So far...

- **Units**: ions, conductance, membrane potential, firing.
- **Networks**: transformations, amplifications, attractors, basic building blocks of cognition.

How do networks ever come to do interesting things?

- **Learning**
  - Learning rules: How to adjust weights based only on local information (presynaptic and postsynaptic activity) to produce appropriate network behavior.
  - Two main types:
    - Learning internal model (statistics) of environment (Ch 4)
    - Learning to solve a task (produce output from input) (Ch 5)
    - Doing both at the same time (Ch 6)

What is Model Learning?

- "Things" in the world have relatively stable sets of features.
- How do detectors in our brains come to detect these things?
- The features of a particular thing tend to appear together and disappear together.
- A thing is nothing more than a correlated cluster of features.
- Learning mechanisms sensitive to correlation will lead to representation of useful things.

Pick up on correlations in the world.

Learning mechanisms sensitive to correlation will lead to representation of useful things that are not co-activated.

Biology: Associative/Hebbian Learning

- Learning mechanisms sensitive to correlation will lead to representation of useful things.
- A thing is nothing more than a correlated cluster of features.
- Two main types:
  - Learning internal model (statistics) of environment (Ch 4)
  - Learning to solve a task (produce output from input) (Ch 5)
  - Doing both at the same time (Ch 6)

Biology suggests associative or "Hebbian" learning:

- Proposed by Donald Hebb: "Units that fire together wire together!"
- "Units that fire out of sync, lose their link."
- Synaptic efficacy (weights) change when neurons are excited.

LTP is dependent on NMDA receptor activation.

APV = NMDA antagonist

**LTP**

Pick up on correlated features in the world.

Model Learning

Pick up on correlated features in the world.

Learning mechanisms sensitive to correlation will lead to representation of useful things that are not co-activated.

So far...

- **Learning**
  - How do networks ever come to do interesting things?
  - Building blocks of cognition.
  - How networks instantiate problem-solving, attention, etc.
  - Units/Cells: correlation, membrane potential, firing.

Learning

Learning mechanisms sensitive to correlation will lead to representation of useful things that are not co-activated.
And so is LTD

APV = NMDA antagonist

NMDA = Associativity: Both Pre & Post Active

Axon

Presynaptic

Dendrite

complex

chem processes

Postsynaptic

NMDA

Ca++

1. Mg+ unblocks NMDA channels as postsynaptic V_m increases

2. Glutamate released with spike binds to and opens NMDA channels

3. Ca++ enters through NMDA channels

Chem processes → new AMPA receptors (or trafficking of existing ones to membrane)

AMPAR's open Na channels → excitation

NMDAR's: learning via Ca.

Biology: NMDA-mediated LTP/LTD

Strong activity (Ca++) = LTP weak = LTD

For details of mechanisms of LTP/LTD:

THE NEUROBIOLOGY OF LEARNING AND MEMORY

Jerry W Rudy 2008

Very clear and well-written

LTP -> LTD

Studying the movements -- research

Learn processing -- how did we process or store data? Is memory preserved?

NMBA = NMDA antagonist

And so is LTD
Linear Activation:

\[ y_j = \sum_i x_i w_{ij} \quad (1) \]

Simple Hebb rule:

\[ \Delta w_{ij} = \epsilon x_i y_j \quad (2) \]

Weights get stronger for two units that are correlated, but not for uncorrelated units.

Technically: weights evolve toward principal eigenvector of correlation matrix among inputs.

\[ \Delta w_{ij} + (1)^\eta w_{ij} = (1 + \eta)^\eta w_{ij} \]

Note that the receiving unit is active, given the sending unit is active.

Weight converges on the probability that the sending unit is active, given that the receiving unit is active.

If \( a_j \) is active, learned weight reflects the likelihood that \( a_i \) caused that activation.

Allows units to learn about correlations among input patterns that activate it, but not among those that don't activate it.

Problem: Weights will grow infinitely large.

→ Normalize by subtracting off weight.

\[ \Delta w_{ij} = \epsilon a_j (a_i - \mu w_{ij}) \]

→ Weight converges on the probability that the sending unit is active, given that the receiving unit is active.

\[ \epsilon w_{ij} + (1)^\eta w_{ij} = (1 + \eta)^\eta w_{ij} \]

\[ \mu = \text{learning rate} \]

\[ a_i = \text{act of sending unit} \]

\[ a_j = \text{act of receiving unit} \]

Problem:

Weights will grow infinitely large.

