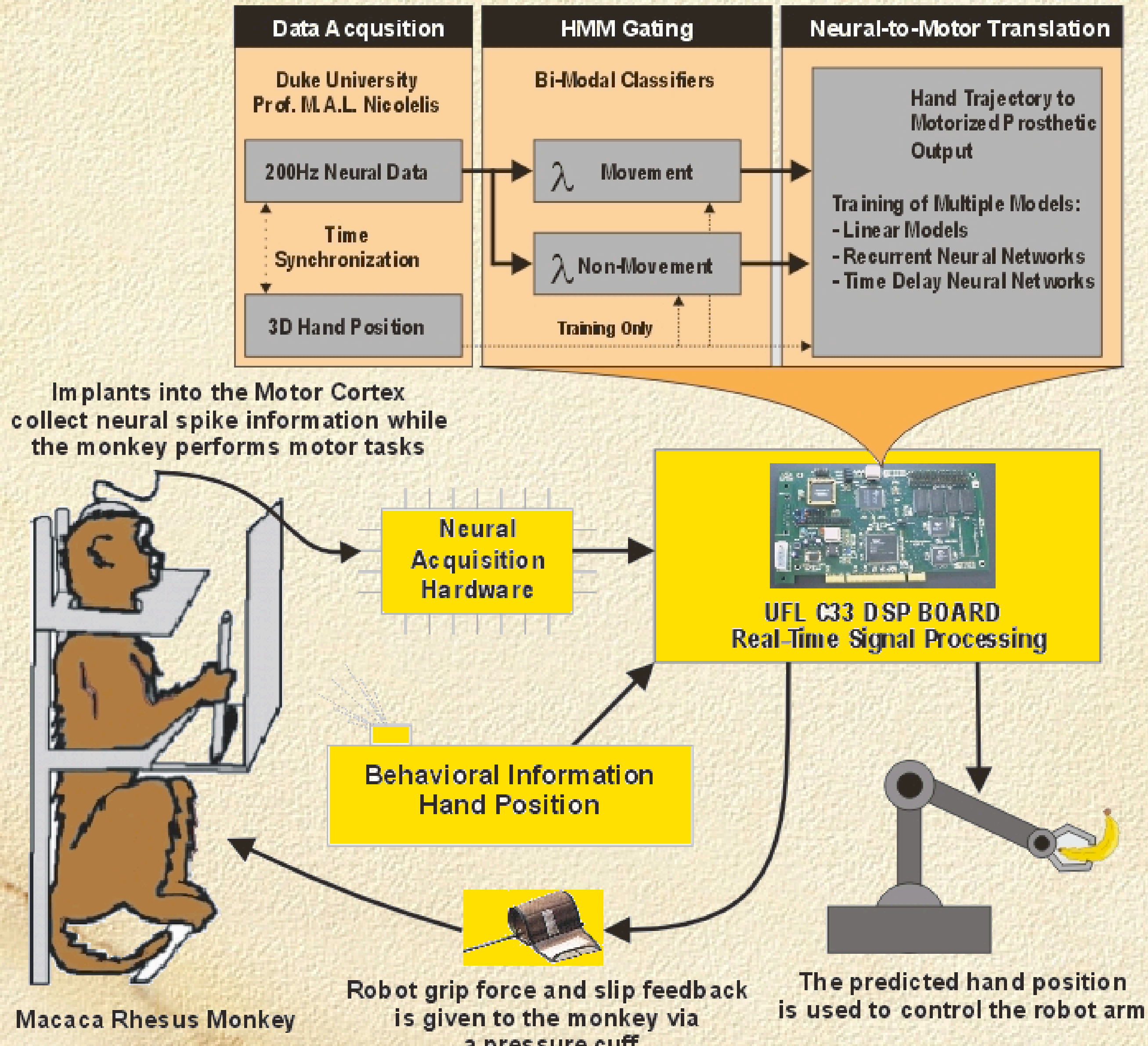
Bimodal Brain-Machine Interface for Motor Control of Robotic Prosthetic

