

Representation Learning

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics in Representation Learning

- Role of Representation in Deep Learning
 1. Greedy Layer-Wise Unsupervised Pretraining
 2. Transfer Learning and Domain Adaptation
 3. Semi-supervised Disentangling of Causal Factors
 4. Distributed Representation
 5. Exponential Gains from depth
 6. Providing Clues to Discover Underlying Causes

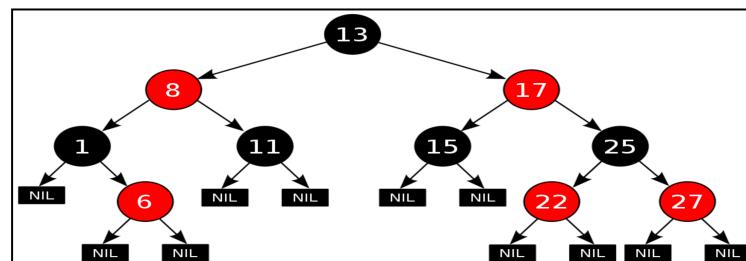
Representation influences task difficulty

1. Performing arithmetic: Arabic vs Roman numerals

- Task of $210/6$ versus CCX/VI using long division
 - Most modern people convert from Roman to Arabic

2. Inserting an entry into a list: Sorted list vs Tree

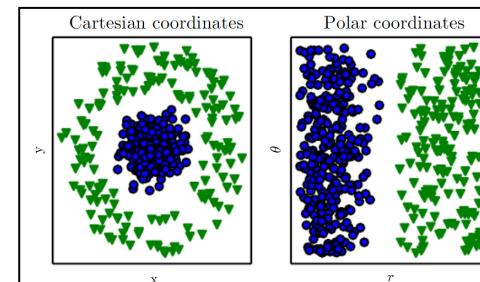
- For a sorted list it is $O(n)$
- Only $O(\log n)$ if list represented as a red-black tree



Left is smaller than parent,
Right is larger than parent
Tree is traversed Left-Root-Right

3. Classification: Cartesian vs Polar coordinates

- Quadratic complexity
- Linear discriminant



Discussion of Representation in DL

1. How is notion of representation useful in deep architecture design
2. Learning algorithms share statistical strength across different tasks
 1. From an unsupervised task to a supervised task
 2. Handle multiple modalities or domains
 3. Transfer learned knowledge to tasks with few examples

Discussion: Reasons for success of Representation Learning in DL

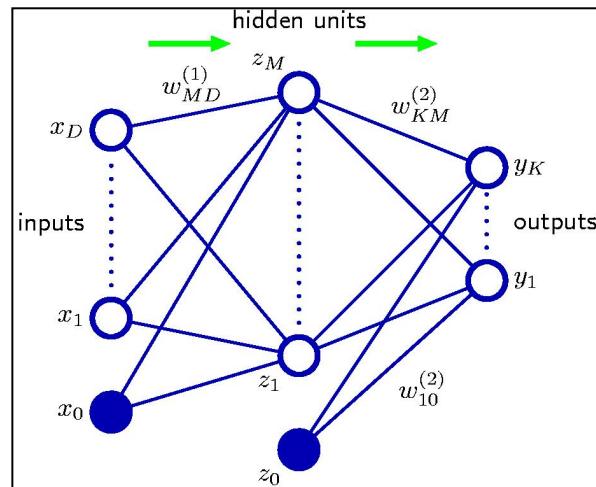
1. Theoretical advantages of distributed representation and deep representations
2. Underlying assumptions about data generation process, e.g.,
 - Underlying causes of the observed data

What is a good representation for ML?

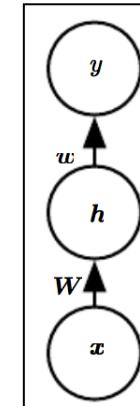
- Answer:
 - Representation makes subsequent learning easier
- Choice usually depends on subsequent learning task

