DD2437 – Artificial Neural Networks and Deep Architectures (annda)

Lecture 10: **Representation learning and deep generative models I**

Pawel Herman

Computational Science and Technology (CST)

KTH Royal Institute of Technology

Recap on deep learning

- What is the motivation for deep network architectures?
  - expressive power (*expressibility*) and compactness (*efficiency*) – exponential gain
  - hierarchical brain (cortex) organisation
  - multiple levels of abstraction
  - multiple levels of representations suitable for multi-task learning
- *Learning data representations* in deep learning approach vs *hand-engineering features* in traditional pattern recognition
- Learning protocol for DBNs, stacked autoencoders:
  - PHASE I: greedy layer-wise unsupervised pre-training (autoencoders or RBMs)
  - PHASE II: supervised tuning with gradient descent-like optimisation (the last layers or the entire network)

Recap on deep learning

- Hypotheses about the role of unsupervised pre-training: *regularisation* vs *optimisation* hypotheses
- However, currently there is a trend to avoid pre-training and employ ReLU units (less risk for overfitting and local minima) and new regularisation approaches, e.g. dropout, batch normalisation
- What does DL have to offer?
  - learning data representations
  - hierarchy of distributed features (multi-task and transfer learning, non-local generalisation, mitigating the effect and consequences of curse of dimensionality)
  - good performance (large-scale problems) with relatively compact models
  - semi-supervised learning opportunities

Recap on deep learning

- Why does DL works so well?
  - “cheap learning”
  - ”no-flattening” theorems
  - hierarchical structure of the physical world
- Still plenty of challenges ahead!

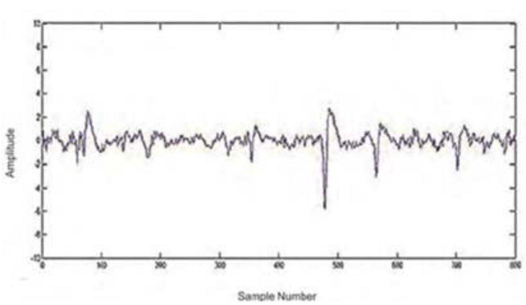
Lecture overview

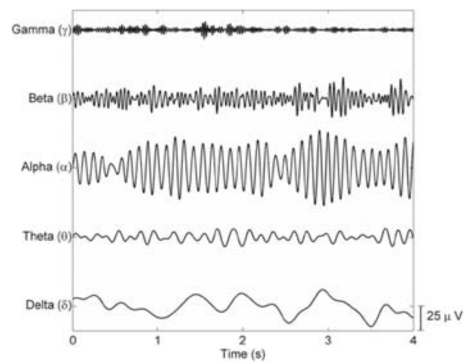
1. Data representations: desirable characteristics and key concepts.
2. Restricted Boltzmann machine (RBM) and Contrastive Divergence (CD) learning.
3. Autoencoders.
4. Deep neural network models for learning representations.

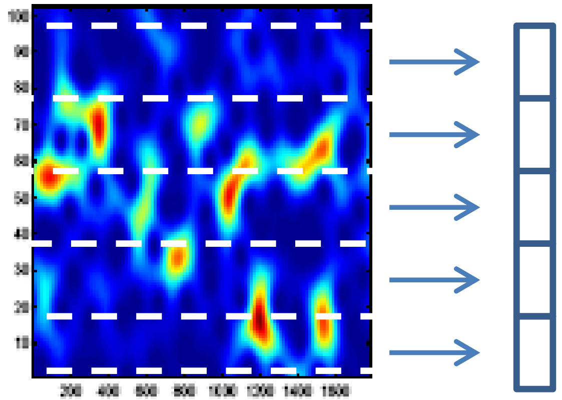
Data representations

- Multiple ways of representing information – what is the difference? Why should we care?

From low-level data description to higher-order representations







feature vector

How are attributes/features determined, extracted?