Brain-Machine Interface: Project Overview

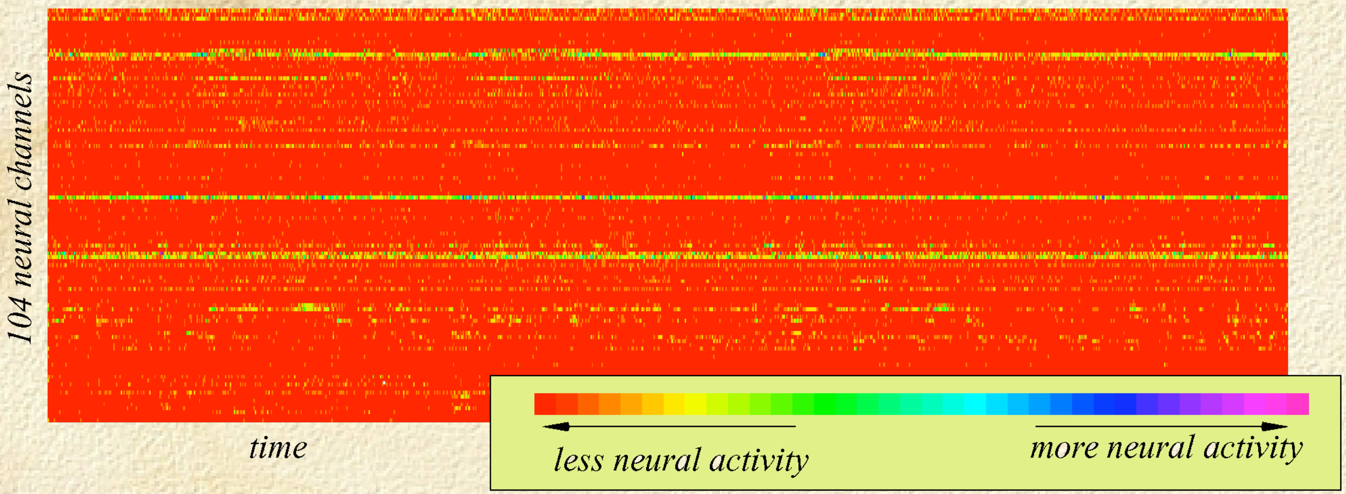
- Four-university effort (*Duke, UF, MIT, SUNY*)
- Funded by DARPA
- *Project goal*: develop direct brain-machine interfaces
- *Application*: intelligent prosthetic devices for handicapped people
- *UF's role*: develop mapping between motor-cortex neural activity and arm movements (in monkeys)

Brain-Machine Interface: System Diagram

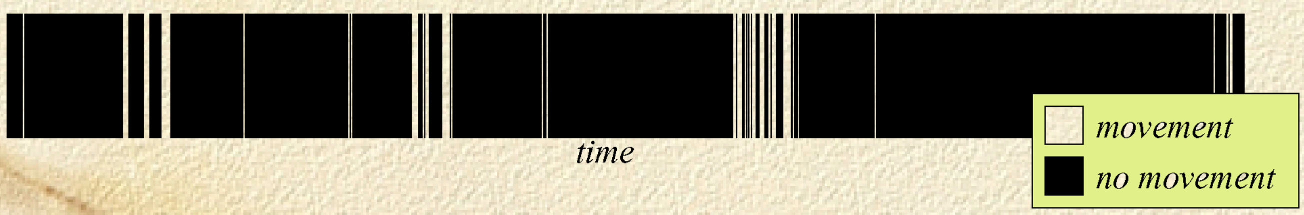
Super Monkey



Sample neural data



Input:104-channel neural spike data



Output: stationary/moving

Output: hand trajectory

Mapping neural activity: two approaches

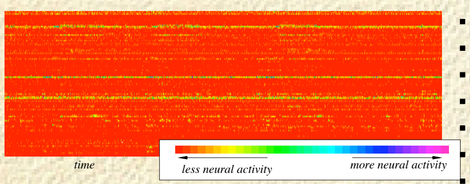
- *Global models*: FIR, recurrent neural networks, etc.
- *Multiple local models*: two-step approach
  - *Partition neural input space to motion primitives*
  - *Train model for each motion primitive partition*
  - *Benefit: reduce noise when arm is at rest*

