NEF Framework

- General methodology that allows builders of large scale, biologically plausible neural models of cognition.
  - ☆ neural compiler.
  - ☆ works for recurrent connections for complex dynamical systems.
  - ☆ local error driven learning
- Reasons for creating biologically accurate models.
  - ☆ evaluation of theories about the brain
  - ☆ match to behavioral data
  - ☆ suggests new types of algorithms.
  - ☆ Claims about algorithms that cannot be implemented on the human brain.
- NEF Representation
  - ☆ Distributed representations.
  - ☆ Distinction between the activity of a group of neurons and the value represented.
  - ☆ Value is a vector x.
- To map from x to activity a:
  - ☆ Every neuron i has an encoding vector e_i
  - ☆ Preferred direction vector, vector for strongest neuron firing
  - ☆ Ties with establishing tuning curves for neurons.
  - ☆ $a_i = G(\alpha_i e_i \cdot x + b_i)$ α_i is gain parameter, b_i is background bias, G is neural non-linearity.
  - ☆ G can be a variety of things, just must be mapping between input current and neuron activity
- Usually, each neuron responds to a distinct component of the input.
  - ☆ e_i values are perpendicular / basis vectors.
  - ☆ Varying e_i increases computational power

- Converting neural activity back into vector x
  - ★ Use linear decoder d_i, set of weights that maps activity back into estimate of x.
  - ★ $\hat{x} = \Sigma a_i d_i$
  - ★ finding d_i is a LS min problem, want to minimized difference between x and the esimate.
    $d = \Gamma^{-1} \Upsilon$ $\Gamma_{ij} = \Sigma_x a_i a_j$ $\Upsilon_j = \Sigma_x a_j x$.
- Decoder
  - ★ how accurately a group of neurons represents some value.
  - ★ High level interpretation of spiking activity.
  - ★ Can help solve for ideal neural connection weights
- Computation.
  - ★ Lets say connect A and B to compute $f(x) = x$
    - Need intermediate group of perfectly ideal linear neurons
    - Use vector d.
  - ★ Don't need intermediate, use weights. $\omega_{ij} = d_i \cdot e_j$
  - ★ Can find decoding weights to approximate any function $f(x)$
    - requires single layer of connections
    - More convoluted, lesser accuracy.
  - ★ $f(x)$ is approximated from linear sums of tuning curves.
  - ★ Incredibly fast to simulate
- Dynamics
  - ★ Integrator neural system
  - ★ $dx/dt = A(x) + B(u)$
  - ★ Use neurotransmitter time constant ϵ.
- Symbols
  - ★ Vector symbolic Architectures.
  - ★ Dimensions and operations encode relations