Representation Learning: 2012 Perspectives

Alex Fedorov

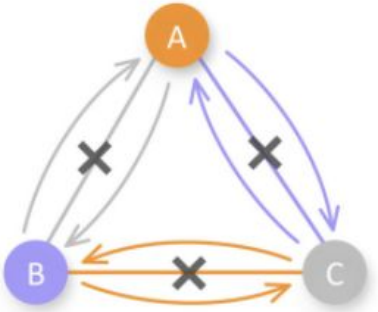
Representation Learning: A Review and New Perspectives
Yoshua Bengio, Aaron Courville, Pascal Vincent
https://arxiv.org/abs/1206.5538

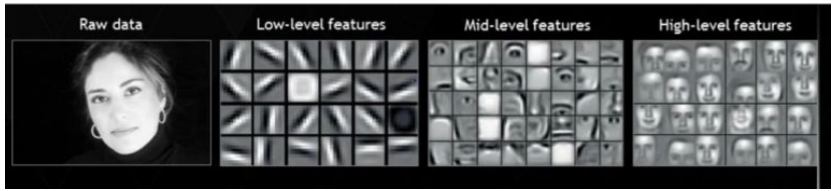
What makes representation good?

Priors for Representation Learning in AI

Priors for Representation Learning in AI

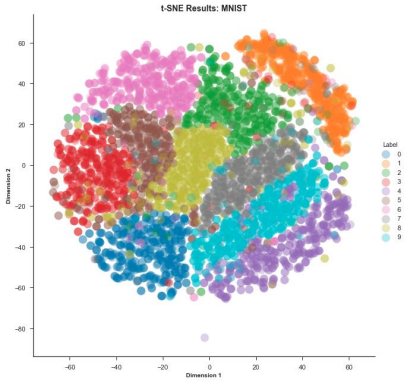
- Smoothness
  - $x \simeq y$ implies $f(x) \simeq f(y)$
- Multiple explanatory factors
  - Disentangling underlying factors of variation
- Hierarchical organization of explanatory factors
  - Deep representations
- Semi-supervised learning
  - P(Y | X) can be useful for modeling P(X) too
- Shared factors across tasks
  - P(Y | X, task)

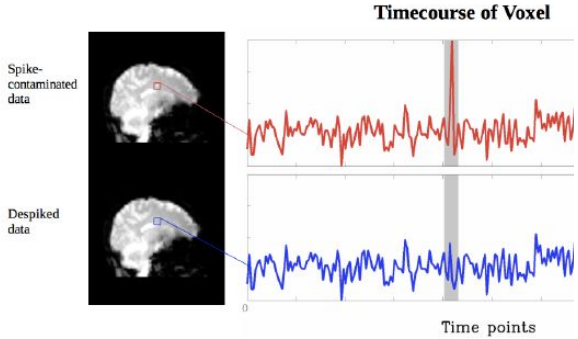




Priors for Representation Learning in AI

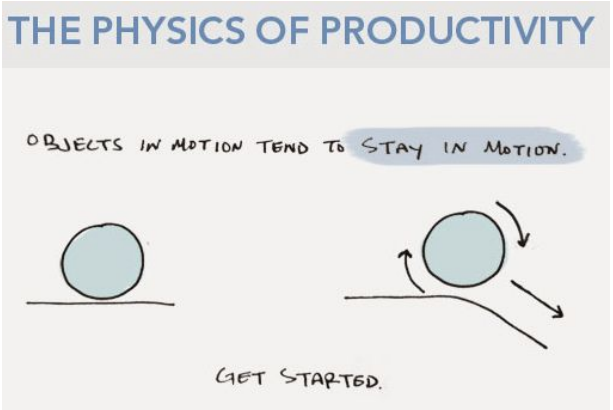
- Manifolds
  - Probability mass concentrates near regions that have a much smaller dimensionality than the original space
- Natural clustering
  - Different values of categorical variables (object classes) are associated with separate manifolds
  - Local variation preserves the value of category
  - Linear interpolation between examples involves going through low density regions
- Temporal and spatial coherence
  - Consecutive or spatially nearby observations tend to be associated with same categories
    - Results in a small move on the surface of the high-density manifold
    - Categorical concepts of interest change slowly
    - Can be implementented as penalizing changes in values over time and space (Becker, Hinton, 1992)





