So far...
So far...

- **Units**: ions, conductance, membrane potential, firing.
So far...

- **Units**: ions, conductance, membrane potential, firing.

- **Networks**: transformations, amplifications, attractors, basic building blocks of cognition.
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

• Describe an “ideal” learning system. (If you could design one, what features would it have?)
Learning

● Describe an “ideal” learning system. (If you could design one, what features would it have?)

● Describe the human learning system. (How do our models need to learn to capture how we learn?)
Learning

- Tuning detectors locally to achieve global results.
Learning

- Tuning detectors locally to achieve global results.

- Two main types (require different learning mechanisms):
  - Learning internal **model** of environment (ch 4).
Learning

- Tuning detectors locally to achieve global results.

- Two main types (require different learning mechanisms):
  - Learning internal **model** of environment (ch 4).
  - Learning to solve a **task** (produce output from input) (ch 5)
• Tuning detectors locally to achieve global results.

• Two main types (require different learning mechanisms):
  – Learning internal **model** of environment (ch 4).
  – Learning to solve a **task** (produce output from input) (ch 5)
  – Doing both at the same time (ch 6).
Model learning

Pick up on correlations in the world.
Model learning

Pick up on correlations in the world.

... t t+1 t+2 ...

Positive Correlation
Model learning

Pick up on correlations in the world.

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

Pick up on correlations in the world.

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

Simulation.
Synaptic efficacy (weights) change when neurons are excited:

Going up: Long Term Potentiation (LTP)

Going down: Long Term Depression (LTD).
NMDA = Associativity: Both Pre & Post Active

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
Biology: NMDA-mediated LTP/D

Strong activity ($Ca^{++}$) = LTP, weak = LTD
Hebbian Learning Picks up on Correlations
Hebbian Learning Picks up on Correlations

Simple Hebb rule: \( \Delta w_{ij} = \epsilon a_i a_j \)
Hebbian Learning Picks up on Correlations

Simple Hebb rule: $\Delta w_{ij} = \epsilon a_i a_j$

Linear Activation: $a_j = \sum_i a_i w_{ij}$
Hebbian Learning Picks up on Correlations

Simple Hebb rule: $\Delta w_{ij} = \epsilon a_i a_j$

Linear Activation: $a_j = \sum_i a_i w_{ij}$
Hebbian Learning Picks up on Correlations

Simple Hebb rule: \( \Delta w_{ij} = \epsilon a_i a_j \)

Linear Activation: \( a_j = \sum_i a_i w_{ij} \)

Weights get stronger for the two units that are correlated, don’t get stronger for uncorrelated unit!
Problem: Infinite Weights

Simple Hebb rule: \( \Delta w_{ij} = \epsilon a_i a_j \)
Problem: Infinite Weights

Simple Hebb rule: $\Delta w_{ij} = \epsilon a_i a_j$

Weights will grow infinitely large.
Problem: Infinite Weights

Simple Hebb rule: \( \Delta w_{ij} = \epsilon a_i a_j \)

Weights will grow infinitely large.

Normalize by subtracting off weight.

\[
\Delta w_{ij} = \epsilon a_j (a_i - w_{ij})
\]  \(\text{(1)}\)
\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) are large:
\[ \Delta w_{ij} = \varepsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) large:
LTP, lots of \( Ca^{++} \)
\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) large:
LTP, lots of \( Ca^{++} \)

When \( y_j \) large but \( x_i \) small:
\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) large:
LTP, lots of \( Ca^{++} \)

When \( y_j \) large but \( x_i \) small:
LTD, some \( Ca^{++} \)
\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) large:
LTP, lots of \( Ca^{++} \)

When \( y_j \) large but \( x_i \) small:
LTD, some \( Ca^{++} \)

When \( y_j \) small:
\[ \Delta w_{ij} = \varepsilon y_j (x_i - w_{ij}) \]

When both \( y_j \) and \( x_i \) large:
LTP, lots of \( Ca^{++} \)

When \( y_j \) large but \( x_i \) small:
LTD, some \( Ca^{++} \)

When \( y_j \) small:
Nothing, Mg+ blocking NMDA channels.
Model learning

Pick up on correlations in the world.

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

Based on Hebbian (LTP/LTD) mechanisms.
Multiple Units

What happens when you add more receiving units?
Multiple Units

What happens when you add more receiving units?

They will all represent the same correlations!
Multiple Units

What happens when you add more receiving units?

They will all represent the same correlations!

Soln: kWTA = units compete to learn
learning causes specialization

= Darwinian natural selection: survival (& specialization) of the fittest!
1. kWTA inhibition = only strongest units active.
Self-Organized Learning

1. kWTA inhibition = only strongest units active.

2. Hebbian learning = winners get stronger
   (losers don’t do anything; can win on something else).
Self-Organized Learning

1. kWTA inhibition = only strongest units active.

2. Hebbian learning = winners get stronger
   (losers don’t do anything; can win on something else).

3. Goto 1
Self-Organized Learning

1. kWTA inhibition = only strongest units active.

2. Hebbian learning = winners get stronger
   (losers don’t do anything; can win on something else).

3. Goto 1

Result: different units tuned for different input features.
Summary: Model Learning
Get a lot of poor quality information.
Get a lot of poor quality information.

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

The remaining stuff is technical mumbo-jumbo that was unfortunately included in the book but will be removed/condensed in the second edition.
Sequential (Standard) PCA (SPCA)

First compute correlations across all inputs for first unit
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.
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).
Conditional PCA (CPCA)

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

CPCA of natural visual scenes:

World is a *heterarchy* – no uber-components, just lots of features!
Comparison

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

kWTA competition ensures that different units are active for different inputs.
\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]  

Weight moves towards \( x_i \), conditional on \( y_j \).
CPCA Equations

\[ \Delta w_{ij} = \epsilon y_j (x_i - w_{ij}) \]  \hspace{1cm} (2)

Weight moves towards \( x_i \), conditional on \( y_j \).

Achieves conditional probability goal:

\[ w_{ij} = P(x_i = 1 | y_j = 1) = P(x_i | y_j) \]  \hspace{1cm} (3)

Weight = prob. that the sender \( x_i \) is active given that receiver \( y_j \) is.
Corrections

- CPCA weights don’t have much *dynamic range* or *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 $\alpha$).
Renormalization

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

Parameter $s_{avg\_cor}$ controls how much of this sending average (savg) correction (cor) happens. 1 = full, 0 = none. .4 = default.
Contrast Enhancement

Between strongest and weaker correlations, via sigmoidal function:
Contrast Enhancement

Between strongest and weaker correlations, via sigmoidal function:

Slope (sharpness of contrast) via $\gamma (wt\_sig\_gain)$
Between strongest and weaker correlations, via sigmoidal function:

Slope (sharpness of contrast) via $gain \gamma (wt\_sig.gain)$
Offset (where midpoint is) via $\theta (wt\_sig.off)$