Representation Learning and Autoencoders

Lena Kasberger

February 2, 2017

Overview

1. Motivation
2. Representation learning
   - Clues for disentangling factors
   - Training of deep networks
3. Autoencoders
   - Principles
   - Types of autoencoders
   - Application and implementation
4. Conclusion

Learning representations - what for?

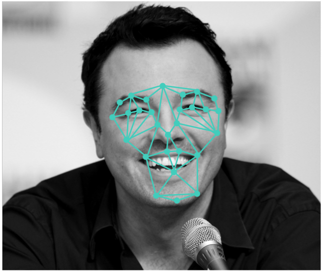

Figure: Key features of human facial expression

- Fundamental concept in deep learning
- Applications: speech/object recognition, NLP, transfer learning
- Making use of inexpensive unlabeled data in unsupervised feature learning, e.g. autoencoders

Good representation

- Feature engineering as main task in ML
  $\rightarrow$ good representation essential for success
- Handcrafting features vs. learning them
- Characterized by facilitating subsequent tasks
- Good representation of observed data consists of *guessing* features, factors and causes
  $\rightarrow$ invariant features
  $\rightarrow$ learning to disentangle underlying explanatory factors

$\Rightarrow$ Clues for disentangling factors necessary

Distributed representations

- Existence of multiple factors
- Only limited generalization possible for non-distributed representations
- Influence of parameters on many regions, not just local neighbors
- Advantages and statistical importance:
  $\rightarrow$ large set of concepts (k^n)
  $\rightarrow$ non-local generalization to never-seen regions

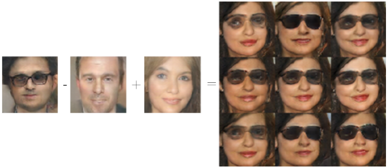

Figure: Example for distributed representations

Depth of representations

- Learning multiple levels of representation increasing with regard to complexity → hierarchical structure

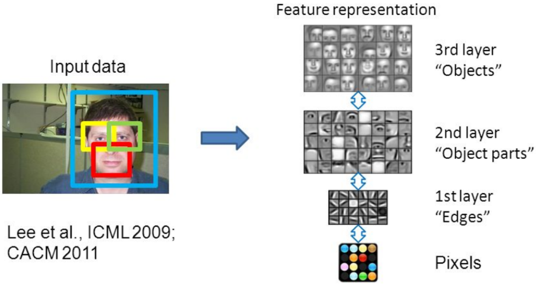

Figure: Learning feature hierarchy

- Higher-level features formed by composition of lower-level ones

Semi-supervised learning

- Hypothesis: $P(x)$ shares structure with $P(y|x)$

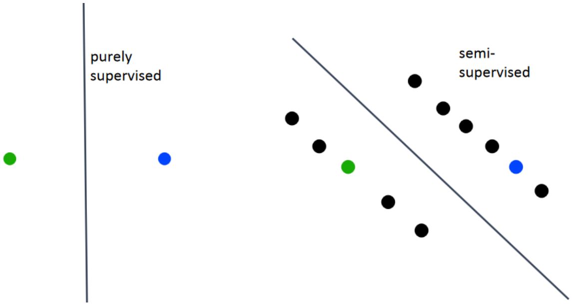

Figure: Structure of supervised vs. semi-supervised learning

- Sharing statistical strength between unsupervised and supervised learning task

Sharing factors in different settings

Transfer learning

- Knowledge transfer for different tasks
- Occurrence of shared factors in input or output

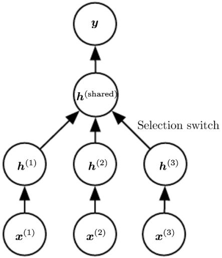

Figure: Example architecture of transfer learning

Domain adaption $\rightarrow$ same tasks but varying input distributions

Further priors

- Manifold hypothesis
- Natural clustering
- Temporal and spatial coherence
- Sparsity
- Simplicity of factor dependencies

Idea of unsupervised pretraining

- Initialization of hidden layers by using unsupervised learning
  $\rightarrow$ network forced to represent latent structure of input distribution

Figure: Character image vs. random image

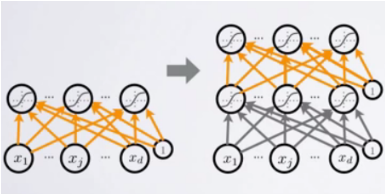

Figure: Structure of unsupervised pretraining

Steps

Unsupervised pretraining

- Greedy layer-wise procedure:
  training of one layer at a time with supervised criterion
  → parameters of previous hidden layers fixed
  → previous layers viewed as feature extraction

