Understand Deep Learning

Xiaogang Wang

xgwang@ee.cuhk.edu.hk

March 29, 2015

Outline

What is a good representation?

- Makes the further learning easier
  - The joint distribution of different elements of the representation **h** is easy to model (e.g. they are mutually independent)
  - The representation keeps the information (or at least all the relevant information in the supervised case) and makes it easy to learn functions of interest from this representation
- An ideal representation is one that disentangles the underlying causal factors of variation that generated the observed data
- Once we “understand” the underlying explanations for what we observe, it generally becomes easy to predict one thing from others

When does unsupervised learning help supervised learning?

- Whether unsupervised learning on input variables $\mathbf{x}$ can yield representations that are useful when later trying to learn to predict some target variable $\mathbf{y}$, given $\mathbf{x}$
- Whether $P(\mathbf{y}|\mathbf{x})$ seen as a function of has anything $\mathbf{x}$ to do with $P(\mathbf{x})$
  - If $P(\mathbf{x})$ is uniformly distributed and $E[\mathbf{y}|\mathbf{x}]$ is some function of interest, observing $\mathbf{x}$ alone gives us no information about $P(\mathbf{y}|\mathbf{x})$
  - If $\mathbf{x}$ arises from a mixture, with one component per value of $\mathbf{y}$, and the mixture components are well separated, then modeling $P(\mathbf{x})$ tells us precisely where each component is, and a single single labeled example of each component will then be enough to perfectly learn $P(\mathbf{y}|\mathbf{x})$.
- If $\mathbf{y}$ is closely associated with one of the causal factors of $\mathbf{x}$, $P(\mathbf{x})$ and $P(\mathbf{y}|\mathbf{x})$ will be strongly tied, and unsupervised representation learning that tries to disentangle the underlying factors of variation is likely to be useful as a semi-supervised learning strategy.

Disentangle the underlying factors of variation

- Assuming that **y** is one of the causal factors of **x**, and let **h** represent all those factors, then the data has marginal probability

$$P(\mathbf{x}) = \int_{\mathbf{h}} P(\mathbf{x}|\mathbf{h})p(\mathbf{h})d\mathbf{h}$$

  or, in the discrete case

$$P(\mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{x}|\mathbf{h})P(\mathbf{h})$$

- The best possible model **x** is the one that uncovers the above "true" structures, with **h** as a latent variable that explains the observed variations in **x**
- The "ideal" representation learned should recover these latent factors
- If **y** is one of them (or closely related to one of them), then it will be very easy to learn to predict **y** from such a representation
- Not knowing which of the factors in **h** will be the one of the interest, say $\mathbf{y} = \mathbf{h}_i$, an unsupervised learning should learn a representation that disentangles all the generative factors $\mathbf{h}_i$ from each other, then making it easy to predict **y** from **h**.

Why is generative model more robust to discriminative model?

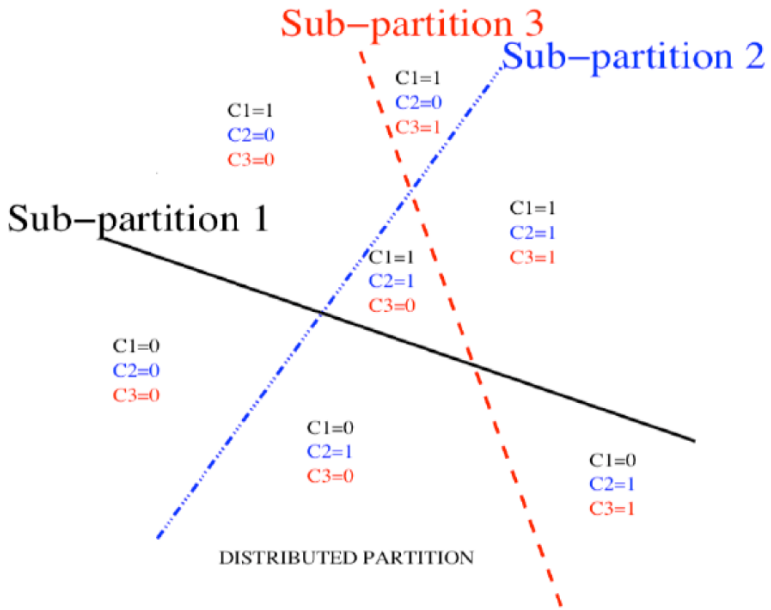
- Assuming **x** is an effect and **y** is a cause, the generative model learns $P(\mathbf{x}|\mathbf{y})$ and estimate $P(\mathbf{y}|\mathbf{x})$ with the Bayes rule

$$P(\mathbf{y}|\mathbf{x}) = \frac{P(\mathbf{x}|\mathbf{y})P(\mathbf{y})}{P(\mathbf{x})}$$

