Representation Learning

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics in Representation 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

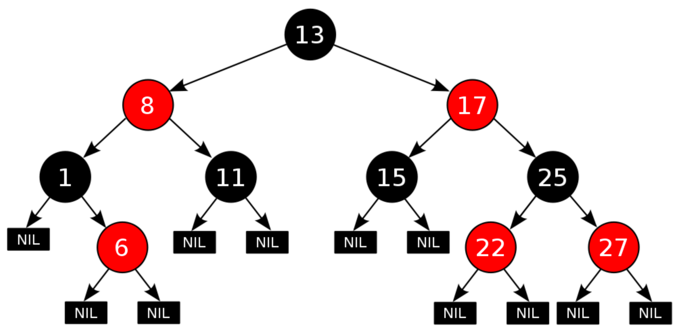
Representation Learning Overview

- What it means to learn representations
  - Role of representation in deep architecture design
  - Sharing statistical strength across different tasks
- Shared representations useful to handle multiple modalities or domains
  - Or to transfer learned knowledge to tasks for which few or no examples are given but a task representation exists
- Reasons for success of representation learning
  - Theoretical advantages
  - Underlying principles of data generation process

Importance of Representation

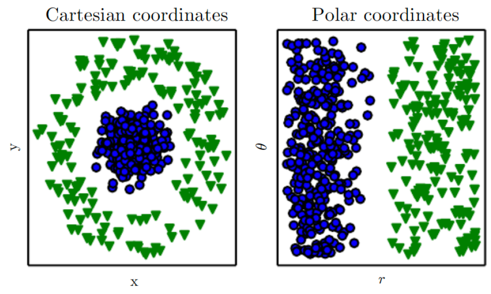
- Tasks can be easy or difficult depending on how information is represented
  - Arabic or Roman numeral
    - Task of 210/6 versus CCX/VI using long division
      - Most modern people convert from Roman to Arabic
  - We can quantify asymptotic run time of various operations using appropriate or inappropriate reps.
    - Inserting a no. in a sorted list is $O(n)$
    - But only $O(\log\ n\ )$ if list represented as a red-black tree

Tree is traversed
Left-Root-Right



What's a good representation for ML?

- Ans: It makes subsequent learning task easier
- Feedforward networks trained by supervised learning perform representation learning
  - Last layer is a linear classifier such as softmax regression classifier
  - Rest of network learns representation for classifier
  - Every hidden layer makes the classification easier
    - Ex: classes not linearly separable in input features may become linearly separable in the last layer
    - Last layer could also be another model:
      - Such as a nearest neighbor classfier



How do we specify representation?

- Supervised learning of feed-forward networks:
  - No imposition of any conditions on learned intermediate features
  - Other representation learning algorithms do so
    - Ex: to make density estimation easier
      - Distributions with more independences are easier to model, so encourage elements of representation h to be independent
- Unsupervised deep learning algorithms
  - have a main training objective, but like supervised learning they learn a representation as a side effect
- Regardless of representation was obtained, it can be used for another task

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
- Humans learn quickly from few labeled examples
  - One hypothesis is that the brain leverages unsupervised or semi-supervised learning

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

Greedy Layer-wise Unsupervised Pretraining

- Relies on:
  - A single-layer representation learning algorithm
  - A single-layer autoencoder
  - A sparse coding model
  - Or another model that learns latent representations
- Each layer is pretrained using unsupervised learning
  - Taking the output of the previous layer and producing as output a new representation of data,
    - Whose distribution (or relation to categories) is simpler
- Formal algorithm is given next

Greedy layer-wise unsupervised pretraining protocol

Algorithm:

- Given unsupervised feature learning algorithm $\mathcal{L}$
  - Which takes as input a training set of examples and returns an encoder or feature function f.
- Raw input data is $\boldsymbol{X}$, with one row per example, $f^{(1)}(\boldsymbol{X})$ is output of the first stage encoder on $\boldsymbol{X}$.
- In the case where fine tuning is performed we use

  a learner $\mathcal{T}$ which takes an initial function f, input examples $\boldsymbol{X}$ (and in the supervised fine-tuning case, associated targets $\boldsymbol{Y}$) and returns a tuned function.
  The no of stages is m.

$f \leftarrow$ Identity function
$\tilde{\boldsymbol{X}} = \boldsymbol{X}$
for $k = 1, \ldots, m$ **do**
  $f^{(k)} = \mathcal{L}(\tilde{\boldsymbol{X}})$
  $f \leftarrow f^{(k)} \circ f$
  $\tilde{\boldsymbol{X}} \leftarrow f^{(k)}(\tilde{\boldsymbol{X}})$
end for
if *fine-tuning* ***then***
  $f \leftarrow \mathcal{T}(f, \boldsymbol{X}, \boldsymbol{Y})$
end if
Return f

History: layer-wise unsupervised

- Unsupervised greedy layer-wise training
  - was used to sidestep difficulty of training layers of a deep neural net for a supervised task
  - Origins in Neocognitron (Fukushima, 1975)
  - Deep learning renaissance of 2006 began with
    - Greedy learning to find initialization for all layers
      - Useful for fully connected architectures
    - Earlier, only deep CNNs or depth resulting from recurrence were feasible to train
- Today greedy layer-wise pretraining is not required to train fully connected deep networks

Greedy pretraining terminology

- Greedy layer-wise pretraining
  - Greedy because
    - It is a greedy algorithm that optimizes each piece of the solution independently
      - One piece at a time rather than jointly
  - Layer-wise because
    - Independent pieces are the layers of the network
    - Training proceeds one layer at a time
      - Training the k^{th} layer while previous ones are fixed
  - Pretraining because
    - It is only a first step before applying a joint training algorithm is applied to *fine-tune* all layers together

When/why does pretraining work?

- Greedy layer-wise unsupervised pretraining can yield substantial improvements for classification
  - However it is sometimes *harmful*
- Pretraining accesses new part of space:
  - With pretraining: halt in one region of function space
  - Without pretraining: another region

Visualization of functions projected into 2d space.
(Each function is an infinite-dimensional vector that associates every input x with output y).
Color indicates time.
Area where pretrained networks arrive is smaller.