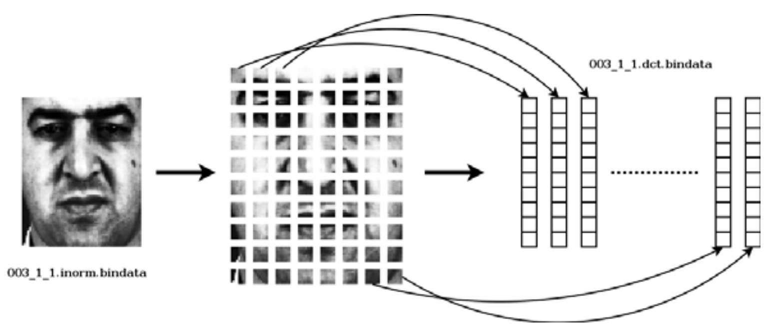
Data representations

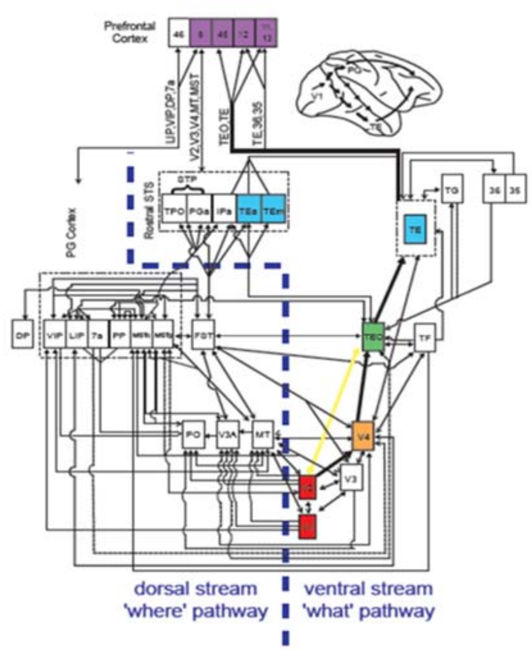
- Multiple ways of representing information – what is the difference? Why should we care?

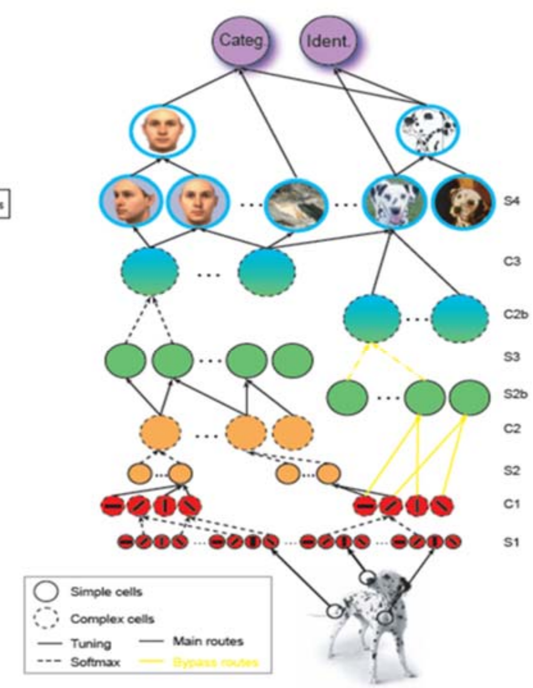
Data parameterisation

12102 → 10111101000110 (bin)

12102 → 2f46 (hex)



