Sensory-motor computations
Outline

- Introduction (Day 1) – Sabes
- Sensory-motor transformations – Blohm
  - Population coding and parallel computing
  - Modelling sensory-motor mappings with artificial neural networks
- Tutorial: gain modulations for reference frames – Blohm

- Multi-sensory integration (Day 2) – Blohm
- Sensory-motor control – Sabes
  - Efference copy, forward models and Kalman filters
  - Prisms and inter-sensory adaptation
- Tutorial: Kalman filter and LQR – Sabes
Motor planning

- Hand-goal distance computation

Blohm et al. 2009
Sensory-motor transformations
Sensory-motor transformations

- Justin: DLR robot – ball catching
  - Sensory ref frames ~ motor ref frame…
  - Sensory code ~ motor code…
Sensory-motor transformations

- Reference frames
  - Determined by sensory and motor apparatus
    - Vision: attached to the retina
    - Audition: attached to the head
    - Proprioception: relative joint angles
    - Arm movement: relative to attachment at shoulder
Sensory-motor transformations

- Reference frames
  - Knowledge about reference frames is required to localize sensory and motor events
    - Same retinal image – different spatial locations
Sensory-motor transformations

- **Reference frames**
  - The relative orientation of different sensory and motor reference frames is not fixed
    - Changes with every movement
    - 6D changes (e.g. Eyes re. Shoulder)
    - Non-commutative

Tweed & Vilis, 1987

Blohm et al. 2009
Reference frames

A reference frame transformation is needed to map sensory to motor coordinates

Requires estimates of body geometry

Blohm et al. 2009
Reference frames

- A reference frame transformation is needed to map sensory to motor coordinates
  - Requires estimates of body geometry
  - Noise in estimates $\rightarrow$ stochastic reference frame transformations

$\mu_1, \sigma_1^2$ $\xrightarrow{\text{Ref. Frame Transformation}}$ $\mu_2, \sigma_2^2$

$\sigma_1^2 < \sigma_2^2$
Signal-dependent sensory noise

- Reach variability depends on gaze fixations
- Signal-dependent noise in muscle spindles explains arm position variability
- Neuronal noise is signal-dependent (Poisson)

![Graph showing relationship between spike-count mean and variance](Maimon & Assad (2009))

![Graph showing angle errors](Scott & Loeb (1994))

![Graph showing reach standard deviation](Blohm & Crawford (2007))
Examples: reference frame transformations

- Saccades

  - Shortest path
  - Listing's law

Klier & Crawford (1998)
Examples: reference frame transformations

- **Reaching / pointing**

  - **C** Head-unrestrained (45° oblique fixation)
  - **D** Head-roll (30° CCW)

  Blohm, & Crawford (2007)

  Leclercq et al. (in preparation)

  Blohm, Keith & Crawford (2009)
Examples: reference frame transformations

- Moving body / objects

Smooth pursuit

Blohm & Lefevre (2010)
Leclercq et al. (in preparation)
Examples: reference frame transformations

- Moving body / objects

  Manual tracking

  Leclercq et al. (in preparation)

  Leclercq et al. (2012)
Examples: reference frame transformations

- Visual-motor transformation deficits in optic ataxia

![Visual-motor transformation deficits in optic ataxia](image)

Khan et al. (2005a, b, 2009)
Examples: reference frame transformations

- Reference frame transformation deficits in optic ataxia

Khan, Pisella, Blohm (in revision)
Population coding and parallel computing
Direct vs. population coding

- **Vision:** population code
- **Movement:** digital / current command to actuators
Cosine tuning

- Tuning curves to wind direction for low-velocity interneurons of cricket cercal system
- Cosine tuning: \( \left( \frac{f(s)}{r_{\text{max}}} \right) = [\cos(s - s_a)]_+ \)
  - Firing rate \( f \)
  - Preferred direction \( s_a \)
- 4 neurons can encode all wind directions!

\[
\vec{v}_{\text{est}} = \sum_{a=1}^{4} \left( \frac{r}{r_{\text{max}}} \cdot \vec{c}_a \right)
\]

