Frequency Components of Neural Signals

- < 300 Hz: Local Field Potentials (LFP)
  - < 3 Hz: Delta (slow wave sleep)
  - 4 – 7 Hz: Theta (wake and REM)
  - 8 – 12 Hz: Alpha (drowsiness)
  - 12 – 30 Hz: Beta (decision-making, motor planning)
  - 36 – 100 Hz: Gamma (sensory attentive)

- 300 – 6000 Hz: Single-Unit Activity (“spikes”)
Agenda

• Neural Spike Sorting
  – Spike Detection
  – Feature Extraction
  – Clustering

• Next time: LFP processing

Spike Sorting

• In a neural recording, one electrode typically receives electrical signals from multiple (1-7) neurons simultaneously.

• Spike Sorting: The process of assigning spikes to different neurons.

• Why spike sort?
  – Functional reasons
    • Adjacent cells can encode completely different information
    • Neuroscientists want to know which spike came from which neuron in order to learn how the brain processes information
    • BMIs often depend on single-unit activity as input
  – Practical reasons
    • Electrode arrays have increasing numbers of channels (100, 128, 256, 1000, etc.) → increasing amount of data
    • Need to compress large amounts of data in order to transmit wirelessly
Spike Sorting

- Spike sorting is possible because each neuron produces a different, distinct spike shape during a recording that remains constant throughout the recording.

- Three Main Steps
  1. **Spike Detection**: Separating spikes from noise.
  2. **Feature Extraction**: Transforming spikes into a certain set of features (e.g., principal components).
  3. **Clustering**: Classifying spikes into different groups (neurons) based on extracted features.

Some Existing Algorithms

1. Spike Detection
   - Amplitude Thresholding
   - Energy Operators
   - Wavelet Transform

2. Feature Extraction (FE)
   - Principal Component Analysis (PCA)
   - Wavelet Transform
   - Integral Transform (IT)
   - Discrete Derivatives (DD)

3. Clustering
   - k-Means
   - Single-Linkage
   - Valley Seeking
   - Superparamagnetic Clustering (SPC)
   - Online clustering
Spike Detection: Thresholding

- **Strategies:**
  - one (or multiple) voltage thresholds
  - static or adaptive
  - windowing (incorporate limited time ranges)

7.7

Spike Detection: Errors

- **False Negative**
  - Missed spike (perhaps overlapping)
- **False Positive**
  - “Detected” noise or misclassified cell

7.8
Spike Detection: Overlapping Spikes

- Really tough problem...

Figure 5. The peak level of the neuron of interest can change dramatically depending on the location and size of adjacent spikes.

Spike Detection Algorithms

Apply threshold to...
- Amplitude of signal
- Absolute value of amplitude

\[ Thr = 4\sigma_N \]
\[ \sigma_N = \text{median} \left\{ |x(n)| \right\} \]

7.9

7.10
Spike Detection Algorithms

Apply threshold to...

- **Nonlinear Energy Operator (NEO)**

\[ \psi[x(n)] = x^2(n) - x(n+1) \cdot x(n-1) \]

\[ Thr = C \frac{1}{N} \sum_{n=1}^{N} \psi[x(n)] \]

- **Stationary Wavelet Transform Product (SWTP)**

1. Calculate SWT at 5 consecutive dyadic scales:

\[ W(2^j, n), j = 1, \ldots, 5 \]

2. Find the scale \( 2^{j_{\text{max}}} \) with the largest sum of absolute values:

\[ j_{\text{max}} = \arg \max_j \left( \sum_{n=1}^{N} |W(2^j, n)| \right) \]

3. Calculate point-wise product between SWTs at this scale and at the two previous scales:

\[ P(n) = \prod_{j=j_{\text{max}}-3}^{j_{\text{max}}} |W(2^j, n)| \]

4. Smooth with Bartlett window

5. Threshold:

\[ Thr = C \frac{1}{N} \sum_{n=1}^{N} w(n) \cdot P(n) \]
Feature Extraction Algorithms