- The discriminative model directly learns $P(\mathbf{y}|\mathbf{x})$
- $P(\mathbf{x}|\mathbf{y})$ is more robust to changes in $P(\mathbf{y})$. To different domains, the causal mechanisms remain invariant ("the laws of the universe are constant") whereas what changes are the marginal distribution over the underlying causes (or what factors are linked to our particular task).
- If the cause-effect relationship was reversed, it would not be true, since by Bayes rule, $P(\mathbf{y}|\mathbf{x})$ would be sensitive to changes in $P(\mathbf{x})$.
- Hence, better generalization and robustness to all kinds of changes can be expected via learning a generative model that attempts to recover the causal factors **h** and $P(\mathbf{x}|\mathbf{h})$

Distributed representation

- The core assumption behind most neural network and deep learning research relies on the notion of distributed representation
- A distributed representation can express an exponentially large number of concepts by allowing to compose the activation of many features
- An example of distributed representation is a vector of n binary features, which can take 2^n configurations, each potentially corresponding to a different region in input space

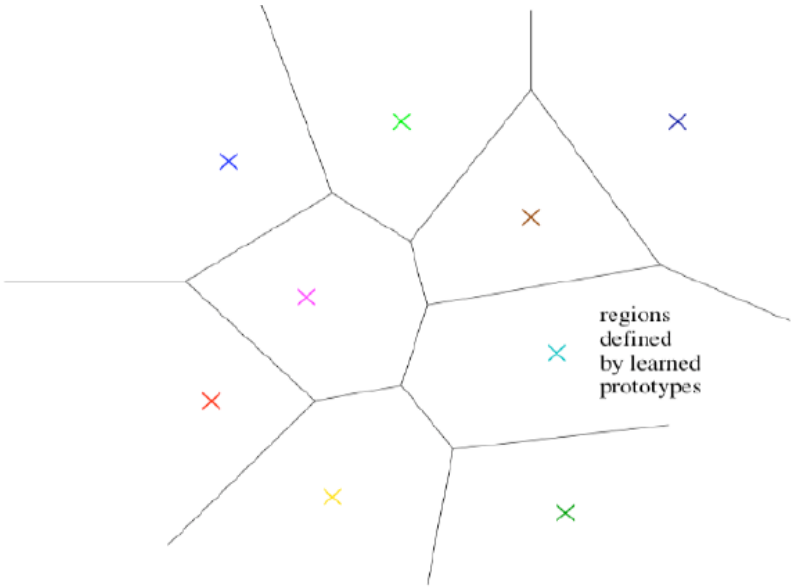


Distributed representation

- The way these regions carve the input space still depends on few parameters: this huge number of regions are not placed independently of each other
- We can thus represent a function that looks complicated but actually has structure
- The assumption is that one can learn about each feature without having to see the examples for all the configurations of all the other features, i.e., these features correspond to underlying factors explaining the data

Non-distributed representation

- Example of symbolic representation: the input is associated with a single symbol or category; only n different configurations of the representation-space are possible, carving n different regions in input space. Such a symbolic representation is also called a one-hot representation, since it can be captured by a binary vector with n bits that are mutually exclusive (only one of them can be active)
- It breaks up the input space into regions, with a separate set of parameters for each region



Examples

- Clustering methods, including the k-means algorithm: only one cluster "wins" the competition
- K-nearest neighbors algorithms: only one template or prototype example is associated with a given input
- Decision trees: only one leaf (and the nodes on the path from root to leaf) is activated when an input is given
- Gaussian mixtures and mixtures of experts: the templates (cluster centers) or experts are now associated with a degree of activation
- Kernel machines with a Gaussian kernel (or other similarly local kernel): the degree of activation of each "support vector" or template example is continuous-valued
- Disadvantages: there is no generalization to new regions, except by extending the answer for which there is data, exploiting solely a smoothness prior; it makes it difficult to learn a complicated function, with more ups and downs than the available number of examples.