Dayan & Abbott, 2001

---

CoSMo 2012 - G. Blohm
Aug 6-7, 2012
Cosine tuning

Motor-related cosine tuning in PMv and M1

Kakei, Hoffman, Strick, 2003
Population codes

- Cricket wind direction: 4 neurons = population coding!
  - Principle: each neuron codes for a different set of stimulus values
  - Together, all neurons encode all possible stimuli as a population
  - Redundancy is always present! (counter-example: cricket)
  - Example: Gaussian receptive fields

Dayan & Abbott, 2001
Population codes

- Encoding a stimulus using population codes

Dayan & Abbott, 2001
Narrow versus wide tuning

- **Question:** what is better, narrow or wide tuning?
  - For fixed noise, within one layer: narrow is better!
  - In a neural network, output tuning curves should be wider than input tuning curves
    - Information in the output layer cannot be greater than in the input layer
    - Sharpening tuning curves in the output can only decrease (or at best preserve) information content
  - Result: the wide tuning of the input layer contains more information that the narrow tuning in the output layer
- **Consequence for the brain:** narrow-to-wide in processing hierarchy

Pouget & Sejnowski, 1997
Population codes

- Example: cue combination with population codes
  - Probabilistic population codes
    - Poisson-like neural noise
    - Variance inversely related to gains of population code

Ma et al. Nat Neuro 2006
Modelling sensory-motor mappings with artificial neural networks

ANN architecture and connectivity
Goals

- Feasibility of neural network implementation
  - Mostly trivial...
- More interesting questions
  - What is the optimal network structure given a fixed number of neurons / units?
  - What properties emerge from training?
  - Can these emerging properties explain aspects of real brain function / dysfunction?
  - Can we understand the difference between electrophysiological techniques (e.g. recording vs. stimulation)?
  - ...

Aug 6-7, 2012 CoSMo 2012 - G. Blohm
From spikes to firing rates

- **Approximations**
  - **Size**
    - One unit in a rate-based network represents average local population behaviour
    - One units’ behaviour mimics population computations
  - **Time**
    - Average firing rate does not capture spike dynamics, variability in spikes etc
    - Complexity of spike time interactions within a network lost
Feed-forward networks

- **Input**
  - E.g. sensory feature vector
  - Sampling

![Image of a feed-forward network diagram](Trappenberg 2010)
Feed-forward networks

- Perceptron

\[ r_i^{in} = x_i \quad \rightarrow \quad y = r_i^{out} \]

Example: \( y = w_1 \cdot x_1 + w_2 \cdot x_2 \)

General single-layer mapping (= simple perceptron)

\[ r_i^{out} = g \left( \sum_j w_{ij} r_j^{in} \right) \quad \Leftrightarrow \quad r^{out} = g(w r^{in}) \]
Feed-forward networks

- Perceptron

Example: \( g(x) = x \)

\[ y = w_1 \cdot x_1 + w_2 \cdot x_2 \]

\( w_1 = 1, \quad w_2 = -1 \)

\( \rightarrow \) Good generalization of network!
Feed-forward networks

- **Perceptron**

  ![Perceptron Diagram](image)

  Boolean function $g$: threshold node

  $$g(x) = \begin{cases} 
  1 & \text{if } x > \Theta \\ 
  0 & \text{elsewhere} 
  \end{cases}$$

  Linear separable function

  Not linear separable

  Trappenberg 2010

---

CoSMo 2012 - G. Blohm

Aug 6-7, 2012
Feed-forward networks

- **Multi-layer perceptron**
  - Universal function approximator
  - Given enough hidden nodes, any function can be approximated with arbitrary precision
  - Example: sine wave approx. with logistic transfer function

\[
f(x) = \frac{1}{1 + e^{-x}}
\]

Trappenberg 2010
Feed-forward networks

- Multi-layer perceptron
  - Generalization = performance outside the training set
  - Good interpolation abilities for sigmoid networks

![Graph showing overfitting and underfitting](Trappenberg 2010)
Feed-forward networks

- **Multi-layer perceptron**
  - **Limitations**
    - Brain-like performance does NOT mean the brain performs some mapping the same way
    - Training rules are non-physiologic (see next section)
  
- **Strengths**
  - Hidden layer activity might resemble brain function
    - given appropriate choices of input and output codings
  - The brain = a mapping network
  - Self-organization, analogous to the brain
  - High flexibility in possible computations
Neural transfer functions

\[ r_j^h = f(r_j^{in}, w_h) \]

\[ f : x \in S_1^{n^{in}} \rightarrow y \in S_2^{n^h} \]
Neural transfer functions