Fine-tuning

- After pretraining all layers:
  → adding output layer
  → training whole network by means of supervised learning

- Application for deep networks, large unlabeled data sets and complicated functions

- Recent techniques rely on jointly training with supervised and unsupervised objective

Setup of an autoencoder

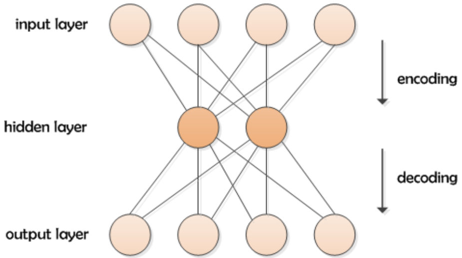

Figure: Example structure of fully connected autoencoder

- Three layers: input, hidden, output layer
- Two steps: encoding and decoding
- Same size of input and output layer, smaller hidden layer in between
  $\rightarrow$ low-dimensional representation

Properties

- Encoding $\mathbf{h} = f(\mathbf{x})$ and decoding $r = g(\mathbf{h})$
- $g(f(\mathbf{x}))$ restricted not to be equal to $\mathbf{x}$
  $\rightarrow$ only approximation of identity function

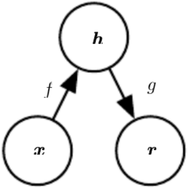

Figure: General structure

- Special kind of artificial neural network for unsupervised learning
- Designed to automatically learn from unlabeled data
- Learn compact and meaningful representation

Example for data compression

- **Example**:
  Program for sending data from cellphone to cloud

- **Steps**:
  $\rightarrow$ Encoding: in the cellphone, map data $\mathbf{x}$ to compressed data $\mathbf{h}$
  $\rightarrow$ Sending: send compressed data $\mathbf{h}$ to the cloud
  $\rightarrow$ Decoding: in the cloud, map compressed data $\mathbf{h}$ back to $\mathbf{r}$

- **Objective function**:

$$
\begin{aligned}
J(W_1, b_1, W_2, b_2) &= \sum_{i=1}^{m} \left(r^{(i)} - x^{(i)} \right)^2 \\
&= \sum_{i=1}^{m} \left(W_2 h^{(i)} + b_2 - x^{(i)} \right)^2 \\
&= \sum_{i=1}^{m} \left(W_2 (W_1 x^{(i)} + b_1) + b_2 - x^{(i)} \right)^2
\end{aligned}
$$

Impact of size

Learning process as minimization of the loss function

$$L(\mathbf{x}, g(f(\mathbf{x}))) = L(\mathbf{x}, \mathbf{r})$$

Undercomplete autoencoders

- Restriction: dimension of hidden layer lower than input
  $\rightarrow$ extraction of most meaningful features
- Good compression only for training example

Overcomplete autoencoders

- Hidden layer larger than input $\rightarrow$ no compression
- Perfect reconstruction by pure copying possible

$\Rightarrow$ Solution: some type of **regularization**

Denoising autoencoders

- **Idea**: Robustness to noise
  $\rightarrow$ feed noisy input ($\tilde{\mathbf{x}}$) to autoencoder and train it to reconstruct the uncorrupted input

- Comparison of reconstruction with original input by loss function $L(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$ $\rightarrow$ learning structure of input distribution

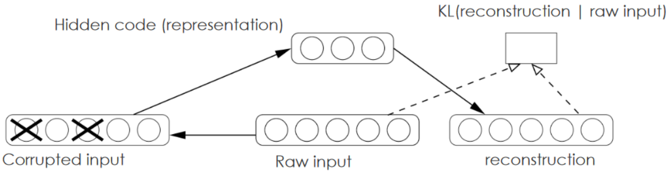

Figure: Structure of denoising autoencoder

Sparse autoencoders

- **Idea**: Discover interesting structure by adding sparsity constraint
- Loss function: $L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h})$,
- $\Omega(\mathbf{h}) = \beta \sum_{j=1}^{J} KL(\rho || \rho_j) \rightarrow$ enforcing the average activation of hidden units to be near a given sparsity parameter
- Learning overcomplete representation but in sparse manner $\rightarrow$ only informative set of units activated