Distributed representation leads to better generalization

- Generalization arises due to shared attributes between different concepts
- As pure symbols, “cat” and “dog” are as far from each other as any other two symbols. However, if one associates them with a meaningful distributed representation, then many of the things that can be said about cats can generalize to dogs and vice versa.
- Distributed representations induce a rich similarity space, in which semantically close concepts (or inputs) are close in distance, a property that is absent from purely symbolic representations

Exponential gain in representational efficiency from distributed representations

- A function that "looks complicated" can be compactly represented using a small number of parameters, if some "structure" is uncovered by the learner
- Traditional "non-distributed" learning algorithms generalize only thanks to the smoothness assumption, which states that if $u \approx v$, then the target function f to be learned has the property that $f(u) \approx f(v)$, in general.
- This assumption suffers from the curse of dimensionality: in order to learn a target function that takes many different values (e.g. many ups and downs) in a large number of regions, we may need a number of examples that is at least as large as the number of distinguishable regions.
  - e.g., exponentially many regions: in a d-dimensional space with at least 2 different values to distinguish per dimension, we might want f to differ in 2^d different regions, requiring $O(2^d)$ training examples

Exponential gain in representational efficiency from distributed representations

- One can think of each of these regions as a category or symbol: by having a separate degree of freedom for each symbol (or region), we can learn an arbitrary mapping from symbol to value. However, this does not allow us to generalize to new symbols, new regions.
- If we are lucky, there may be some regularity in the target function, besides being smooth. For example, the same pattern of variation may repeat itself many times (e.g., as in a periodic function or a checkerboard). If we only use the smoothness prior, we will need additional examples for each repetition of that pattern
- A deep architecture could represent and discover such a repetition pattern and generalize to new instances of it. Thus a small number of parameters (and therefore, a small number of examples) could suffice to represent a function that looks complicated (in the sense that it would be expensive to represent with a non-distributed architecture).

Exponential gain in representational efficiency from distributed representations

- How many regions are generated by an arrangement of n hyperplanes in $\mathcal{R}^d$?
- This corresponds to the number of regions that a shallow neural network (one hidden layer) can distinguish
- Therefore, we see a growth that is exponential in the input size and polynomial in the number of hidden units

$$\sum_{j=0}^{d} \binom{n}{j} = O(n^d)$$

- Distributed representation has better generalization because we can learn about the location of each hyperplane with only $O(d)$ examples: we do not need to see examples corresponding to all $O(n^d)$ regions

Exponential gain in representational efficiency from distributed representations

- It is hard to detect factors such as gender, age, and the presence of glasses with a liner classifiers, i.e. a shallow neural network
- The kinds of factors that can be chosen almost independently in order to generate data are more likely to be very high-level and related in highly non-linear ways to the input
- This demands deep distributed representations, where the higher level features (seen as functions of the input) or factors (seen as generative causes) are obtained through the composition of many non-linearities

Exponential gain in representational efficiency from distributed representations

- Organizing computation through the composition of many nonlinearities and a hierarchy of re-used features can give another exponential boost to statistical efficiency
- Although 2-layer networks (e.g., with saturating non-linearities, boolean gates, sum/products, or RBF units) can generally be shown to be universal approximators , the required number of hidden units may be very large
- The main results on the expressive power of deep architectures state that there are families of functions that can be represented efficiently with a deep architecture (say depth k) but would require an exponential number of components (with respect to the input size) with insufficient depth (depth 2 or depth $k-1$)