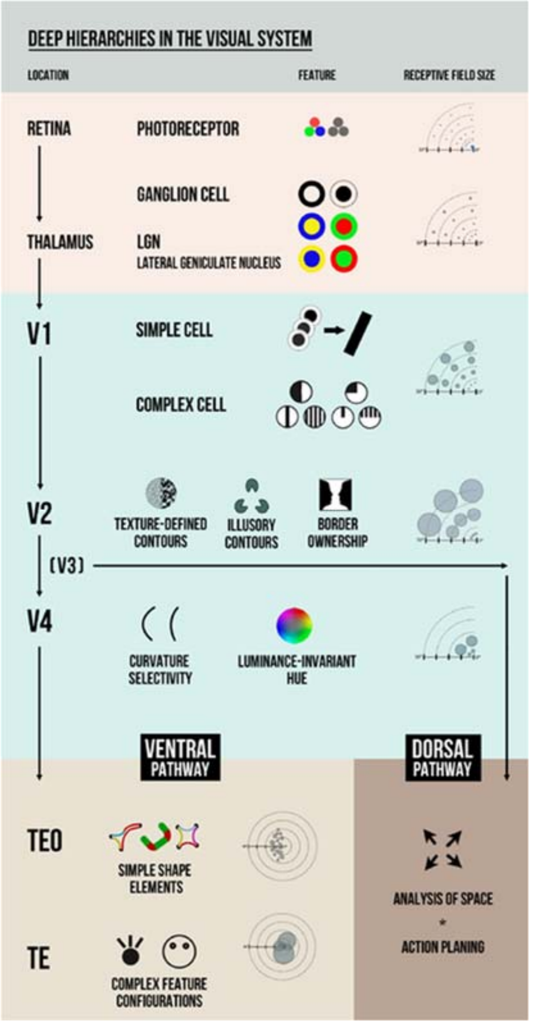




Hypothetical hierarchical representations of visual objects in the brain

Representations in the brain

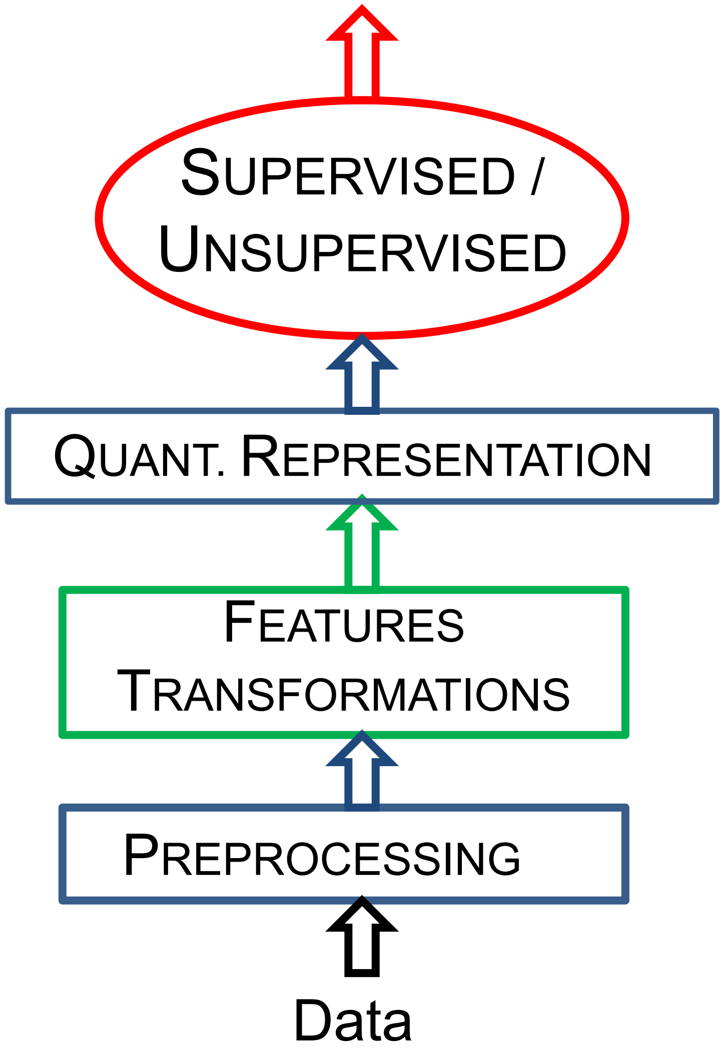
- Sensory information represented by neural activity
  - neurons with different response properties (selectivity, tuning curves)
  - *distributed* nature of neural representations in populations vs grandmother cell concept
  - *sparseness*, redundancy
- Hierarchical representations
  - sensory pathways are organised into *hierarchies*
  - hierarchy of *abstraction* levels



