Unsupervised Learning

• No teacher signal (i.e. no feedback from the environment).

• The network must discover patterns, features, regularities, correlations, or categories in the input data and code them in the output.

• The units and connections must display some degree of self-organization.

• Unsupervised learning can be useful when there is redundancy in the input data.

• A data channel where the input data content is less than the channel capacity, there is redundancy.

What Can Unsupervised Learning Do?

• **Familiarity**: how similar is the current input to past inputs?

• **Principal Component Analysis**: find orthogonal basis vectors (or axes) against which to project high dimensional data.

• **Clustering**: $n$ output class, each representing a distinct category. Each cluster of similar or nearby patterns will be classified as a single class.

• **Prototyping**: For a given input, the most similar output class (or exemplar) is determined.

• **Encoding**: application of clustering/prototyping.

• **Feature Mapping**: topographic mapping of input space onto output network configuration.

Structure, Redundancy, Statistical Dependence

• Each pixel can be seen as a random variable.

• When pixel A can be predicted from looking at pixel B:
  - They are dependent.
  - They are redundant.
  - There is structure.

• Unsupervised learning needs such structure in the input.

Self-Organizing Map (SOM)

• 1-D or 2-D layout of units.

• One reference vector for each unit.

• Unsupervised learning (no target output).
SOM: Map vs. Input Space

- Each weight vector can be plotted in the input space.
- They can then be linked together based on their proximity in the map.

SOM Algorithm

1. Randomly initialize reference vectors $w_i$
2. Randomly sample input vector $x$
3. Find Best Matching Unit (BMU):
   \[ i(x) = \arg\min_j \| x - w_j \| \]
4. Update reference vectors:
   \[ w_j \leftarrow w_j + \alpha \Lambda(j, i(x))(x - w_j) \]
   \( \alpha \) : learning rate
   \( \Lambda(j, i(x)) \) : neighborhood function of BMU.
5. Repeat steps 2 – 4.

Typical Neighborhood Functions

- Gaussian: \( \Lambda(j, i(x)) = \exp(-|r_j - r_{i(x)}|^2/2\sigma^2) \)
- Flat: \( \Lambda(j, i(x)) = 1 \) if \( |r_j - r_{i(x)}| \leq \sigma \), and 0 otherwise.
- \( \sigma \) is called the neighborhood radius.

Training Tips

- Start with large neighborhood radius.
  Gradually decrease radius to a small value.
- Start with high learning rate \( \alpha \).
  Gradually decrease \( \alpha \) to a small value.
Properties of SOM

- **Approximation of input space.**
  Maps continuous input space to discrete output space.

- **Topology preservation.**
  Nearby units represent nearby points in input space.

- **Density mapping.**
  More units represent input space that are more frequently sampled.

Performance Measures

- **Quantization Error**
  Average distance between each data vector and its BMU.
  \[ \epsilon_Q = \frac{1}{N} \sum_{j=1}^{N} \| x_j - w_i(x_j) \| \]

- **Topographic Error**
  The proportion of all data vectors for which first and second BMUs are not adjacent units.
  \[ \epsilon_T = \frac{1}{N} \sum_{j=1}^{N} u(x_j), \]
  \[ u(x) = 1 \text{ if the 1st and 2nd BMUs are not adjacent} \]
  \[ u(x) = 0 \text{ otherwise.} \]

Example: 2D Input / 2D Output

- Train with uniformly random 2D inputs.
  Each input is a point in Cartesian plane.

- Nodes: reference vectors \((x\text{ and } y\text{ coordinate})\).

- Edges: connect immediate neighbors on the map.

Different 2D Input Distributions

- What would the resulting SOM map look like?
- Why would it look like that?
High-Dimensional Inputs

SOM can be trained with inputs of arbitrary dimension.

- Dimensionality reduction: N-D to 2-D.
- Extracts topological features.
- Used for visualization of data.

Applications

- Data clustering and visualization.
- Optimization problems:
  - Traveling salesman problem.
- Semantic maps:
  - Natural language processing.
- Preprocessing for signal and image-processing:
  2. Phonetic map for speech recognition.

Exercise

1. What happens when $N_i(\mathbf{x})$ and $\alpha$ was reduced quickly vs. slowly?

2. How would the map organize if different input distributions are given?

3. For a fixed number of input vectors from real-world data, a different visualization scheme is required. How would you use the number of input vectors that best match each unit to visualize the property of the map?

SOM Example: Handwritten Digit Recognition

- Preprocessing for feedforward networks (supervised learning).
- Better representation for training.
- Better generalization.
SOM Demo

Jochen Fröhlich’s *Neural Networks with JAVA* page:

http://fbim.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html

Check out the *Sample Applet* link.

---

**SOM Demo: Traveling Salesman Problem**

Using Fröhlich’s SOM applet:

- 1D SOM map ($1 \times n$, where $n$ is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 8.
- Stop when radius $< 0.001$.
- Try 50 nodes, 20 input points.

Click on *Parameters* to bring up the config panel. After the parameters are set, click on *Reset* in the main applet, and then *Start learning*.

---

**SOM Demo: Space Filling in 2D**

Using Fröhlich’s SOM applet:

- 1D SOM map ($1 \times n$, where $n$ is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 100.
- Stop when radius $< 0.001$.
- Try 1000 nodes, and 1000 input points.

---

**SOM Demo: Space Filling in 3D**

Using Fröhlich’s SOM applet:

- 2D SOM map ($n \times n$, where $n$ is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 10.
- Stop when radius $< 0.001$.
- Try $30 \times 30$ nodes, and 500 input points. Limit the $y$ range to 15.

Also try $50 \times 50$, 1000 input points, and 16 initial radius.
Other Unsupervised Learning Algorithms

- Hebbian learning: activity-dependent plasticity
- Principal component analysis
- Independent component analysis
- Competitive learning
- Vector quantization
- Various clustering algorithms

Course Wrap Up

- A thought: In ML, learning task is defined by humans. Can machines define their own learning tasks?
- Learning vs. understanding.
- Related courses: Pattern Recognition (689), Neural Networks (636), Cortical Networks (644), Information Retrieval, Sketch Recognition, Robotics, ...
- Conferences: ICML, NIPS, COLT, AAAI, IJCAI, GECCO, IJCNN.

Books