- Principal Component Analysis (PCA)
- Integral Transform (IT)
- Discrete Wavelet Transform (DWT)
- Discrete Derivatives (DD)
- **Special case: tetrode recordings**

### Principal Component Analysis (PCA)

- Also known as Karhunen-Loève transform
- Projects data onto an orthogonal set of basis vectors such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component).
- The theoretically optimum transform for a given data in least square terms.
- Algorithm
  1. Calculate covariance matrix of data (spikes) (N-by-N)
  2. Calculate eigenvectors (“principal components”) of covariance matrix (N 1-by-N vectors)
  3. For each principal component \( i = 1, \ldots, N \), calculate the \( i \)th “score” as the scalar product of the data point (spike) and the \( i \)th principal component
  4. Each data point can be reconstructed by multiplying each score for that data point by each PC and summing over \( i \).
PCA for Spike Sorting

![PCA Images](image)

7.15

### PCA: Pros and Cons

**Pro**  
- Efficient (coding): can represent spike in 2 or 3 PCs

**Cons**  
- Inefficient (implementation): hard to perform eigendecomposition in hardware  
- Only finds independent axes of the data under the Gaussian assumption. For non-Gaussian or multi-modal Gaussian data, PCA simply de-correlates the axes.  
- There is no guarantee that the directions of maximum variance will contain good features for discrimination.

7.16
Integral Transform (IT)

- Most spikes have a negative and positive phase
- Algorithm
  1. Calculate integral of negative phase
  2. Calculate integral of positive phase

\[ I_A = \frac{1}{N_A} \sum_{n=1}^{N_A} x(n) \]
\[ I_B = \frac{1}{N_B} \sum_{n=1}^{N_B} x(n) \]

- Pros and Cons
  - Pros
    - Efficient implementation (accumulators, no multiplies)
    - High dimensionality reduction
  - Con
    - Does not discriminate between neurons well

Discrete Wavelet Transform (DWT)

- Wavelets computed at dyadic scales form an orthogonal basis for representing data
- Convolution of wavelet with data yields wavelet “coefficients”
- Can be implemented by a series of quadrature mirror filter banks
- Pros and Cons
  - Pro: Accurate representation of signal at different frequencies
  - Con: Requires convolutions → multiple multiplies/adds per sample
DWT: Example

Discrete Derivatives (DD)

- Like a simplified version of DWT
- Differences in spike waveform are computed at different scales:
  \[ dd_\delta(n) = x(n) - x(n - \delta) \]
- Dimensionality is reduce by performing K-S test for normality
- Pros and Cons
  - Pros
    - accurate
    - very simple implementation
  - Cons:
    - Increases dimensionality
    - K-S test increases complexity
Discrete Derivatives Example

- **Method:**\[ dd_{d}(n) = x(n) - x(n - \delta) \]

Choose features (e.g. \( d = 1, 3, 7 \)) for spike clustering

Special Case: Tetrode Recordings

- **Tetrodes:** Probe with 4 electrodes
- Electrodes are close enough together to record signals from same neurons
- Amplitude of signal from a given neuron is highest on closest electrode and lowest on farthest electrode
- This information can be used for spike sorting
- Simplifies calculations!

Takahashi, et al. Nature Bristol

Takahashi, et al.
Need for On-Chip Spike Sorting

- **Traditional** neural recording system: wired data; sorting offline in software (in vivo) → ex vivo

  - Disadvantages of traditional approach
    - Not real time
    - Limited number of channels

- **Improved** neural recording system: wireless data; sorting online, on-chip (in vivo)

  - Advantages of in-vivo system
    - Faster processing
    - Data rate reduction
    - Wireless transmission of data possible

Neural Data Recording Today

- **Example:** 100-channel recording
  - 100 channels x 25 kS/s x 10 bits = 25 Mbps (raw data)
- **Technical challenges**
  - Power density: < 0.8mW/mm²
  - Low data rate for wireless transmission
- **Our research**
  - Real-time on-chip data compression (a.k.a. “spike sorting”)
  - Rapid processing of large exiting data records

