Unsupervised Learning 3:

More methods

Emanuel Todorov
Cognitive Science Department
University of California San Diego

Content-addressable memory (or pattern completion)

**Hopfield network**: recurrent network with symmetric weights and binary units

\[ w_{ij} = w_{ji}; \quad o_i = \text{sign} \left( \sum_j w_{ij} o_j \right) \in \{+1,-1\} \]

Think of each pixel in an image as being a variable, and each image as being a data vector. Given a dataset of images, we compute the weights (to be explained how)

Now, if we are given the values of some of the pixel, we can reconstruct the remaining pixels. Or, if we are given a noisy image, we can clean it up.

Hopfield networks are guaranteed to settle! This is because the updates

\[ o_i^{\text{new}} = \text{sign} \left( \sum_j w_{ij} o_j^{\text{old}} \right) \]

always decrease a non-negative “energy” function, also known as a Lyapunov function.
Computing the weights for a Hopfield network

We want to make sure that the input vectors correspond to attractors of the network, so that if we provide a valid input it remains unchanged.

Update rule: \( o_i = \text{sign} \left( \sum_j w_{ij} o_j \right) \)

Vector notation: \( o = \text{sign} \left( W o \right) \)

Suppose we can find a \( W \) for which \( W o = a o \) where \( a \) is a positive scalar.

Then we are done, because \( o = \text{sign} \left( a o \right) \)

Good choice: \( W = o o^T \) This works because: \( W o = o o^T o = a o \)

where \( a = \| o \|^2 \)

Now consider a dataset of \( N \) input vectors: \( \{ o_1, \ldots, o_N \} \)

We want the network to preserve all of them!

Use covariance matrix: \( W = \frac{1}{N} \sum_i o_i o_i^T \)

\( W o_k = \frac{1}{N} \sum_i o_i (o_i^T o_k) = a o_k \)

\( \text{... but does it work?} \)

if all inputs are mutually orthogonal:

\( o_i^T o_k = 0 \) for \( i \neq k \)

Hopfield nets can “remember” no more patterns than the number of units they have. This is because the max number of mutually orthogonal vectors in N-dim space is N.

Boltzmann Machines

![Boltzmann Machines diagram](image)

The units are stochastic, taking output value \( s_i = 1 \) with probability \( g(k) \) and value \( s_i = -1 \) with probability \( 1 - g(k) \), where:

\[
\begin{align*}
    h_i &= \sum_j w_{ij} s_j \\
    y(k) &= \frac{1}{1 + \exp(-2h)}
\end{align*}
\]
Boltzmann Machines

\[ E = \sum_{\alpha} R_{\alpha} \log \frac{R_{\alpha}}{P_{\alpha}} \]  

This may be derived from information-theoretic arguments as for (5.52), though we will discuss an alternative statistical mechanics interpretation below. \( E \) is always positive or zero, and can only be zero if \( R_{\alpha} = P_{\alpha} \) for all \( \alpha \). We therefore minimize \( E \), using gradient descent:

\[ \Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \sum_{\alpha} R_{\alpha} \frac{\partial P_{\alpha}}{\partial w_{ij}} \]

Hebbian learning & unlearning

Canonical correlation analysis (CCA)

So far we had one dataset, and looked for a low-dimensional embedding.

But we could also have two datasets, where the vectors come in pairs. Examples:
- sensory stimuli and corresponding motor acts
- experimental conditions and corresponding fMRI images

Goal: find features of one dataset that are related to features of the other dataset.

CCA: “features” are linear projections, “relation” is correlation.

NOTE: Kernel CCA (Blei & Jordan) and neural-net CCA (Becker & Hinton) exist.

The canonical projections do not correspond to the principal components in each dataset: \( x \) may vary a lot in a given direction, but that variation may be unrelated to any variation in the corresponding \( y \) (due to “noise”).
Canonical correlation analysis (CCA)

**Problem:** Given pairs \( \{ x_i \in \mathbb{R}^p; y_i \in \mathbb{R}^k \}_{i=1}^{D} \) with sample means \( \langle x \rangle = \langle y \rangle = 0 \), find \( a_1 \cdots a_d \in \mathbb{R}^p \) and \( b_1 \cdots b_d \in \mathbb{R}^k \) (where \( d = \min(p,k) \)) such that

\[
\text{Cov}(a_n^T x, b_n^T y) \quad \text{is maximal, subject to } \text{Var}(a_n^T x) = \text{Var}(b_n^T y) = 1
\]

and \( \text{Cov}(a_n^T x, a_m^T x) = \text{Cov}(b_n^T y, b_m^T y) = 0 \) for \( n \neq m \)

**Solution:**

1. Subtract the sample means if the data is not centered
2. Compute the sample covariances \( \Sigma_x = \langle xx^T \rangle \), \( \Sigma_y = \langle yy^T \rangle \), \( \Sigma_{xy} = \langle xy^T \rangle \)
3. Compute the matrices \( K = \Sigma_x^{1/2} \Sigma_{xy} \Sigma_y^{1/2}; \quad [A, D, B] = \text{svd}(K) \)
4. Normalize \( A = \Sigma_x^{1/2} A; \quad B = \Sigma_y^{1/2} B \)
5. The columns of \( A \) and \( B \) are the canonical correlation vectors \( a \) and \( b \); the numbers on the diagonal of \( D \) are the canonical correlation coefficients

**Alternative:** type \([A,B,D] = \text{canoncorr}(X,Y)\) in Matlab (Statistics toolbox needed)

Matching computer-generated random postures

Random hand postures are presented one at a time. The subject has to adjust his hand shape to match the displayed posture, press a key, and then get the next posture. There is no time limit.

Performance is poor! The max correlation between instructed and measured joint angles (max over all 19 joints of the hand) is 0.59
Matching computer-generated random postures

Is there anything about the displayed (visual) hand postures that is well matched by something about the measured (motor) hand postures?

Canonical Correlation Analysis can find such things!

There are linear combinations of visual joint angles that are well-correlated with linear combinations of motor joint angles. Max canonical correlation is 0.95

Joint angle vectors represented in canonical correlation coordinates:

Matching computer-generated random postures

Canonical projections correspond to global shape deformations in each space