- Naka-Rushton function (1966)

\[
S(P) = \begin{cases} 
\frac{MP^N}{\sigma^N + P^N} & \text{for } P \geq 0 \\
0 & \text{for } P < 0
\end{cases}
\]

Visual neurons (LGN, V1, MT)
Response to stimuli with different contrasts
Neural transfer functions

- Idealized transfer functions in nodes
  - Satlin Transfer Function: $a = satlin(n)$
  - Satlins Transfer Function: $a = satlins(n)$
  - Hard-Limit Transfer Function: $a = hardlim(n)$
  - Linear Transfer Function: $a = purelin(n)$
  - Log-Sigmoid Transfer Function: $a = logsig(n)$
  - Tan-Sigmoid Transfer Function: $a = tansig(n)$

The Mathworks Inc
Modelling sensory-motor mappings with artificial neural networks

Training algorithms
Training algorithms

- **Gradient descent**
  
  **Cost function:**
  \[
  E = \frac{1}{2} \sum_i \left( r_{i}^{\text{out}} - y_i \right)^2
  \]
  
  (mean squared error)

  **Goal:** minimize the cost function
  
  - Change network weights
    
    \[
    w_{ij} \leftarrow w_{ij} + \Delta w_{ij}
    \]
    
    \[
    \Delta w_{ij} = -\varepsilon \cdot \frac{\partial E}{\partial w_{ij}}
    \]
Training algorithms

- Gradient descent

\[ \frac{\partial E}{\partial w_{ij}} = \frac{1}{2} \frac{\partial}{\partial w_{ij}} \left( \sum_{i} g \left( \sum_{j} w_{ij} r_{j}^{in} \right) - y_{i} \right)^{2} \]

- With chain rule

\[ \frac{\partial f}{\partial w_{ij}} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial h} \frac{\partial h}{\partial w_{ij}} \]

- Delta rule: \[ \Delta w_{ij} = \varepsilon \cdot g'(h_{i}) \cdot (y_{i} - r_{i}^{out}) r_{j}^{in} \]
Training algorithms

- Gradient descent
  - Linear perceptron: \( g(h_i) = h_i \)
    \[ g'(h_i) = 1 \]
  
- Perceptron learning rule: \( \Delta w_{ij} = \epsilon \cdot (y_i - r_i^{out}) r_j^{in} \)

  - Works also for most other transfer functions \( g \)
  - Similarity to Hebbian learning (supervised Hebbian):
    - Increase / decrease proportional to network error AND input strength
Training algorithms

- Generalization to multi-layer perceptrons
  - Back-propagation
    \[ r^{\text{out}} = g\left(w^{\text{out}} r^{h}\right) \]
    \[ r^{\cdot \text{out}}_i = g\left(\sum_j w^{\text{out}}_{ij} r^{h}_j\right) \]

- 3-layer perceptron:
  \[ r^{\text{out}} = g^{\text{out}}\left(w^{\text{out}} g^{h}\left(w^{h} r^{\text{in}}\right)\right) \]

Weights to adjust
Training algorithms

- Generalization to multi-layer perceptrons
  - Back-propagation
    - Generalized delta rule: output weights

\[
\frac{\partial E}{\partial w_{ij}^{out}} = \frac{1}{2} \frac{\partial}{\partial w_{ij}^{out}} \sum_i (r_{i,\text{out}} - y_i)^2
\]

\[
= \delta_{i,\text{out}}^{\prime} r_{j,\text{out}}^{h}
\]

with
\[
\delta_{i,\text{out}}^{\prime} = g^{\prime}_{\text{out}}(h_{i,\text{out}}^{h})(r_{i,\text{out}} - y_i)
\]
Training algorithms

- Generalization to multi-layer perceptrons
  - Back-propagation
    - Generalized delta rule: hidden layer weights

\[
\frac{\partial E}{\partial w_{ij}^h} = \frac{1}{2} \frac{\partial}{\partial w_{ij}^h} \sum_i \left( r_{i}^{\text{out}} - y_i \right)^2 \\
= \frac{1}{2} \frac{\partial}{\partial w_{ij}^h} \sum_i \left( g_{i}^{\text{out}} \left( \sum_j w_{ij}^{\text{out}} g_j^h \left( \sum_k w_{jk}^h r_{k}^{\text{in}} \right) \right) - y_i \right)^2 \\
= \delta_{i}^{h} \cdot r_{j}^{\text{in}} \\
\text{with } \delta_{i}^{h} = g^h \left( h_{i}^{\text{in}} \right) \sum_k w_{ik}^{\text{out}} \delta_{k}^{\text{out}}
\]

Back-propagation of error term!
Network design and analysis
Network design

- Stick to known physiology as much as possible
  - Input / output coding
  - Connectivity
  - Transfer functions
  - Learning rule?