Wikibooks

Representation learning problem

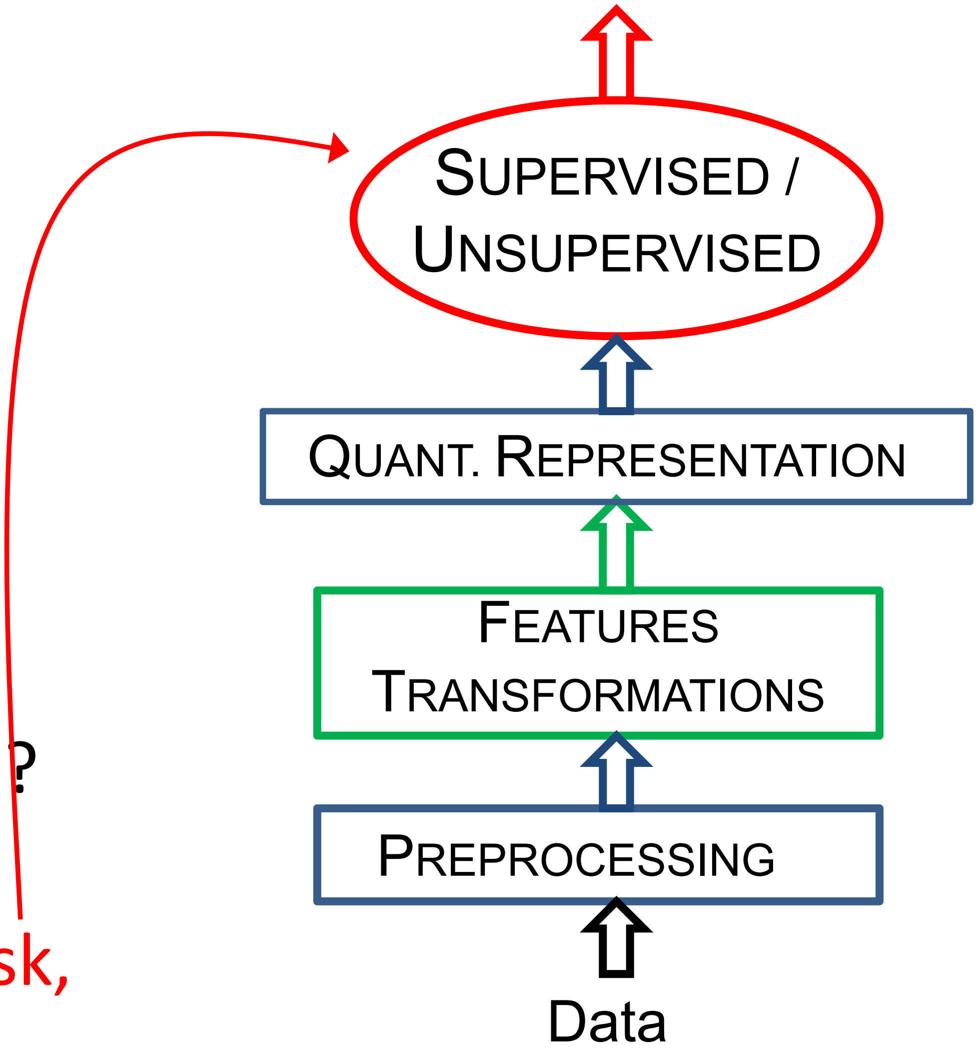
- Importance in the machine learning or pattern recognition context
- there is a trade-off between minimising “information” loss and obtaining “nice” properties
- What makes representation good? What is desirable/useful information?



Representation learning problem

- Importance in the machine learning or pattern recognition context
- there is a trade-off between minimising “information” loss and obtaining “nice” properties
- What makes representation good? What is desirable/useful information?

Facilitate the subsequent learning task, maximise its performance



Representation learning

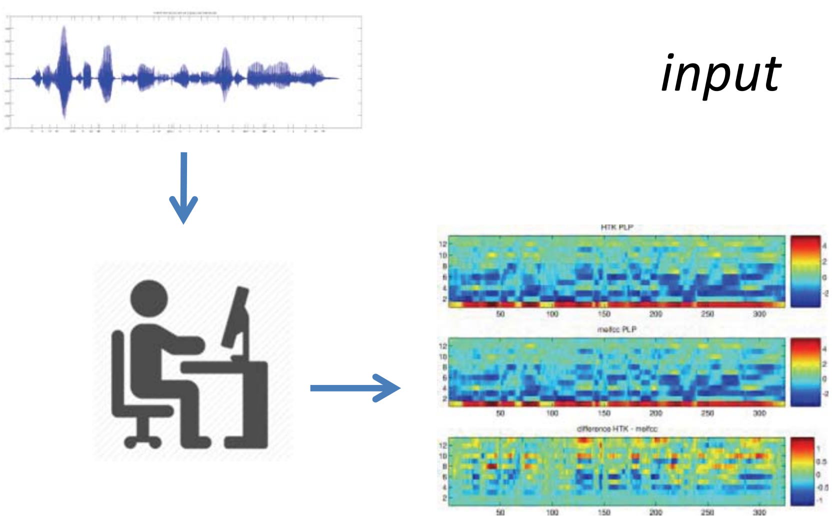
- Computational perspective: disentangling unknown factors causing relevant variation in the data
  - *causes explain* the observed data (discriminative context, both unsupervised and supervised aspects)
  - factors in combination can be used to generate data (generative context)
- Probabilistic perspective
  - *density estimation* – learn probability distribution for data with the use of latent variables (PCA, ICA etc.)
  - P(data|latent var) for recognition and P(latent var|data) for generation