Example of sum-product network

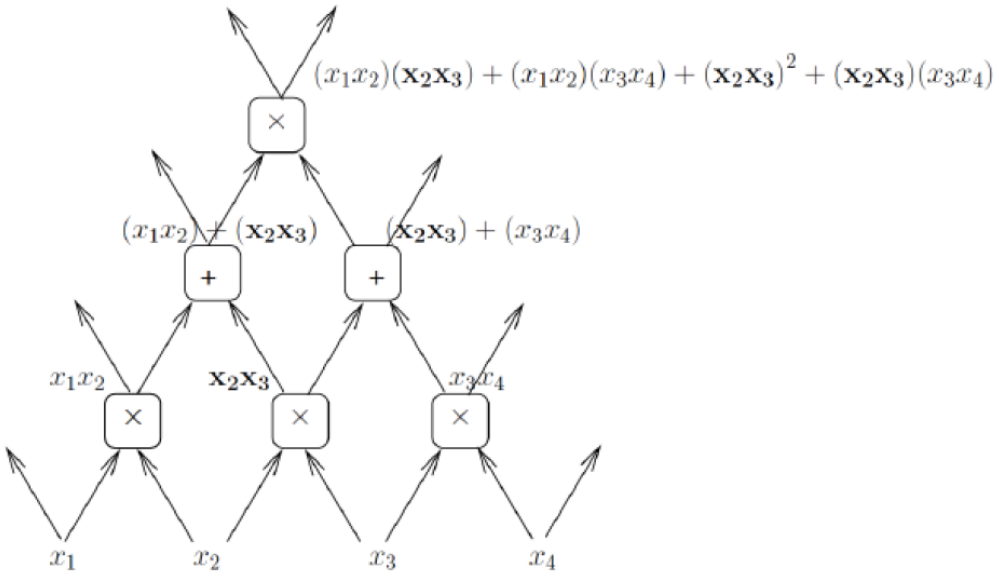


Figure 14.3: A sum-product network (Poon and Domingos, 2011) composes summing units and product units, so that each node computes a polynomial. Consider the product node computing x_2x_3: its value is re-used in its two immediate children, and indirectly incorporated in its grand-children. In particular, in the top node shown the product $x_2\,x_3$ would arise 4 times if that node's polynomial was expanded as a sum of products. That number could double for each additional layer. In general a deep sum of product can represent polynomials with a number of min-terms that is exponential in depth, and some families of polynomials are represented efficiently with a deep sum-product network but not efficiently representable with a simple sum of products, i.e., a 2-layer network (Delalleau and Bengio, 2011).

Example of sum-product network

- More recently, Delalleau and Bengio (2011) showed that a shallow network requires exponentially many more sum-product hidden units than a deep sum-product network (Poon and Domingos, 2011) in order to compute certain families of polynomials.

Example of deep rectifier networks

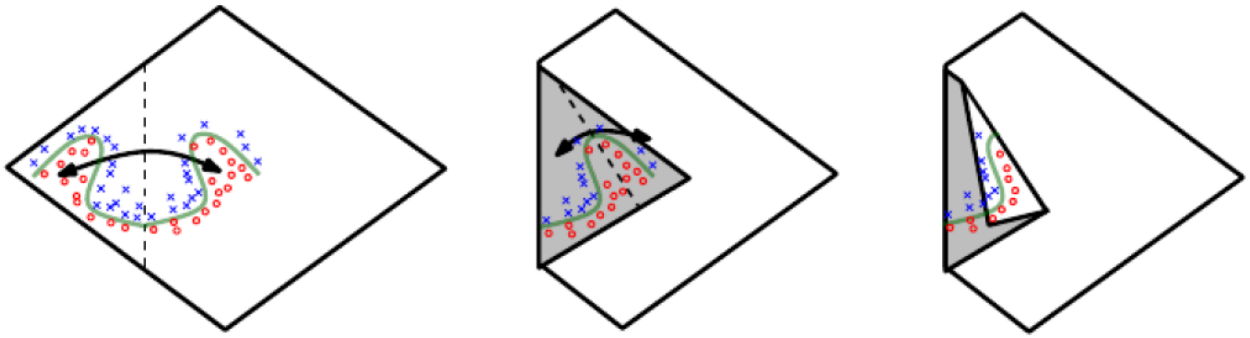


Figure 14.4: An absolute value rectification unit has the same output for every pair of mirror points in its input. The mirror axis of symmetry is given by the hyperplane defined by the weights and bias of the unit. If one considers a function computed on top of that unit (the green decision surface), it will be formed of a mirror image of a simpler pattern, across that axis of symmetry. The middle image shows how it can be obtained by folding the space around that axis of symmetry, and the right image shows how another repeating pattern can be folded on top of it (by another downstream unit) to obtain another symmetry (which is now repeated four times, with two hidden layers). This is an intuitive explanation of the exponential advantage of deeper rectifier networks formally shown in Pascanu *et al.* (2014b); Montufar *et al.* (2014).