- E.g. 3-D reach planning network

Blohm, Keith, Crawford, 2009; Blohm, 2012
Network analysis

- Receptive fields
  - = activation pattern of a neuron for targets across space

Blohm, Khan, Crawford, 2009 (adapted from Andersen, et al., 1985)
Gain modulation

- change of receptive field strength with secondary input
- E.g. eye position gain modulation of visual receptive fields in posterior parietal cortex

Blohm, Khan, Crawford, 2009 (adapted from Andersen, et al., 1985)
Gain modulation

- Reference frame transformations
  - Zipser & Andersen, Nature 1988

Eye position gain modulation of hidden layer units
Gain modulation

- Powerful computational means for
  - Cue combination
  - Reference frame transformations
  - Multi-sensory integration...

![Gain field theory diagram](image)

Blohm & Crawford, 2009
Reference frame transformations

- Reference frames based on “electrophysiological” analysis of a 3-D visuo-motor transformation network

Blohm, Keith, Crawford, 2009
Conclusion
Conclusion

- Feed-forward rate-based networks are the simplest form of ANNs
  - Computationally efficient
  - Powerful
  - But non-trivial mapping to biology?
- Learning algorithms
  - Fast and robust algorithms can be found
  - Mostly remote from biology
  - More complicated algorithms are more realistic
- FF-ANNs are very useful tools for investigating
  - Computations in the brain (reference frames, multi-sensory, …)
  - Hierarchical processing
  - Receptive fields
  - …
This afternoon...

- Implement back-propagation learning
- Analyze gain fields and receptive fields
Matlab tutorial: ANNs and gain fields
Exercise 1: Back-propagation

- Goal: program a simple feed-forward neural network and train it with back-propagation
  - Task: retinal-to-spatial transformation in 1-D
    - Spatial = retinal + eye orientation
  - 3-layer: input, hidden layer, output
    - Input: 1-D retinal map
    - Transfer functions: sigmoid
    - Output: 1-D spatial map
  - Training method: error back-propagation
    - Generate random training set
Exercise 1: Back-propagation

\( w_{ij} \leftarrow w_{ij} + \Delta w_{ij} \)

\( \Delta w_{ij} = -\varepsilon \cdot \frac{\partial E}{\partial w_{ij}} \)

\( \Delta w_{ij} = \varepsilon \cdot (y_i - r^{out}_i) r^{'in}_j \)

\[
\frac{\partial E}{\partial w_{ij}^{out}} = \frac{1}{2} \frac{\partial}{\partial w_{ij}^{out}} \sum_i (r_i^{out} - y_i)^2 = \delta_i^{out} r_j^{h}
\]

with \( \delta_i^{out} = g^{out}(h_i^{h})(r_i^{out} - y_i) \)

\[
\frac{\partial E}{\partial w_{ij}^{h}} = \frac{1}{2} \frac{\partial}{\partial w_{ij}^{h}} \sum_i (r_i^{out} - y_i)^2 = \frac{1}{2} \frac{\partial}{\partial w_{ij}^{h}} \sum_i \left( g^{out} \left( \sum_j w_{ij}^{out} g^{h} \left( \sum_k w_{jk}^{h} r^{'in}_k \right) \right) - y_i \right)^2 = \delta_i^{h} r_j^{in}
\]

with \( \delta_i^{h} = g^{h}(h_i^{in}) \sum_k w_{ik}^{out} \delta_k^{out} \)
Exercise 2: RF & gain field analysis

- Use Matlab neural network toolbox
  - Code provided
- Train a network
  - (just run the code)
  - You can use different versions of back-propagation (default: resilient back-prop)
- Plot RFs for individual hidden layer and output layer units
- How do these RFs change with eye orientation?
  - Gain fields versus RF shifts...