2 TB/week!
Design Challenges

- **Power density**
  - Tissue damage at 800 µW/mm²
- **Data-rate reduction**
  - Low power
  - Large number of channels
- **Low area**
  - Integration with recording array


Design Approach

- Technology-aware algorithm selection
- Energy- & area-efficient architecture and circuits
- Low-power, multi-channel ASIC implementation
Algorithm Evaluation Methodology

- **Goal:** To identify high-accuracy, low-complexity algorithms

- **Algorithm metrics**
  - Accuracy
  - NOPS
  - Area

- Generated test data sets using neural signal simulator
  - SNR: 20 dB to −15 dB

- Tested accuracy of the spike-detection and feature-extraction methods described in the “algorithm overview”

Input: Generic Spike Generator

Library of 600 spike shapes to choose from: 3 spikes in this e.g.
Spike Detection Accuracy Results

- **Left:** Probability of detection vs. SNR for each detection method
- **Right:** Probability of false alarm vs. SNR for each detection method
- Curves for each of the 96 data sets are shown. For each method, the median across all data sets is shown in bold.

7.29

**Spike Detection Accuracy Results**

- Median ROC curve for each detection method ($N = 1632$). The areas under the curves (choice probabilities) are as follows: Absolute Value, 0.925; NEO, 0.947; SWTP, 0.794

7.30
Spike Detection Complexity Results

• The median choice probability of all data sets and noise levels \(N = 1632\), versus normalized computational cost for each spike-detection algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MOPS</th>
<th>Area [mm(^2)]</th>
<th>Normalized Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute Value</td>
<td>0.4806</td>
<td>0.06104</td>
<td>0.0066</td>
</tr>
<tr>
<td>NEO</td>
<td>4.224</td>
<td>0.02950</td>
<td>0.0492</td>
</tr>
<tr>
<td>SWTP</td>
<td>86.75</td>
<td>56.70</td>
<td>2</td>
</tr>
</tbody>
</table>

Feature Extraction Complexity and Accuracy Results

• Mean classification accuracy, averaged over all data sets and noise levels \(N = 1632\), after fuzzy c-means clustering versus computational cost for each FE algorithm. Error bars show standard error of the mean.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MOPS</th>
<th>Area [mm(^2)]</th>
<th>Normalized Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>1.265</td>
<td>0.2862</td>
<td>1.4048</td>
</tr>
<tr>
<td>DWT</td>
<td>3.125</td>
<td>0.06105</td>
<td>1.2133</td>
</tr>
<tr>
<td>DD</td>
<td>0.1064</td>
<td>0.04725</td>
<td>0.1991</td>
</tr>
<tr>
<td>IT</td>
<td>0.05440</td>
<td>0.03709</td>
<td>0.1470</td>
</tr>
</tbody>
</table>
**Single-Channel DSP Kernel**

- **Data Rate (kbps)**
  - 192 kbps: 47% area
  - 16.8 kbps: 33% area
  - 3.84 kbps: 20% area

- **Nonlinear Energy Operator (NEO)**
  \[ \psi(n) = x^2(n) - x(n-1) \cdot x(n-1) \]
  \[ \text{Thr} = C \cdot \frac{1}{N} \sum_{n=1}^{N} \psi(n) \]

**Spike-Sorting DSP Chip**

- **Modular architecture**
  - 4 cores process
  - 16 channels each
  - S/P and P/S conversion
  - Voltage-level conversion

- **Modes of operation**
  - Raw data
  - Aligned spikes
  - Spike features

- **Detection thresholds**
  - Calculated on-chip
  - Independent training for each channel

- **Detected features**
  - 16.8 kbps

- **Chip size**
  - 2.66 mm x 2.66 mm

- **Technology**
  - 1P8M Std-V, 90-nm CMOS
Sample Output (Simulated Data)

(a) Raw Data $x(n)$

SNR = $-2.2$ dB

(b) Aligned Spikes $s(n)$

CA = 77 %

(c) Extracted Features $dd_s(n)$

CA = 97 %

Sample Output (Human Data)