Example of deep rectifier networks

- (Pascanu et al., 2014b; Montufar et al., 2014) showed that piecewise linear networks (e.g. obtained from rectifier non-linearities or maxout units) could represent functions with exponentially more piecewise-linear regions, as a function of depth, compared to shallow neural networks.
- Figure illustrates how a network with absolute value rectification creates mirror images of the function computed on top of some hidden unit, with respect to the input of that hidden unit. Each hidden unit specifies where to fold the input space in order to create mirror responses (on both sides of the absolute value non-linearity). By composing these folding operations, we obtain an exponentially large number of piecewise linear regions which can capture all kinds of regular (e.g. repeating) patterns.
- The main theorem in Montufar et al. (2014) states that the number of linear regions carved out by a deep rectifier network with d inputs, depth L, and n units per hidden layer, is

$$O\left(\binom{n}{d}^{d(L-1)} n^d\right)$$

Priors regarding the underlying factors

- *No-free-lunch theorem for machine learning*: with absolutely no priors, it is not possible to generalize
- In the space of all functions, which is huge, with any finite training set, there is no general purpose learning recipe that would dominate all other learning algorithms. Whereas some assumptions are required, when our goal is to build AI or understand human intelligence, it is tempting to focus our attention on the most general and broad priors, that are relevant for most of the tasks that humans are able to successfully learn.
- **Smoothness**: we want to learn functions f s.t. $\mathbf{x} \approx \mathbf{y}$ generally implies $f(\mathbf{x}) \approx f(\mathbf{y})$. This is the most basic prior and is present in most machine learning, but is insufficient to get around the curse of dimensionality, as discussed previously
- **Multiple explanatory factors**: the data generating distribution is generated by different underlying factors, and for the most part what one learns about one factor generalizes in many configurations of the other factors. This assumption is behind the idea of distributed representations

Priors regarding the underlying factors

- **Depth**, or **a hierarchical organization of explanatory factors**: the concepts that are useful at describing the world around us can be defined in terms of other concepts, in a hierarchy, with more abstract concepts higher in the hierarchy, being defined in terms of less abstract ones. This is the assumption exploited by having **deep representations**
- **Causal factors**: the input variables $\mathbf{x}$ are consequences, effects, while the explanatory factors are causes, and not vice-versa. As discussed above, this enables the **semi-supervised learning** assumption, i.e., that $P(\mathbf{x})$ is tied to $P(\mathbf{y}|\mathbf{x})$, making it possible to improve the learning of $P(\mathbf{y}|\mathbf{x})$ via the learning of $P(\mathbf{x})$. More precisely, this entails that representations that are useful for $P(\mathbf{x})$ are useful when learning $P(\mathbf{y}|\mathbf{x})$, allowing sharing of statistical strength between the unsupervised and supervised learning tasks.
- **Shared factors across tasks**: in the context where we have many tasks, corresponding to different $\mathbf{y}_i's$ sharing the same input $\mathbf{x}$ or where each task is associated with a subset or a function $f_i(\mathbf{x})$ of a global input $\mathbf{x}$, the assumption is that each $\mathbf{y}_i$ is associated with a different subset from a common pool of relevant factors $\mathbf{h}$. Because these subsets overlap, learning all the $P(\mathbf{y}_i|\mathbf{x})$ via a shared intermediate representation $P(\mathbf{h}|\mathbf{x})$ allows sharing of statistical strength between the tasks.

Priors regarding the underlying factors

- **Manifolds**: probability mass concentrates, and the regions in which it concentrates are locally connected and occupy a tiny volume. In the continuous case, these regions can be approximated by low-dimensional manifolds that a much smaller dimensionality than the original space where the data lives. This manifold hypothesis is related to auto-encoders.
- **Natural clustering**: different values of categorical variables such as object classes are associated with separate manifolds. More precisely, the local variations on the manifold tend to preserve the value of a category, and a linear interpolation between examples of different classes in general involves going through a low density region, i.e., $P(\mathbf{x}|\mathbf{y} = i)$ for different i tend to be well separated and not overlap much. This hypothesis is consistent with the idea that humans have named categories and classes because of such statistical structure (discovered by their brain and propagated by their culture), and machine learning tasks often involves predicting such categorical variables.