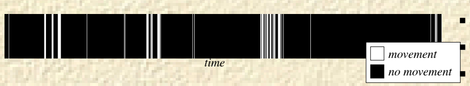
Initial approach

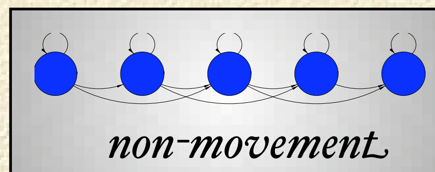
Multi-channel neural data

1st-level partitioning

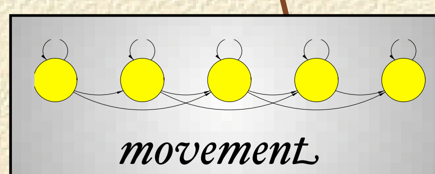
3d arm movement



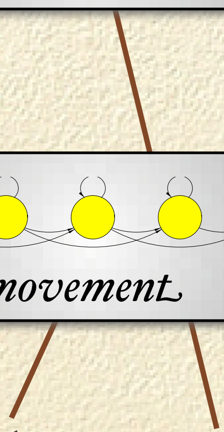




model #1



model #2



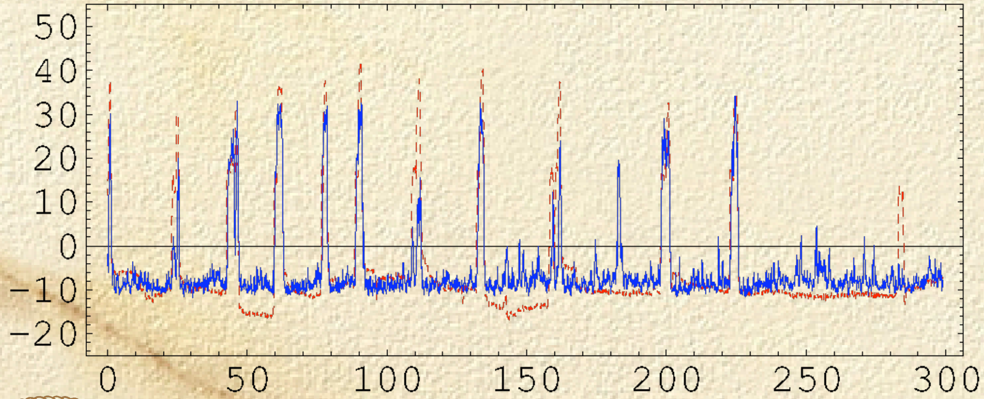
Hidden Markov Models (HMMs)

Bimodal system details

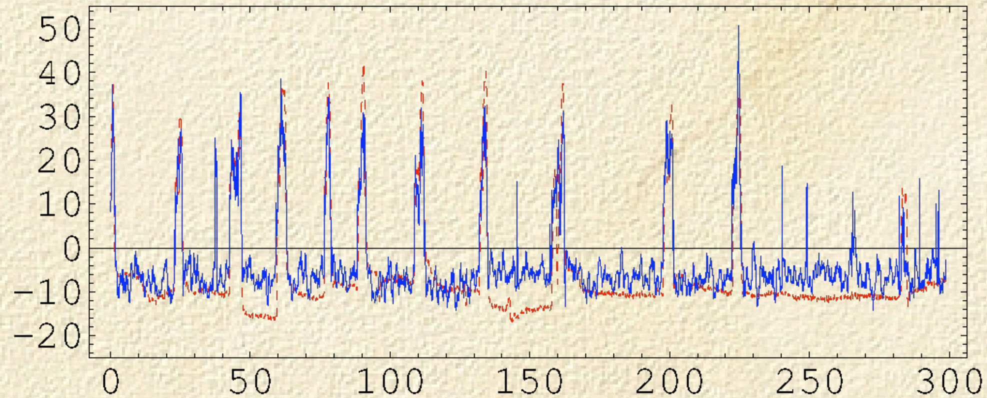
- *Two partitions*: movement vs. non-movement:
- Classification of data into two classes
  - VQ 104-channel neural spike data
  - Train two HMMs on quantized input data
- 3d-trajectory modeling
  - FIR filter for each class

Results & analysis

- 87%/90% correct classification on test data
- Tracking error of arm movement comparable to RNN (global model)
- Better end-point reaching than RNN



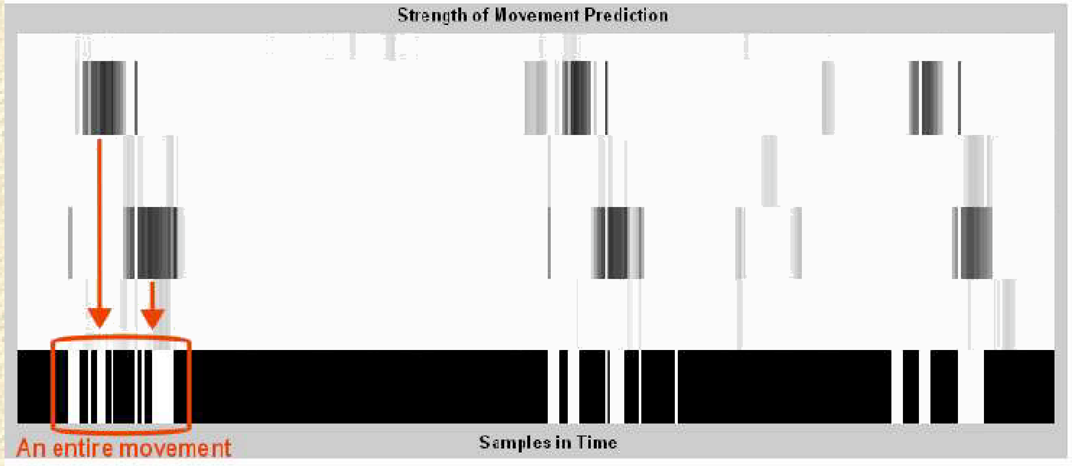
RNN



Bimodal system

Insights...

