augmenting Spike-Wave

Response in the Adult Rat Through Repeated Strobe Exposure
Journal of Neurophysiology 94:3925-3937

Index