→ Normalize by subtracting off weight.

\[ \Delta w_{ij} = \epsilon a_j (a_i - \mu w_{ij}) \]

→ Weight converges on the probability that the sending unit is active, given that the receiving unit is active.

\[ \epsilon w_{ij} + (1)^\eta w_{ij} = (1 + \eta)^\eta w_{ij} \]

Learned weight reflects the likelihood that \( a_i \) caused that activation.

Allows units to learn about correlations among input patterns that activate it, but not among those that don't activate it.
\[ \Delta w_{ij} = \epsilon a_j (a_i - w_{ij}) \Theta - \Theta + LTD \]

When both \(a_j\) and \(a_i\) are large: Lots of \(\text{Ca}^{++}\) → LTP

When \(a_j\) is large but \(a_i\) is small: Some \(\text{Ca}^{++}\) → LTD

When \(a_j\) is small: Mg+ blocking NMDA channels → nothing.

---

**Model learning**

Pick up on correlations in the world.

**Positive Correlation**

(whether for pixels in visual images, emotions and people, etc.)

Based on Hebbian (LTP/LTD) mechanisms.

---

**What does Hebbian Learning Do?**

- Hebbian learning tunes units to represent correlated sets of input features.
- If a unit has 100 inputs, turning on and off a single input feature won't have a big effect on the unit's activity.
- In contrast, turning on and off a large cluster of 900 input features will have a big effect on the unit's activity.
Hebbian Learning

The figure shows a neural network with excitatory and inhibitory connections. The network learns by adjusting the weights of the connections based on the activity of the neurons. The excitatory connections are represented by the black circles, while the inhibitory connections are represented by the white circles. The arrows indicate the direction of the connections.

Because smaller clusters of inputs do not reliably activate the network, the network learns more about big (vs. small) clusters.

Big clusters of inputs reliably activate the network, and the network learns more about big (vs. small) clusters.

Because input activity is not activated in the learning process.
What does Hebbian Learning Do?

• Hebbian learning finds the thing in the world that most reliably activates the unit and tunes the unit to like that thing even more!
What does Hebbian Learning Do?

- Hebbian learning finds the thing in the world that most reliably activates the unit and tunes the unit to like that thing even more.
- The thing in the world that most reliably activates the unit = principal component function of how well an input activates the unit, and how frequently it is presented.
Integrating over Experiences

• What a unit learns to represent is a function of how excitable the unit is.

• Units that are activated (initially) by a wide range of stimuli end up representing an average of all of these stimuli.

– Imagine a unit that is activated by every stimulus; in this case, it learns a little about every one of these stimuli.

– If all of the input patterns have something in common, then the unit will learn to have different units in the network learn to represent those commonalities.

• There are very few meaningful things that are present in all of the input patterns, if you try to average over too large a set of stimuli then you get mush.

• Thus the learning rule allows you to represent concepts at varying levels of abstraction, depending on how excitable the unit is.

Multiple Units

• One detector can only represent one "thing" (pattern of correlated features).

• Goal: We want to have different units in the network learn to represent different things.

• Random initial weights and inhibitory competition are important for achieving this goal.

• What happens when different units have the same initial weights and no competition...
Solution: Self-Organized Learning

1. Random initial weights
2. kWTA inhibition = only strongest units active.
3. Hebbian learning = winners get stronger (losers don't do anything; can win on something else).
4. Goto 1

Result: different units tuned for different input features. → "Self-Organized Learning"
Choose one of the things in this world: love, life, and sex. Each layer is composed of a random combination of two layers.
Summary: Model Learning

The World

The Learner

Model Parameters

World State Projection

Visible State Model StateInverse

Get a lot of poor quality information.

Need biases to augment, structure this info (e.g., parsimony).

One good bias is focus on correlations (causality).

Built in biases? Biological/genetic: associative LTP, different cortical layers, connectivity...

but not necessarily specific representations or knowledge!

→ Bias to learn! (but no bias to know language, etc)

EXTRA: Fine-Tuning Hebbian Learning

The remaining stuff in chapter 4 describes various methods of fine-tuning the Hebbian learning rule to do better under different conditions. This is not critical for understanding the main principles, which remain essentially unaltered.

Corrections

• CPCA weights don't have much dynamic range of selectivity.

• Solution: renormalizing weights and contrast enhancement.

• Quantitative adjustments – retain qualitative features of CPCA motivated by biology.

Renormalization

Keep a weight of .5 for uncorrelated inputs, even with sparse activity (otherwise, uncorrelated input weights go to $\alpha$).