Feed-forward nets learn representations

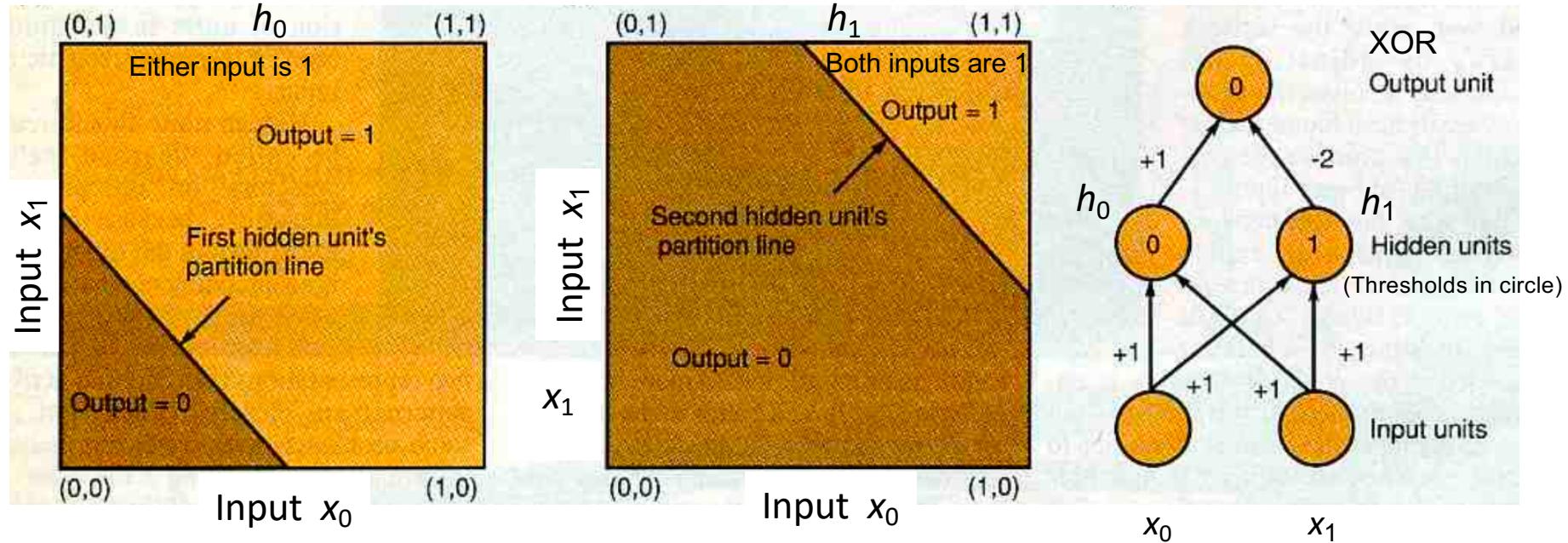
- A feed-forward network $f(\mathbf{x}) = f^{(3)} [f^{(2)} [f^{(1)}(\mathbf{x})]]$ trained by supervised learning performs representation learning



$$\mathbf{y}(\mathbf{x}) = f^{(2)} [f^{(1)}(\mathbf{x})]$$



Representation learnt in XOR



Famously, XOR cannot be solved by a linear function.
But the network discovers a representation that is linearly separable.

Input space (x_0, x_1) is a unit square

h_0 partitions the space so that it is activated when either input is 1, i.e., $(0,1), (1,0), (1,1)$
 h_1 partitions the space so that it is activated only when both inputs are 1

Example: An input $x=01$ is represented as $h=10$, inputs to final output are 10, network outputs 1

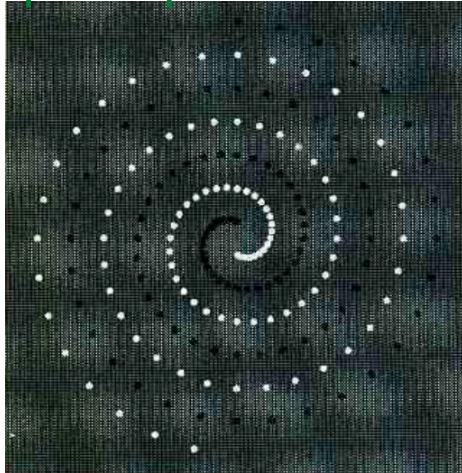
Input $x=00$ represented as $h=00$, inputs to final output are 00, and network output is 0

Input $x=11$ represented as $h=11$, inputs to final output are 11 and network output is 0

Input $x=10$ represented as $h=10$, inputs to final output are 10 and network output is 1