- Different neurons encode different things



- *VQ introduces substantial loss of information*

...therefore...

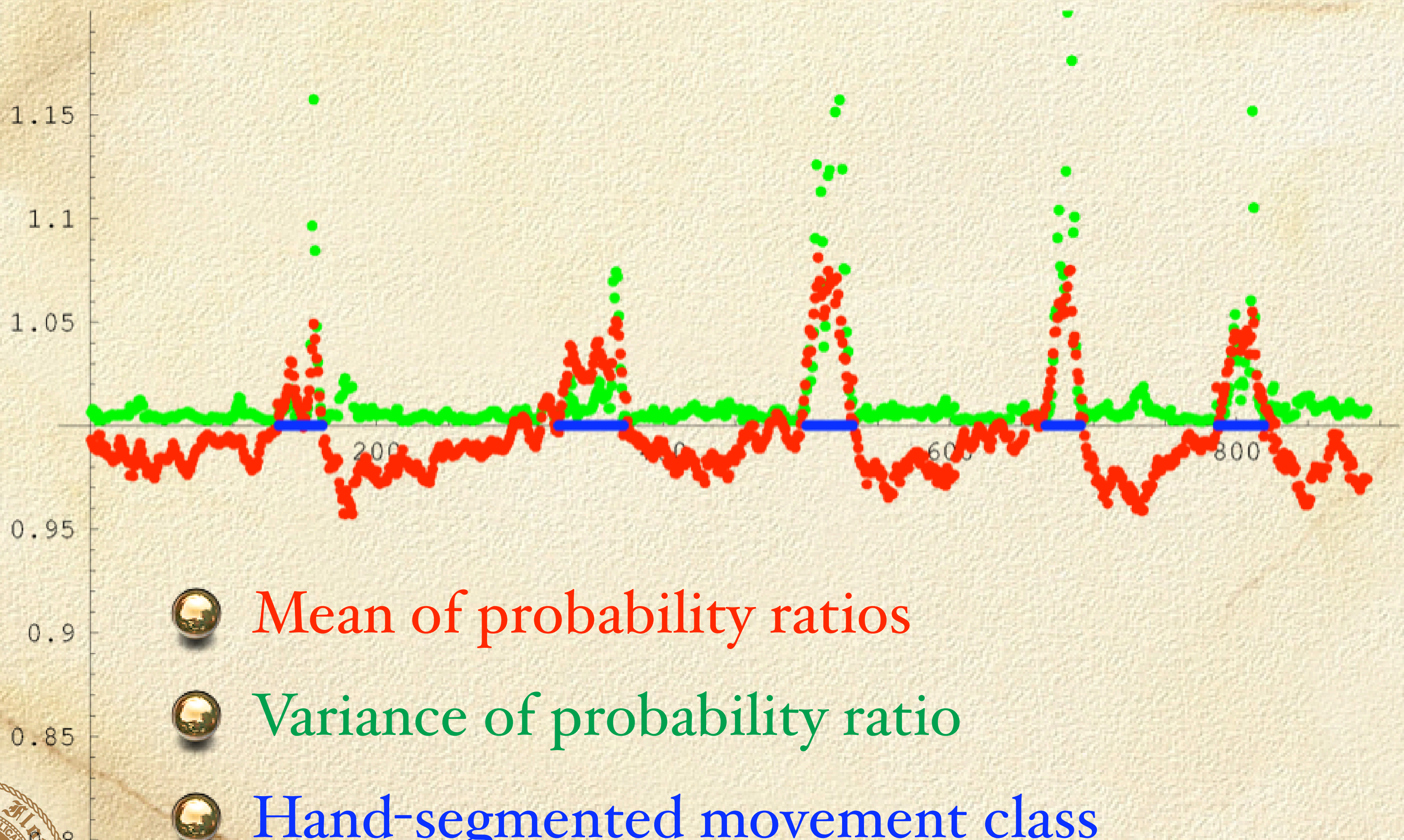
- Train HMMs on individual neurons (data is already discrete).
- Similar to mixture-of-experts approach.
- How to combine observation probabilities?

$$\sum \frac{P(O|\lambda_m)}{P(O|\lambda_s)}$$

$$\frac{\sum P(O|\lambda_m)}{\sum P(O|\lambda_s)}$$

$$\boxed{\sum \ln\left[\frac{P(O|\lambda_m)}{P(O|\lambda_s)}\right] = \ln\left[\prod \frac{P(O|\lambda_m)}{P(O|\lambda_s)}\right]}$$

Let's see how this works…



- Mean of probability ratios
- Variance of probability ratio
- Hand-segmented movement class

Results

- Improved classification performance from 87%/90% to 93%/93%.
- Biased classifier to equalize classification performance for two classes.