Priors for Representation Learning in AI

- Sparsity
  - For any given X a small fraction of factors are relevant
  - This can be achieved
    - priors on latent variables (peaked at 0)
    - by using a nonlinearity whose value is often flat at 0 (i.e., 0 and with a 0 derivative),
    - simply by penalizing the magnitude of the Jacobian matrix (of derivatives) of the function mapping input to representation
- Simplicity of Factor dependencies
  - In good high level representation the factors a each other through simple and typically linear dependencies.
    - Laws of physics



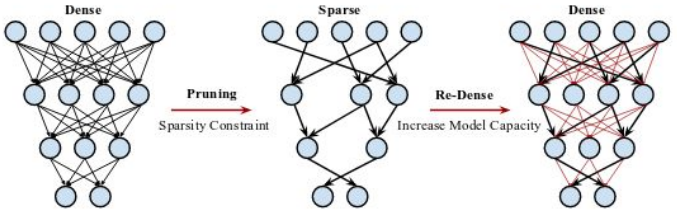
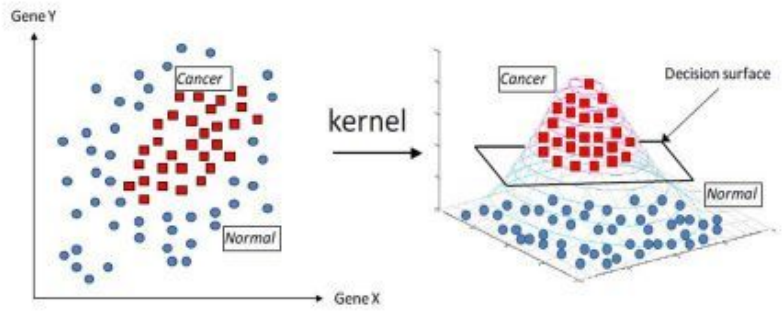


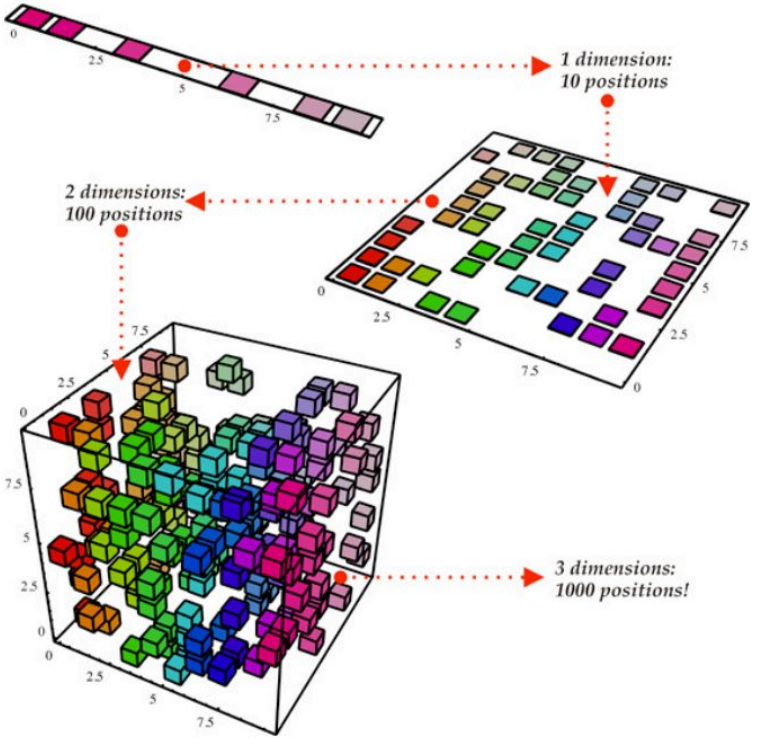
Figure 1: Dense-Sparse-Dense Training Flow. The sparse training regularizes the model, and the final dense training restores the pruned weights (red), increasing the model capacity without overfitting.

