Learning frameworks

Supervised learning
- Assumes environment specifies correct output (targets) for each input

Unsupervised learning
- Assumes environment only provides input; learning is based on capturing the statistical structure of that input (efficient coding)

Reinforcement learning
- Assumes environment provides evaluative feedback on actions (how good or bad was the outcome) but not what the correct/best action would have been

Efficient coding: Principal Components Analysis (PCA)
Recode high-dimensional data into smaller number of orthogonal dimensions that capture as much variance (information) as possible

Self-supervised learning: (Auto)encoder networks
- Network must copy inputs to outputs through a “bottleneck” (fewer hidden units)
- Hidden representations become a learned compressed code of the inputs/outputs
  - Capture systematic structure among full set of patterns
  - Due to bottleneck, don’t have capacity to overlearn idiosyncratic aspects of particular patterns
- For N linear hidden units, hidden representations span the same subspace as the first N principal components (≈ PCA)

Autoencoder can approximate a recurrent network
Patterns can be multiple groups coding different types of information
- Can present all or only some of the information as input, and require network to generate all of the information as output [supervised]

Social attachment learning
(Thrush & Plaut 2008)
Self-supervised learning: Prediction

Simple recurrent (sequential) networks

- Target output can be prediction of next input

Boltzmann Machine learning: Unsupervised / generative version

Supervised version

- Negative phase: Clamp inputs only; run hidden and outputs units (cf. forward pass)
- Positive phase: Clamp both inputs and outputs (targets); run hidden (cf. backward pass)

Unsupervised / generative version

- Positive phase: Visible units clamped to external input
- Negative phase: Network “free-runs” (nothing clamped)
- Network learns to make its free-running behavior look like its behavior when receiving input (i.e., learns to generate input patterns)

Objective function (unsupervised)

\[ G = \sum_{\alpha} p^+(V_{\alpha}) \log \frac{p^+(V_{\alpha})}{\tilde{p}^-(V_{\alpha})} \]

\[ [G = \sum_{\alpha,\beta} p^+(I_{\alpha}, O_{\beta}) \log \frac{p^+(O_{\beta}|I_{\alpha})}{\tilde{p}^-(O_{\beta}|I_{\alpha})}] \]

Restricted Boltzmann Machines

- No connections among units within a layer; allows fast settling
- Fast/efficient learning procedure
- Can be stacked; successive hidden layers can be learned incrementally (starting closest to the input) (Hinton)

Hinton's handwritten digit generator/recognizer

- Multilayer generative model trained on handwritten digits (generates image and label)
- Final recognition performance fine-tuned with back-propagation
Competitive learning

- Units in a layer are organized into non-overlapping cluster of competing units.
- Each unit has a fixed amount of total weight to distribute among its input lines (usually $\sum_i w_{ij} = 1$).
- All units in a cluster receive the same input pattern.
- The most active unit in a cluster shifts weight from inactive to active input lines:
  \[ \Delta w_{ij} = \epsilon \left( \frac{a_i}{\sum_k a_k} - w_{ij} \right) \]
- Units gradually come to respond to clusters of similar inputs.

Competitive learning: Recovering “lost” units

**Problem:** poorly initialized units (far from any input) will never win competition and so will never adapt.

**Solution:** Adapt losers as well (but with much smaller learning rate); all units eventually drift towards input patterns and start to win.

Self-Organizing Maps (SOMs)/Kohonen networks

- Extension of competitive learning in which competing units are topographically organized (usually 2D).
- Neighbors of “winner” also update their weights (usually to a lesser extent), and thereby become more likely to respond to similar inputs.
- Input space similarity gets mapped onto (2D) topographic unit space.
Lexical representations