Representation learning in deep models

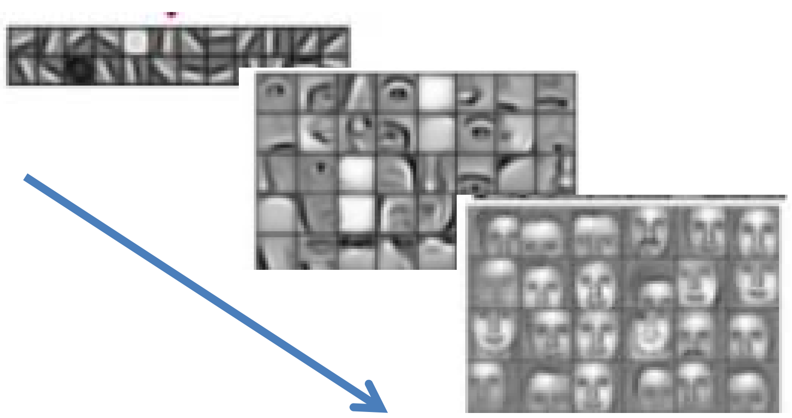
- The composition of multiple non-linear transformations with the expectation for the hierarchy of abstraction levels

Hand-engineered features in a traditional pattern recognition approach

End-to-end networks with learned features spaces, data representations



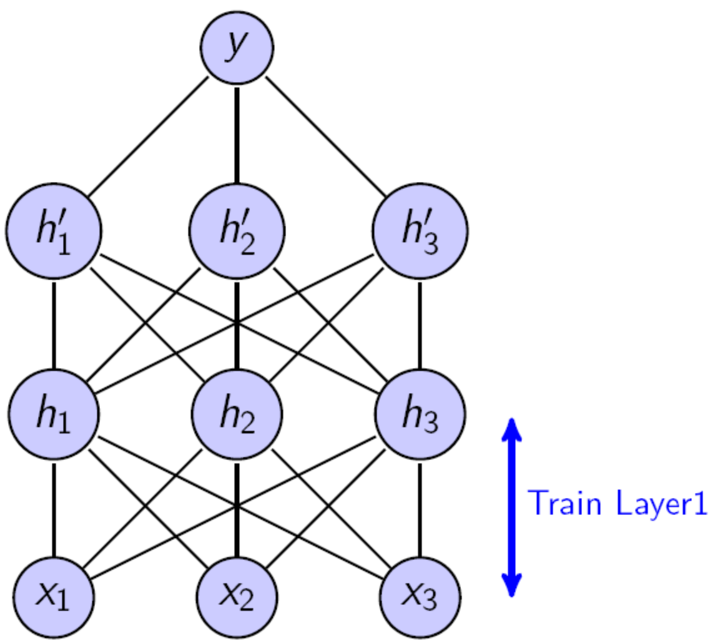
input



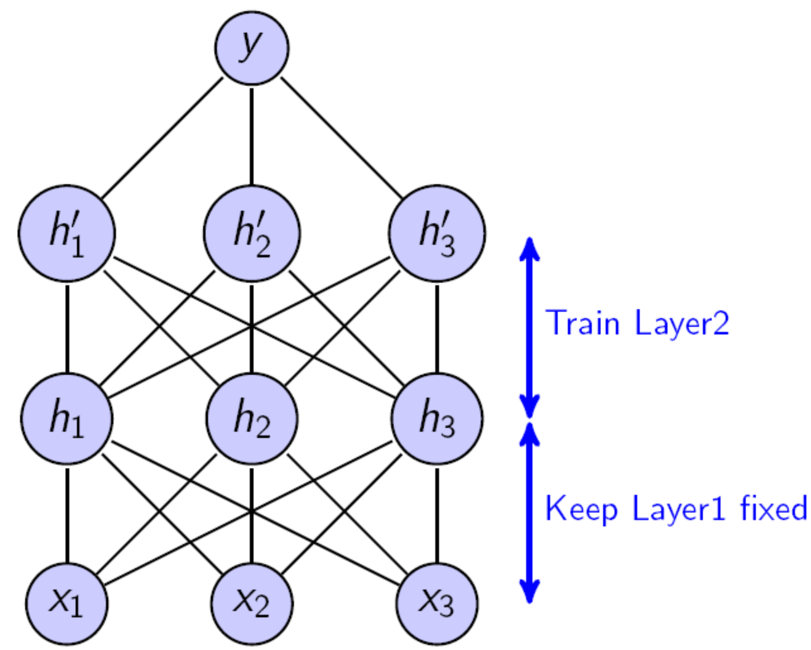
features, representations

Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised learning



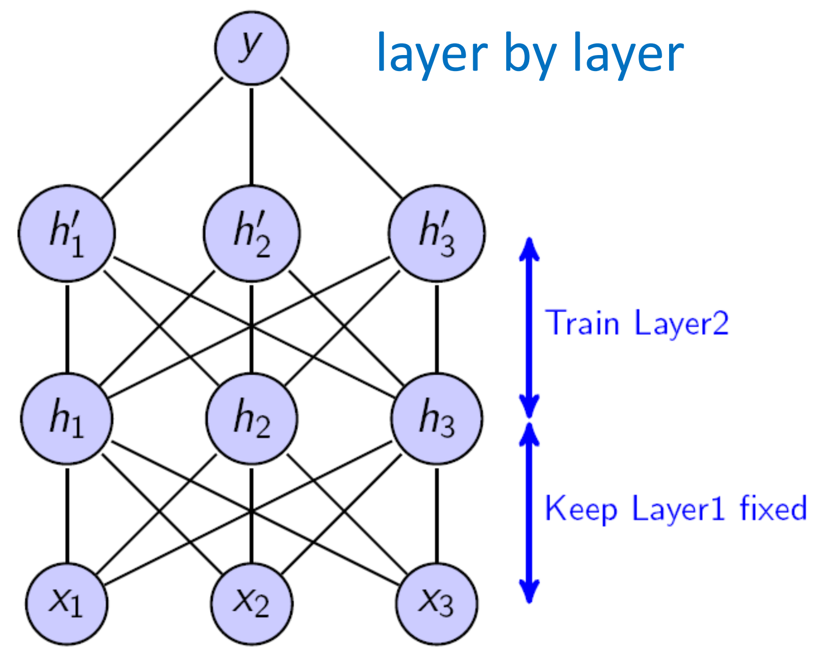
Single layer at a time