Smoothness and the Curse of Dimensionality

Smoothness and the Curse of Dimensionality

- For AI-tasks it hopeless to rely on simple parametric models (Linear)
  - Don't capture enough of the complexity of interest
    - Unless provided with appropriate feature space
- Local non-parametric learners



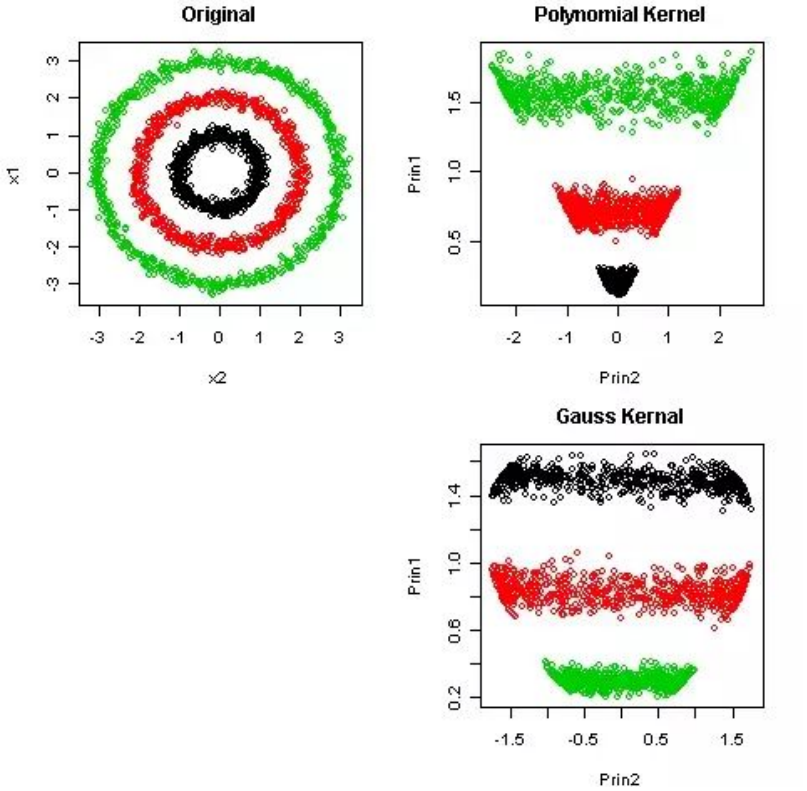


Local non-parametric learners

- Kernel machines with fixed generic local-response kernel
- Exploit the principle of **local generalization**
  - The assumption that the target function (to be learned) is smooth enough, so they rely on examples to explicitly map out the wrinkles of the target function
  - Generalization is mostly *achieved* by a form of *local interpolation between neighboring training examples*
- Smoothness can be a useful assumption, it is insufficient to deal with the curse of dimensionality
- The number of such wrinkles (ups and downs of the target function) may grow exponentially with the number of relevant interacting factors, when the data are represented in raw input space
  - **Curse of dimensionality**

Local non-parametric learners

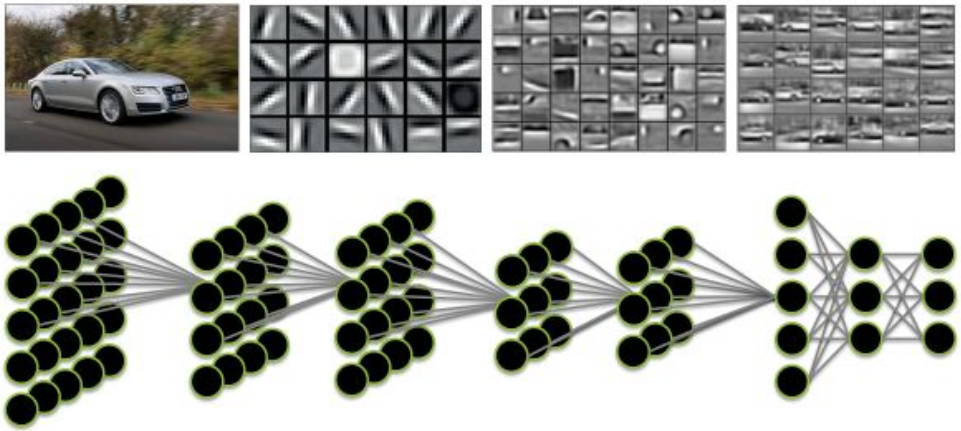
- The combination of learning a representation and kernel machine is equivalent to learning the kernel
- Kernel machines are useful, but they depend on a prior definition of a suitable similarity metric, or a feature space in which naive similarity metrics suffice



Distributed representations

Distributed representations

- Good representations are expressive
  - A reasonably-sized learned representation can capture a huge number of possible input configurations



NVIDIA

How many parameters does it require compared to the number of input regions (or configurations) it can distinguish?