($\alpha$ = expected average activity level of sending layer)

($\alpha$ = \alpha) (expected average activity level of receiving layer)

何度も元のあるものに到るウィンドウ経由エネルギー

Want low weights ($< .5$) to reflect negative correlation, i.e., when $y_j$ is active, $x_i$ is more than likely to not be active.

Then set maximum to 0.5 instead of 1:

$$\Delta w_{ij} = \epsilon [y_j x_i - y_j w_{ij}]$$

Then set maximum to $m$ instead of 1:

$$\Delta w_{ij} = \epsilon [y_j x_i (m - w_{ij}) + y_j (1 - x_i)(0 - w_{ij})]$$

Then set $m$ to compensate for sparse activity:

$$m = 0.5\alpha$$

Corrections

Parameter $savg_{cor}$ controls how much of this sending average (savg) correction (cor) happens. 1 = full, 0 = none. .4 = default.

$\Delta w_{ij}$ instead of $w_{ij}$:

$$\Delta w_{ij} = \epsilon [y_j x_i (m - w_{ij}) + y_j (1 - x_i)(0 - w_{ij})]$$

Then set maximum to $m$ instead of 1:

$$\Delta w_{ij} = \epsilon [y_j x_i (m - w_{ij}) + y_j (1 - x_i)(0 - w_{ij})]$$

Set $m$ to compensate for sparse activity

Quantitative adjustments – retain qualitative features of CPCA

Solution: renormalizing weights and contrast enhancement

CPCA weights don’t have much dynamic range or selectivity.

Extra: Fine-Tuning Hebbian Learning

"One good bias is focus on correlations (causality)."

Need biases to augment structure this info (e.g., parsimony).

Can get a lot of poor quality information..."
Introduce a parameter $q_m(\text{savg\_cor})$ to allow continuum (not all or none):

$$\alpha_m = 0.5 - q_m(0.5 - \alpha) \quad (6)$$

Cleaning up Messy Receptive Fields

- The "gang" project ends up with a big weight to the first two (correlated) features, and a moderate weight to the third feature
- So, there is a sense in which the unit is selective for the first two features, but it is only weakly selective
- How can we enhance selectivity?

Contrast Enhancement

- Between strongest and weaker correlations, via sigmoidal function:

  \[
  \begin{array}{cccccc}
  \text{Linear Weight Value} & 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 1.0 \\
  \text{Effective Weight Value} & 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 1.0 \\
  \end{array}
  \]

  - Contrast Enhancement
  - Slope (sharpness of contrast) via gain $\gamma$ ($\text{wt\_sig\_gain}$)
  - Offset (where midpoint is) via $\theta$ ($\text{wt\_sig\_off}$)

Effects of wt contrast enhancement

2. Effects of wt contrast enhancement

\[(\prescript{\ast}{0} - 5^{-})^{\text{wt\_sig\_off}} - 5^{-} = \prescript{\ast}{0}\]

Introduction to parameter $b_m(savg\_cor)$ to allow continuum (not all)
Sequential (Standard) PCA (SPCA)

First compute correlations across all inputs for first unit
Then do same for second unit, but keep it orthogonal to 1st, etc.

As applied to natural visual scenes:
1st is blob, 2nd is 1/2 blob, etc: Average over all inputs = blob!

Problem: assumes world is hierarchy, but it isn’t!

Conditional PCA (CPCA)

Compute correlations conditional on only subset of inputs (i.e., where particular features are present).

CPCA of natural visual scenes:
- Different units encode different things by figuring out different subsets of inputs
- CPCA operators over different subsets encode different things

Comparison

SPCA operates over all inputs, ensures different units encode different things by making them orthogonal.

CPCA operates over subsets of inputs, ensures different units encode different things by giving them different subsets of inputs.

KVLTA competition ensures that different units are active for different inputs

CPCA operators over different subsets of inputs, ensures different units encode different things by figuring out different subsets of inputs
Weight moves towards $x$ conditional on $y$.

→ learns principal components conditional on whether unit $y$ is active... principal component that activates that unit.

Achieves conditional probability goal:

$$w_{ij} = P(x_i = 1 | y_j = 1) = P(x_i | y_j)$$

Weight = prob. that the sender $x_i$ is active given that receiver $y_j$ is active.

CPCA equations

$$\Delta w_{ij} = 0_{A_{ij} + B_{ij}}$$

$$(t + 1) = 0_{A_{ij} + B_{ij}}$$

$$w_{ij} = P(x_i = 1 | y_j = 1) = P(x_i | y_j)$$

Weight moves towards $x_i$, conditional on $y_j$.

CPCA operators over different subsets of inputs, encodes different things.