(a) Raw Data $s(n)$

(b) Aligned Spikes $s(n)$

(c) Extracted Features $dd_s(n)$

The results confirm manual observations by a neuroscientist
**Chip Summary**

- **Power**
  - 130 μW for 64 channels
  - 52 μW for 16 channels

- **Area**
  - Die: 7.07 mm$^2$
  - Core: 4 mm$^2$

- **Classification accuracy**
  - Over 90% for SNR > 0 dB

<table>
<thead>
<tr>
<th>Technology</th>
<th>1P8M 90-nm CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core $V_{DD}$</td>
<td>0.55 V</td>
</tr>
<tr>
<td>Gate count</td>
<td>650 k</td>
</tr>
<tr>
<td>Clock domains</td>
<td>0.4 MHz, 1.6 MHz</td>
</tr>
<tr>
<td>Power</td>
<td>2 μW/channel</td>
</tr>
<tr>
<td>Data reduction</td>
<td>91.25%</td>
</tr>
<tr>
<td>No. of channels</td>
<td>16, 32, 48, 64</td>
</tr>
</tbody>
</table>

**Power density:** 30 μW/mm$^2$

---

**Clustering**

- Partitioning a data set into subsets (“clusters”), so that the data in each subset share some common trait - often proximity according to some defined distance measure
- Previous slides show scatter plots of one extracted feature vs. another (e.g., PC2 vs. PC1, amplitude on electrode4 vs. electrode1)
- In order to determine which neuron generated each spike, we must partition these scatter plots
- Types of clustering:
  - Hierarchical: Successive clusters are found using previously established clusters
    - Agglomerative – “bottom-up”
    - Divisive – “top-down”
  - Partitional: All clusters are determined at once
Clustering Algorithms

- k-Means
- Single-Linkage
- Valley-Seeking
- Superparamagnetic Clustering (SPC)
- Online

K-Means Clustering

- Most commonly-used clustering method
- Partitional
- Based on distance measure
- Algorithm
  1. Define $k$ (number of clusters/neurons)
  2. Randomly define the $k$ cluster centroids
  3. Assign each data point to the cluster with the closest (usually by Euclidean distance measure) centroid
  4. Recompute each cluster centroid as mean of that cluster
  9. Repeat Steps 3-4 until convergence criterion is met (e.g., assignments stop changing)
- Pros and Cons
  - Pro
    - very simple
  - Cons
    - Need to know $k$
    - Assumes spherical distribution of data
Single-Linkage Hierarchical Clustering

- Hierarchical, agglomerative
- Based on a distance metric
- Algorithm
  1. Initialization: Create one cluster for each data point.
  2. Find the two closest clusters clusters $c_1$ and $c_2$
  3. Merge these clusters
  4. Compute distance $D(c_1, c_2)$ between the new cluster and all of the old clusters
  9. Repeat steps 2 – 4 until either:
     - $D(c_1, c_2)$ exceeds a certain threshold (distance criterion) or
     - A certain number of clusters is reached (number criterion) or
     - All objects are in one cluster.

Requires building the entire hierarchical tree, then cutting the k-1 longest links to get k clusters.

Pros and Cons
- Pro: conceptually simple
- Cons
  - computationally complex (requires many searches)
  - not automatic (requires setting of either k, distance criterion, or number criterion)
Valley-Seeking Clustering