- Learners of one-hot representations all require **O(N) parameters** (and/or **O(N) examples**) to distinguish **O(N) input regions**
  - such as traditional clustering algorithms, Gaussian mixtures, nearest neighbor algorithms, decision trees, or Gaussian SVMs
- RBMs, sparse coding, auto-encoders or multi-layer neural networks can all represent up to **O(2^k) input regions** using only **O(N) parameters** (with k the number of non-zero elements in a sparse representation, and k = N in non-sparse RBMs and other dense representations).
  - These are all distributed or sparse representations.
  - The generalization of clustering to distributed representations is multi-clustering

Why Distributed representations?

- The exponential gain from distributed or sparse representations
- It comes about because each parameter (e.g. the parameters of one of the units in a sparse code, or one of the units in a Restricted Boltzmann Machine)
- Can be re-used in many examples that are not simply near neighbors of each other
- Whereas with local generalization, different regions in input space are basically associated with their own private set of parameters, e.g., as in decision trees, nearest-neighbors, Gaussian SVMs, etc.

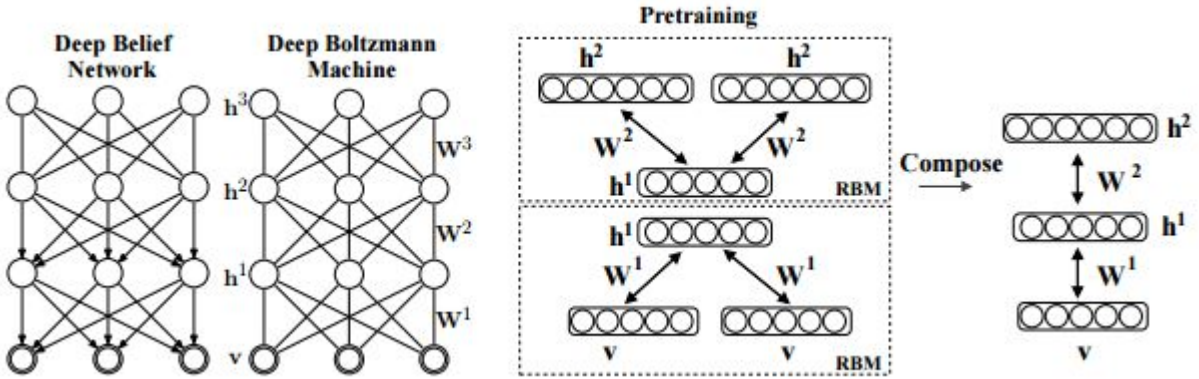
Why Distributed representations?

- In a ***distributed representation***, an exponentially large number of possible subsets of features or hidden units can be activated in response to a given input.
- In a ***single-layer model***,
  - each feature is typically associated with a preferred input direction, corresponding to a hyperplane in input space,
  - the code or representation associated with that input is precisely the pattern of activation
- A non-distributed representation such as the one learned by most clustering algorithms, e.g., k-means
  - the representation of a given input vector is a one-hot code identifying which one of a small number of cluster centroids best represents the input

Depth and abstraction

Depth and abstraction

- Depth is a key aspect to representation learning strategies in Deep Learning
- Deep architectures are often challenging to train effectively
- Advantages
  - Deep architectures promote the re-use of features,
  - Deep architectures can potentially lead to progressively more abstract features at higher layers of representations



Feature re-use

- The power of distributed representations
- The heart of the theoretical advantages behind deep learning
  - Constructing multiple levels of representation or learning a hierarchy of features
- The typical computations we allow in each node include:
  - weighted sum, product, artificial neuron model (such as a monotone nonlinearity on top of an affine transformation), computation of a kernel, or logic gates
- A deep representation can be exponentially more efficient than one that is insufficiently deep
  - (Hastad, ˚ 1986; Hastad and Goldmann, 1991; Bengio ˚ et al., 2006a; Bengio and LeCun, 2007; Bengio and Delalleau, 2011)
- The family of functions can be represented with a smaller VC-dimension yielding improvements in both computational efficiency and statistical efficiency

Abstraction and invariance

- Abstractions comes explicitly via a pooling mechanism
- More abstract concepts are generally invariant to most local changes of the input
- More abstract representations detect categories that cover more varied phenomena (e.g. larger manifolds with more wrinkles) and thus they potentially have greater predictive power
- Abstraction can also appear in high-level continuous-valued attributes that are only sensitive to some very specific types of changes in the input.