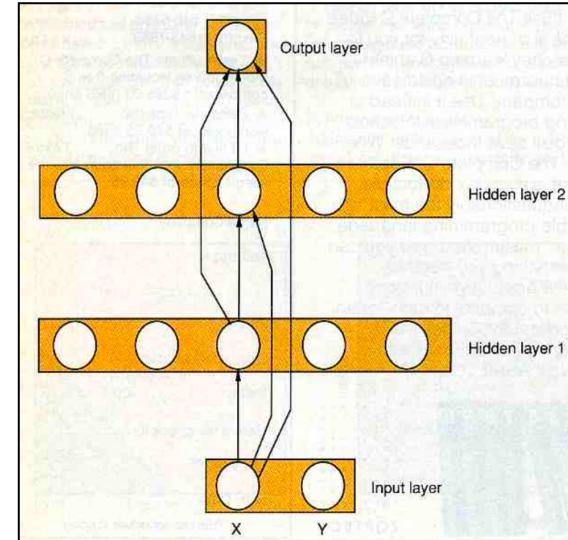
Representations learnt in 2-Spirals

Training points for the Two-spirals problem.



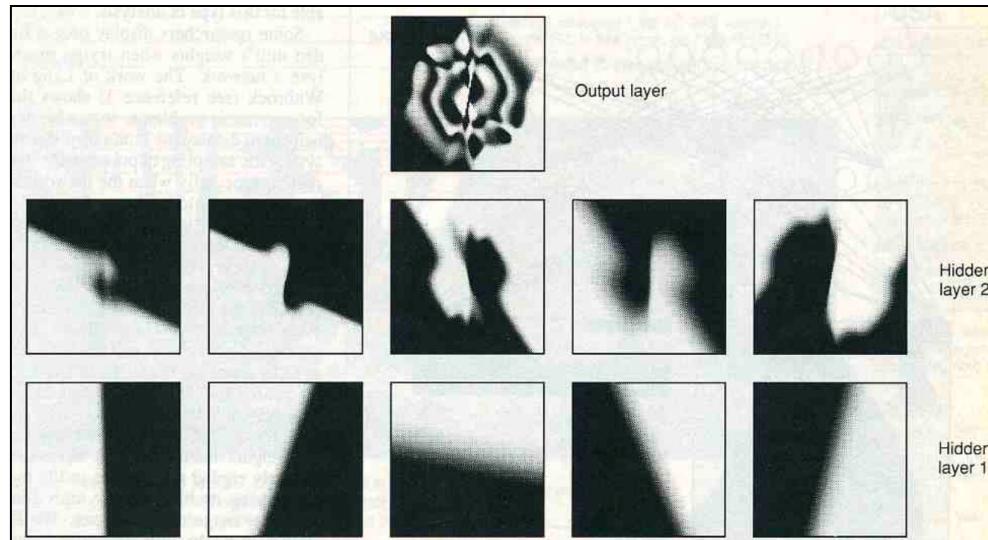
Black points should produce an output 0;
White points should produce an output 1

Feedforward Network



Each unit receives input from all the units in all the layers below it.

Response function plots for the units in the network.