- Hierarchical, agglomerative (extension of single-linkage)
- Algorithm
  1. Calculate distance matrix \( D = (d_{ij}) \)
  2. Calculate neighbor number matrix \( L = (l_{ij}) \)
  3. Calculate \( S = (s_{ij}) \), where \( s_{ij} = (l_{ij} + l_{ji})/2 \)
  4. Calculate NDD matrix \( J = (J_{ij}) \), where \( J_{ij} = \frac{|l_{ij} - l_{ji}|}{s_{ij}^{1/4}} \)
  5. Estimate convexity matrix \( D2 = (d2_{ij}) \), where
    \[
    d2_{ij} = \left( \frac{1}{2} \sum_{i=1}^{n} l_{ii} + l_{ij} \right) / \left( \frac{1}{2} \sum_{i=1}^{n} l_{ii} \right)
    \]
  6. Construct connectivity matrix \( C = (c_{ij}) \), where
    \[
    c_{ij} = l(s_{ij} - J_{ij}) - l_{ij} \leq l_{ij} \leq 2l_{ij} \leq l_{ij}
    \]
  7. Assign data to clusters using connectivity matrix
- Pros and Cons
  - Pros: automatic, makes no assumption about shape of data
  - Con: complex

Zhang, et al. 7.43

Valley-Seeking: Examples

Zhang, et al. 7.44
Superparamagnetic Clustering (SPC)

- Hierarchical, divisive
- Data is modeled as a granular magnet, where each point is assigned a spin.
- The model is heated from low temperatures to high temperatures
  - Low temperatures = “ferromagnetic region”: all spins are aligned
  - High temperatures = “paramagnetic region”: system is disordered and all spins are random
  - “superparamagnetic region”: spins within the same high-density region are aligned while the spins of different high-density regions are not aligned. At these temperatures, the clusters are revealed.

SPC: Pros and Cons

- Pros
  - Automatic (don’t need to know the number of clusters/neurons a priori)
  - Makes no assumptions about shapes of clusters

- Cons
  - extremely complex
SPC: Algorithm

1. Calculate strength of interaction between each point and every other point
   \[ J_{ij} = \begin{cases} \frac{1}{2} \exp \left( -\frac{J_{ij}}{k_B T} \right) & \text{if } v_i \text{ and } v_j \text{ are neighbors} \\ 0 & \text{otherwise.} \end{cases} \]

2. Randomly assign each point a spin

3. Run \( N \) Monte Carlo simulations to determine whether each particle’s spin changes (points that are close together will change their states together)
   \[ p_{ij} = 1 - \exp \left( -\frac{J_{ij}}{k_T} \right) \]

4. Calculate the correlation \( G \) between each point and every other point
   \[ C_i = \langle q_i \rangle , \quad G_{ij} = \frac{(q - 1) C_i + 1}{q} \]

5. Assign points to the same cluster if their correlations are above a certain threshold

6. Repeat Steps 3-5 for each temperature

7. Find temperature corresponding to superparamagnetic region  Blatt, et al.

Online Clustering

- Developed for human closed-loop experiments
- Algorithm
  1. Initialization: Assign first data point to its own cluster.
  2. Calculate distance between next data point and each cluster centroid
  3. If smallest distance is less than a threshold, assign point to nearest cluster and recompute that cluster’s centroid. Otherwise, start a new cluster.
  4. Check the distances between each cluster and every other cluster. If any distance is below a threshold, merge those two clusters.
  5. Repeat 2-4 indefinitely.
    - Pros
      • Computationally simple
      • Realtime
      • Automatic
    - Con: Assumes spherical distribution of data
Commercial Spike-Sorting Software

Plexon
Dallas, TX

Processing 1 Hour (60 GB) of Data

• Software simulation
  – HPC with 10 blades
    • 3GHz dual-core Xeon CPU
    • 4 MB cache, 10 GB RAM
  – Total 40 CPUs!
    • Sequential processing

• Results
  – 70 minutes of run-time
  – $60,000 system cost
  – kW's of power

• Hardware emulation
  – 1 Xilinx ML506 Virtex-5 FPGA board
    • 300 MHz clock
  – Parallel hardware
    • Parallel processing

• Results
  – 0.36 seconds of run-time
  – $1,200 system cost
  – W's of power

Hardware processing:
0.5M times more cost-efficient
0.5B times more cost*power-efficient
“Green” Data-Center of the Future

Server-rack blade
FPGA array
CPU nodes
FPGA nodes
other nodes

Higher energy-efficiency
Faster data processing

References: Spike Sorting Algorithms

References: Spike-Sorting Hardware