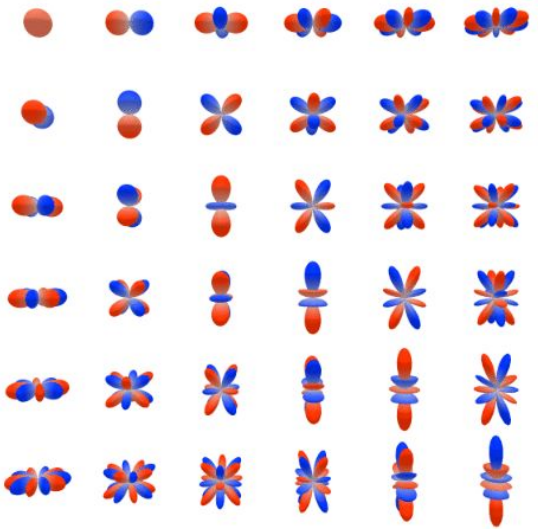
Disentangling Factors of Variation

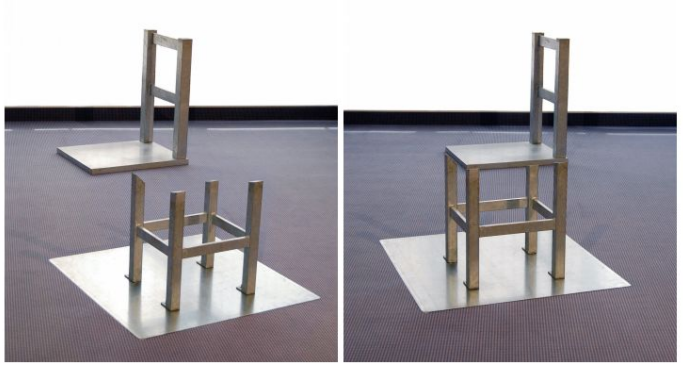
Disentangling Factors of Variation

- Beyond being ***distributed*** and ***invariant***, we would like to ***disentangle*** the factors of variation
- Complex data arise from the rich interaction of many sources.
  - between one or more light sources or shadows
  - the object shapes,
  - the material properties of the various surfaces present in the image
- How can we cope with these complex interactions?
- How can we disentangle the objects and their shadows?
- To solve
  - must leverage the data itself, using vast quantities of unlabeled examples, to learn representations that separate the various explanatory sources.
  - more robust to the complex and richly structured variations extant in natural data sources

Invariance and Disentanglement

- The central difference is the preservation of information
- ***Invariant features,*** by definition, have reduced sensitivity in the direction of invariance which are uninformative to the task at hand
  - Unfortunately, it is often difficult to determine a priori which set of features and variations will ultimately be relevant to the task at hand
  - Further, we might to reuse features as in multi-task learning and meta-learning
- Most robust approach is Disentanglement
  - to disentangle as many factors as possible, discarding as little information about the data as is practical





Good criteria for learning representations?