Priors regarding the underlying factors

- **Temporal and spatial coherence**: consecutive or spatially nearby observations tend to be associated with the same value of relevant categorical concepts, or result in a small move on the surface of the high-density manifold. More generally, different factors change at different temporal and spatial scales, and many categorical concepts of interest change slowly. When attempting to capture such categorical variables, this prior can be enforced by making the associated representations slowly changing, i.e., penalizing changes in values over time or space.
- **Sparsity**: for any given observation x, only a small fraction of the possible factors are relevant. In terms of representation, this could be represented by features that are often zero, or by the fact that most of the extracted features are insensitive to small variations of x. This can be achieved with certain forms of priors on latent variables, or by using a non-linearity whose value is often flat at 0 (i.e., 0 and with a 0 derivative), or simply by penalizing the magnitude of the Jacobian matrix (of derivatives) of the function mapping input to representation.

Priors regarding the underlying factors

- Simplicity of Factor Dependencies: in good high-level representations, the factors are related to each other through simple dependencies. The simplest possible is marginal independence, $P(\mathbf{h}) = \prod_i P(\mathbf{h}_i)$, but linear dependencies are also reasonable assumptions. This can be seen in many laws of physics, and is assumed when plugging a linear predictor or a factorized prior on top of a learned representation.

Better mixing via deep representations

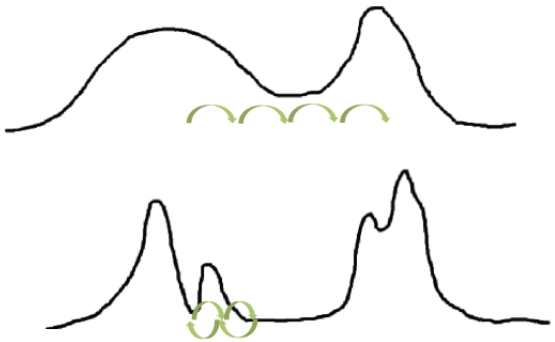
- Why and to what extent different deep learning algorithms may help disentangle underlying factors?
- Better representations can be exploited to produce Markov chains that mix faster between modes
- Mixing between modes would be more efficient at higher levels of representation
- The higher-level samples fill more uniformly the space they occupy and the high-density manifolds tend to unfold when represented at higher levels

Challenge of sampling with MCMC

- Unsupervised learned deep models can be used generate samples
- In general the associated sampling algorithms involve a Markov Chain and MCMC techniques, and these can suffer from a fundamental problem of mixing between modes: it is difficult for the Markov chain to jump from one mode of the distribution to another, when these are separated by large low-density regions, a common situation in real-world data, and under the manifold hypothesis
- This hypothesis states that natural classes present in the data are associated with low-dimensional regions in input space (manifolds) near which the distribution concentrates, and that different class manifolds are well-separated by regions of very low density
- Slow mixing between modes means that consecutive samples tend to be correlated (belong to the same mode) and that it takes many consecutive sampling steps to go from one mode to another and even more to cover all of them, i.e., to obtain a large enough representative set of samples (e.g. to compute an expected value under the target distribution).
- This happens because these jumps through the empty low-density void between modes are unlikely and rare events.

Challenge of sampling with MCMC

- When a learner has a poor model of the data, e.g., in the initial stages of learning, the model tends to correspond to a smoother and higher- entropy (closer to uniform) distribution, putting mass in larger volumes of input space, and in particular, between the modes (or manifolds).
- Keep in mind that MCMCs tend to make moves to nearby probable configurations. Mixing between modes is therefore initially easy for such poor models.
- However, as the model improves and its corresponding distribution sharpens near where the data concentrate, mixing between modes becomes considerably slower. Making one unlikely move (i.e., to a low-probability configuration) may be possible, but making N such moves becomes exponentially unlikely in N



Top: early during training, MCMC mixes easily between modes because the estimated distribution has high entropy and puts enough mass everywhere for small-steps movements (MCMC) to go from mode to mode. Bottom: later on, training relying on good mixing can stall because estimated modes are separated by vast low-density deserts.