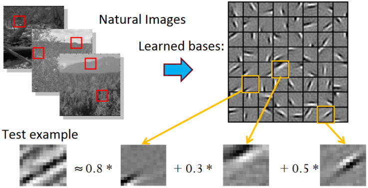

Figure: Sparse autoencoder illustration for images

Contractive autoencoders

- **Idea**: Avoid uninteresting solutions by adding explicit term in the loss function penalizing this solution
- Learning invariant representations to unimportant transformations
- Extraction of features only reflecting variations observed in the training set
- Penalty ensuring the derivatives of the encoder to be as small as possible $\rightarrow$ contraction in all directions
- Encoder keeps only *good* information

Stacked autoencoders

Steps:

- Stack shallow autoencoders in succession to form a deep network
- Train by using greedy layer-wise unsupervised pretraining
- Accomplish supervised training on the last layer using final features on the entire network for fine-tuning the weights

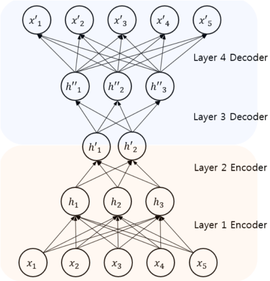

Figure: Structure of stacked autoencoder

Manifold hypothesis

- Data generating distribution assumed to **concentrate near regions of low dimensionality**
- Random choice of configurations very unlikely to generate the kind of information to be modeled
- Probability distribution of interest concentrates in tiny volume concerning total space of configurations
- Goal of manifold learning: characterizing where probability concentrates
- **Probable configurations surrounded by the same ones**
- Difficulty: generalization only with huge amount of data possible
- Attempt of autoencoders to explicitly learn the structure of the manfifold

Learning manifolds

- Regularized autoencoder: **Tradeoff** between two forces
  1. Reconstruction error ($\rightarrow$ keep enough information)
  2. Constraint ($\rightarrow$ as insensitive as possible to the input)

- Solution for learned representation:
  sensitive to changes along the manifold, invariant to changes orthogonal to the manifold (contraction in orthogonal direction)

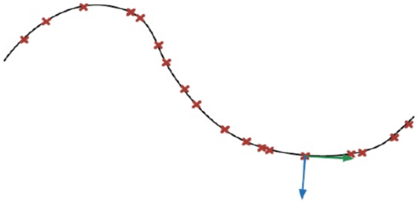

Figure: Learning manifolds with regularized autoencoders

Package '*autoencoder*' in R

```
# Training an autoencoder

autoencoder.object <-
autoencode(X.train=training.matrix,nl=3,N.hidden=5*5,
unit.type="sigmoid",lambda=0.0002,beta=6,
rho=0.01,epsilon=0.001,
max.iterations=2000,rescale.flag=TRUE)

# lambda: weight decay parameter,
# beta: weight of sparsity penalty term,
# rho: desired sparsity parameter,
# epsilon: small parameter for initialization of weights
```

Alternatives: 'RcppDL' and 'h2o' (R), Theano and Keras (Python)

Example patches

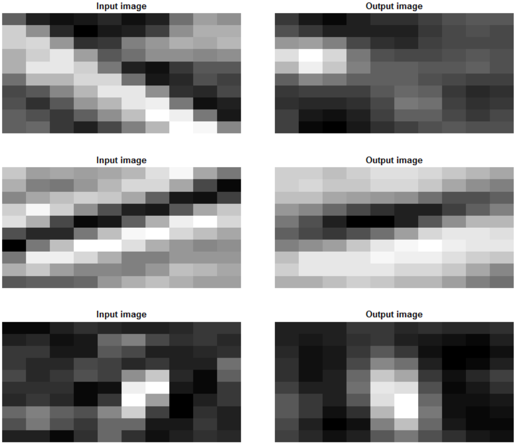

Figure: Input vs. output images

Conclusion and outlook

- Representation learning as crucial part of many concepts concerning deep learning
- Existence of regularization strategies for detection of underlying causal factors
- Former importance of greedy layer-wise unsupervised pretraining
- Finding best representation as worthwhile focus of future research
- Some form of regularization as key property of autoencoders
- Advantages of stacked auotencoders being composition of shallow ones
- Usage of autoencoders for dimensionality reduction, feature extraction and initialization method

Bibliography

- Yoshua Bengio, Aaron Courville, Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (8): 1798-1828, 2013.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, p. 502-557. *MIT Press*, 2016. `http://www.deeplearningbook.org`
- Yoshua Bengio. Deep Learning of Representations. *Google Tech Talk* (11/13/2012). `https://www.youtube.com/watch?v=4xsVFLnHC_0` (last accessed on 02/01/2017)
- Andrew Ng. Sparse autoencoder. *CS294A Lecture notes.* `http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf` (last accessed on 02/01/2017)