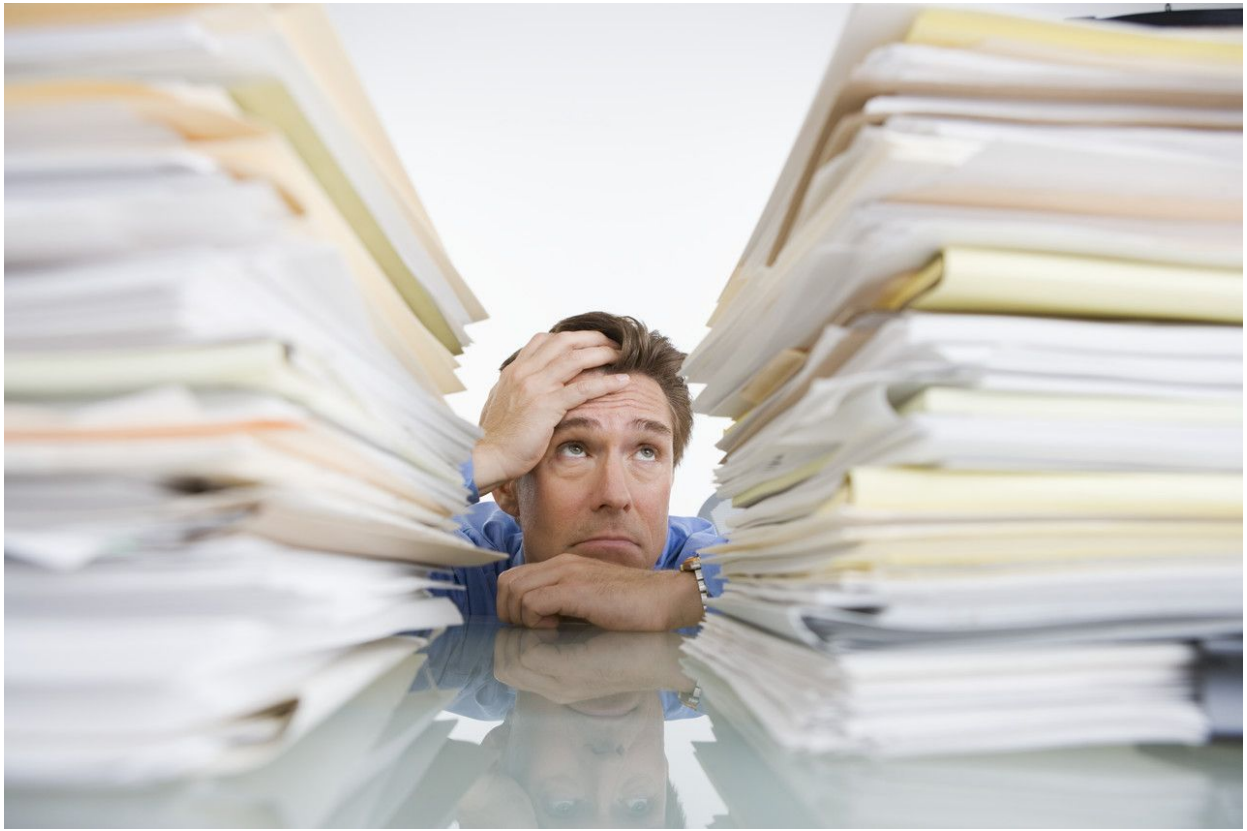
Good criteria for representation learning?

- It is difficult to establish a clear objective, or target for training
  - In classification
    - minimize the number of misclassifications on the training dataset
- Our objective is far-removed from the ultimate objective, which is typically learning a classifier or some other predictor
- A good representation is one that disentangles the underlying factors of variation
- How do we translate that into appropriate training criteria? I
  - Is it even necessary to do anything but maximize likelihood under a good model?
  - Can we introduce priors possibly that help the representation better do this disentangling?

BUILDING DEEP REPRESENTATIONS

History

- Greedy layerwise unsupervised pre-training
  - Was to learn a hierarchy of features one level at a time, using unsupervised feature learning to learn a new transformation at each level to be composed with the previously learned transformations
- Deep Boltzmann Machine
  - Salakhutdinov and Hinton, 2009
  - Deep Belief Network
    - Hinton et al., 2006
- Deep auto-encoder
  - Hinton and Salakhutdinov, 2006
- A free energy function
  - Ngiam et al., 2011
  - The question is then how to train a model defined by an arbitrary parametrized (free) energy function.
    - Ngiam et al. (2011) have used Hybrid Monte Carlo (Neal, 1993),
    - contrastive divergence (Hinton, 1999; Hinton et al., 2006),
    - score matching (Hyvarinen, 2005; Hyvarinen, ¨ 2008),
    - denoising score matching (Kingma and LeCun, 2010; Vincent, 2011),
    - ratio-matching (Hyvarinen, 2007)
    - noise-contrastive estimation (Gutmann and Hyvarinen, 2010).



To be continued...