Challenge of sampling with MCMC

- Since sampling is an integral part of many learning algorithms (e.g., to estimate the log-likelihood gradient), slower mixing between modes then means slower or poorer learning, and one may even suspect that learning stalls at some point because of the limitations of the sampling algorithm.
- Bengio et al. showed that mixing between modes is easier when sampling at the higher levels of representation
- Deeper generative models produce not only better features for classification but also better quality samples (in the sense of better corresponding to the target distribution being learned)

Hypothesis H1

- **Depth vs Better Mixing Between Modes**: A successfully trained deeper architecture has the potential to yield representation spaces in which Markov chains mix faster between modes.
- If experiments validate that hypothesis, the most im- portant next question is: why?

Sequences of 25 samples generated with a Contractive Auto-Encoder (CAE) on TFD (rows 1 and 2, respectively for 1 or 2 hidden layers) and with an RBM on MNIST (rows 3 and 4, respectively for 1 or 2 hidden layers). On TFD, the second layer clearly allows to get quickly from woman samples (left) to man samples (right) passing by various facial expressions whereas the single hidden layer model shows poor samples. Bottom rows: On MNIST, the single-layer model gets stuck near the same mode while the second layer allows to mix among classes.

Hypothesis H2

- **Depth vs Disentangling**: Part of the explanation of H1 is that deeper representations can better disentangle the underlying factors of variation
- Imagine an abstract (high-level) representation for object image data in which one of the factors is the "reverse video bit", which inverts black and white, e.g., flipping that bit replaces intensity $x \in [0, 1]$ by $1 - x$. With the default value of 0, the foreground object is dark and the background is light. Clearly, flipping that bit does not change most of the other semantic characteristics of the image, which could be represented in other high-level features.
- However, for every image-level mode, there would be a reverse-video counterpart mode in which that bit is flipped: these two modes would be separated by vast "empty" (low density) regions in input space, making it very unlikely for any Markov chain in input space (e.g. Gibbs sampling in an RBM) to jump from one of these two modes to another, because that would require most of the input pixels or hidden units of the RBM to simultaneously flip their value.
- Instead, if we consider the high-level representation which has a "reverse video" bit, flipping only that bit would be a very likely event under most Markov chain transition probabilities, since that flip would be a small change preserving high probability.

Hypothesis H2

- As another example, imagine that some of the bits of the high-level representation identify the category of the object in the image, independently of pose, illumination, background, etc. Then simply flipping one of these object-class bits would also drastically change the raw pixel-space image, while keeping likelihood high. Jumping from an object-class mode to another would therefore be easy with a Markov chain in representation-space, whereas it would be much less likely in raw pixel-space.
- Disentangling cannot be perfect in deep learning. Better disentangling would mean that some of the learned features have a higher mutual information with some of the known factors. One would expect at the same time that the features that are highly predictive of one factor be less so of other factors, i.e., that they specialize to one or a few of the factors, becoming invariant to others.

Hypothesis H3

- **Disentangling Unfolds and Expands.** Part of the explanation of **H2** is that more disentangled representations tend to
  - (a) unfold the manifolds near which raw data concentrates, as well as
  - (b) expand the relative volume occupied by high-probability points near these manifolds
- **H3(a)** says is that disentangling has the effect that the projection of high-density manifolds in the high-level representation space have a smoother density and are easier to model than the corresponding high-density manifolds in raw input space.
- Let us again use an object recognition analogy. If we have perfectly disentangled object identity, pose and illumination, the high-density manifold associated with the distribution of features in high-level representation-space is at: we can interpolate between some training examples (i.e. likely configurations) and yet stay in a high-probability region.
- For example, we can imagine that interpolating between two images of the same object at different poses (lighting, position, etc.) in a high-level representation-space would yield images of the object at intermediate poses (i.e., corresponding to likely natural images), whereas interpolating in pixel space would give a superposition of the two original images (i.e., unlike any natural image).

Hypothesis H3

- If interpolating between high-probability examples (i.e. within their convex set) gives high-probability examples, then it means that the distribution is more uniform (fills the space) within that convex set, which is what **H3(b)** is saying.

we interpolate between samples of different classes, at different depths (top=raw input, middle=1st layer, bottom=2nd layer). Note how in lower levels one has to go through unplausible patterns, whereas in the deeper layers one almost jumps from a high-density region of one class to another (of the other class)

Course materials are from

- Y. Bengio, I. J. Goodfellow, and A. Courville, “Distributed Representations: Disentangling the Underlying Factors” in “Deep Learning” Book in preparation for MIT press, 2014.
- Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, “Better Mixing via Deep Representations” ICML 2013.