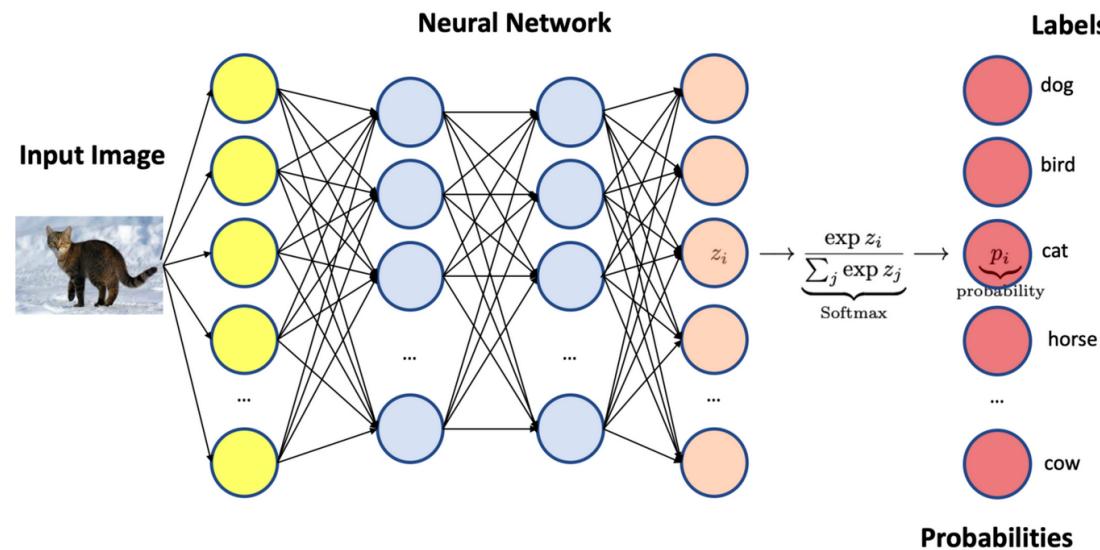


Each plot shows the activation level of a single unit as the x, y input to the network ranges over the unit square.

Level 1 layer outputs
are all linear

Supervised Feedforward Learning

- Last layer of network is a linear classifier such as softmax regression classifier
 - Rest of network learns representation for classifier



Hidden layer properties

- Every hidden layer takes on properties that make the classification easier
 - Ex: classes not linearly separable in input features may become linearly separable in the last hidden layer
- Last layer could also be another model:
 - Such as a nearest neighbor classifier
 - Features in penultimate layer should learn different properties depending on type of last layer

How do we specify representation?

1. Supervised learning of feed-forward networks:
 - No imposition of any conditions on learned features
2. Other representation learning algorithms do so
 - Ex: In density estimation, encourage h_i to be independent
3. Unsupervised deep learning algorithms
 - have a main training objective, but like supervised learning they learn a representation as a side effect
 - Regardless of how representation was obtained, it can be used for another task
 - Multiple tasks (supervised/unsupervised) can share representation

Trade-off in representation

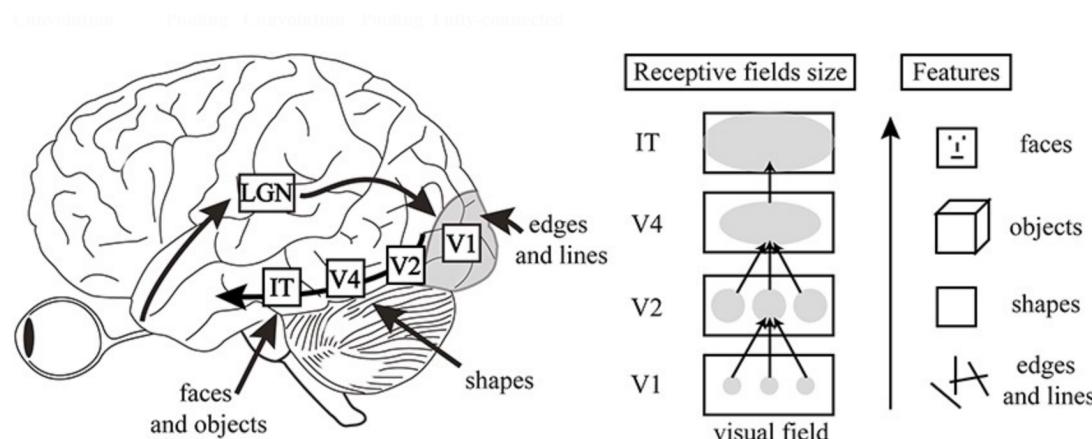
- Representation learning involves a trade-off between:
 1. Preserving as much information about the input as possible
 2. Attaining nice properties (such as independence)

Semi-supervised Learning

- Representation learning is a way of performing unsupervised and semi-supervised learning
 - Often we have very little labeled data and very large amounts of unlabeled data
 - Training on labeled data results in severe overfitting
 - Semi-supervised learning offers a solution

Human/Animal Learning

- Humans learn with few labeled samples
- We do not yet know how this is possible



Hypotheses for learning with few samples

- The brain may use
 1. Very large ensembles of classifiers
 2. Bayesian inference techniques
 3. Leverages unsupervised or semi-supervised learning
- There are many ways to leverage unlabeled data
 - We focus on hypothesis that unlabeled data can be used to learn a good representation

Unsupervised Learning and Deep Learning

- Unsupervised learning revived deep neural networks
 - Enabling training a deep supervised network without specializations such as convolution or recurrence
- Canonical example of a representation learned for one task can be useful for another task
 - First task: unsupervised learning (trying to capture the shape of a distribution)
 - Other task: supervised learning with the same input domain