Train another layer while keeping the lower layer fixed

Representation learning in deep models

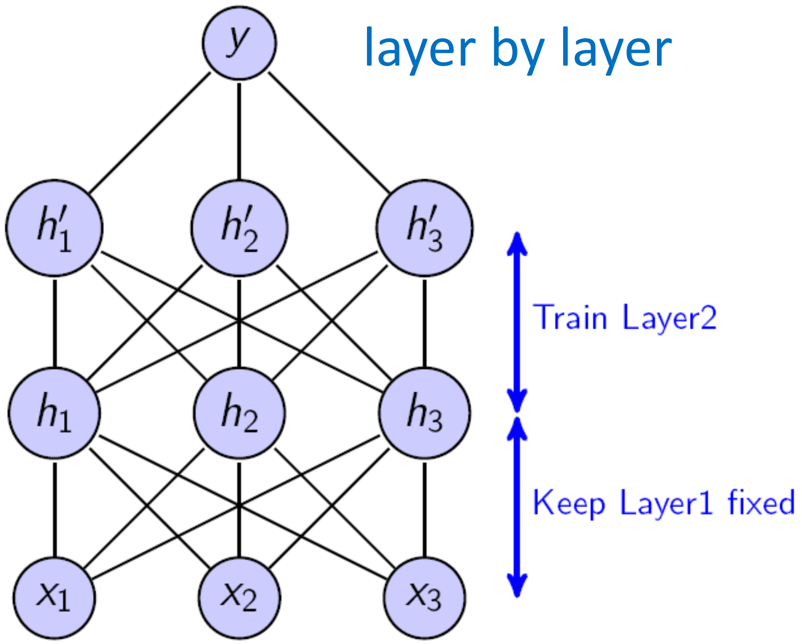
- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - restricted BM (RBM), autoencoders
    - leads to lower test classification error
    - pretraining as an initialisation scheme
      - prior to supervised fine-tuning
      - starting from / accessing the regions often hard to access by means of gradient descent
      - initialisation for other unsupervised algorithms such as DBM, DBN etc.
      - esp. useful when the number of unlabeled samples is much higher than labelled data



Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - restricted BM (RBM), autoencoders
    - leads to lower test classification error
    - pretraining as an initialisation scheme
    - *optimisation* vs *regularisation* hypothesis
      - lower variance in learning, less risk for overfitting (reaching similar minima)
      - as a regulariser, it urges the learning algorithm to discover features that explain *underlying causes* that generate the data (also, causal factors often remain *invariant*)
      - unlike weight decay, there is no bias towards simpler solutions

layer by layer

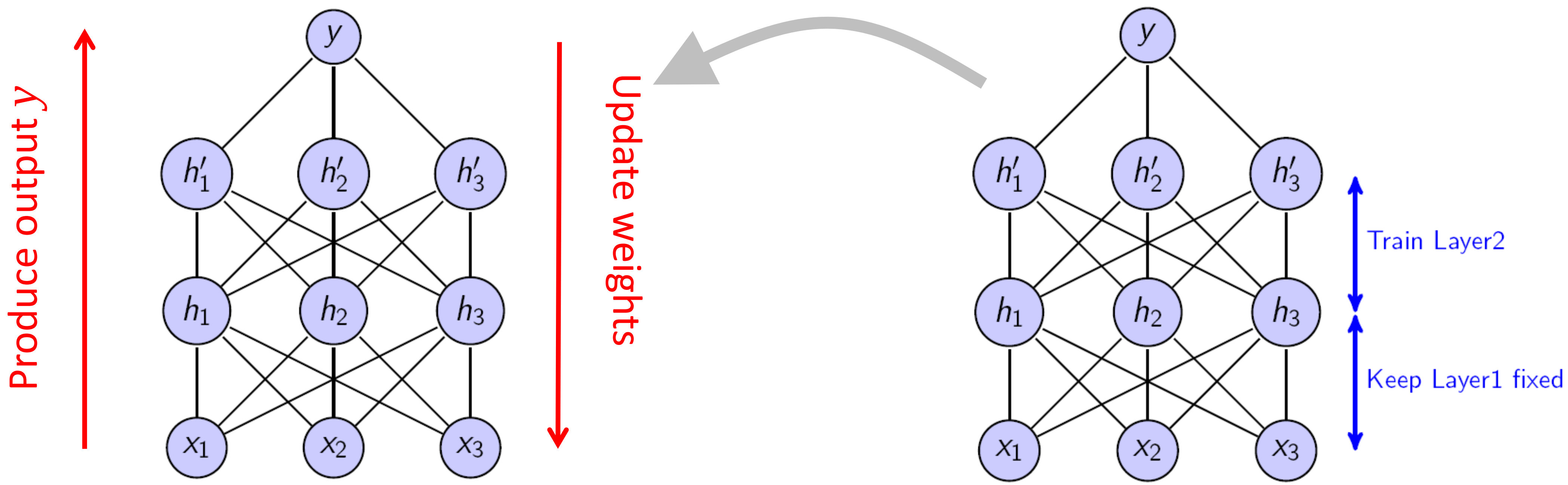


Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - intuitively, learning about the input distribution should help in learning the *mapping* between the input and output space

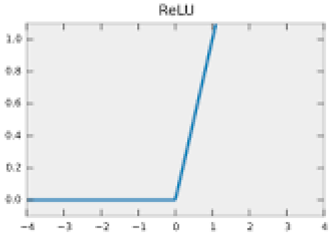
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - intuitively, learning about the input distribution should help in learning the *mapping* between the input and output space
    - BUT having two separate phases has *disadvantages*



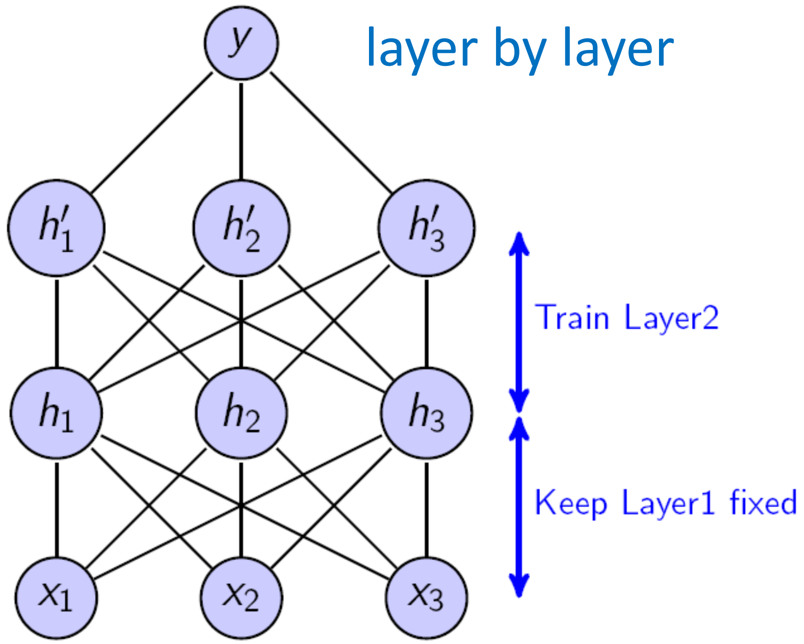
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - intuitively, learning about the input distribution should help in learning the *mapping* between the input and output space
    - BUT having two separate phases has *disadvantages*
  - ULTIMATELY, the approach with <u>unsupervised pretraining</u> is largely ***abandoned*** (except word embeddings in NLP)
    - new regularisation techniques: dropout, batch normalisation
    - smaller datasets -> Bayesian methods
    - units with ReLU activation



Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning
    - restricted BM (RBM), autoencoders
    - leads to lower test classification error
    - pretraining as an initialisation scheme
      - prior to supervised fine-tuning
      - initialisation for other unsupervised algorithms such as DBM, DBN etc.
    - *optimisation* vs *regularisation* hypothesis
      - lower variance in learning, less risk for overfitting
      - as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)



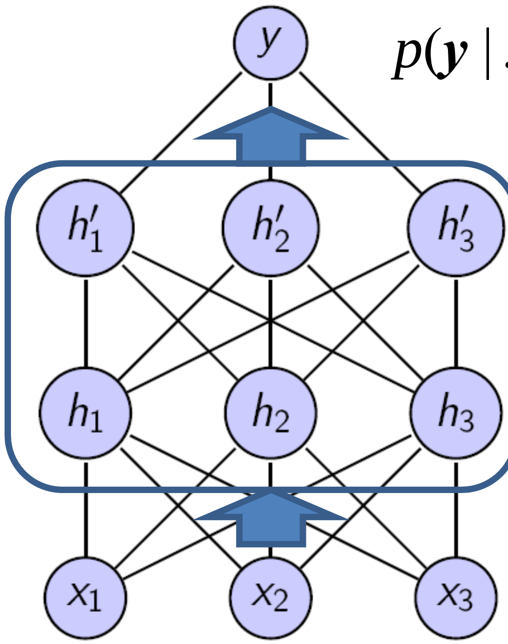
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - ➢ greedy layer-wise unsupervised representation learning

Representation learning should strive towards uncovering latent factors, $\boldsymbol{h}$, which capture underlying causes in $\boldsymbol{x}$.

Then, if $\boldsymbol{y}$ is one of them, i.e. $\boldsymbol{y}=h_i$, it should be easy to learn to predict $\boldsymbol{y}$ from this representation.

$p(\boldsymbol{y} \mid \boldsymbol{x})$



$p(\boldsymbol{h}|\boldsymbol{x})=p(\boldsymbol{x}|\boldsymbol{h})\ p(\boldsymbol{h})$

Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

marginal factor independence

$p(\boldsymbol{x})$

erfitting

– as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

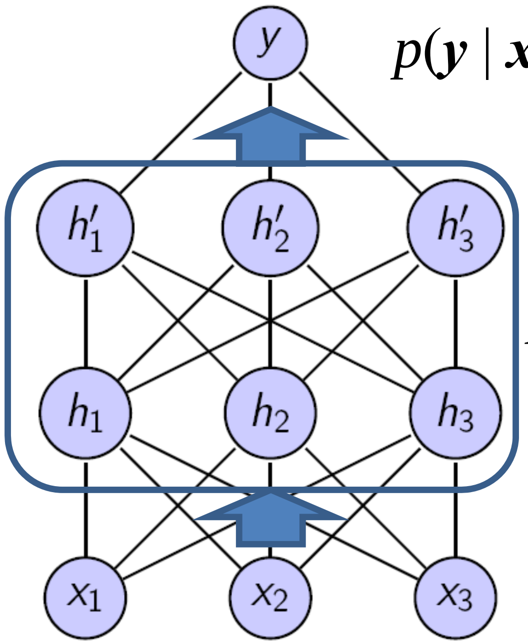
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning

Representation learning should strive towards uncovering latent factors, $\boldsymbol{h}$, which capture underlying causes in $\boldsymbol{x}$.

Then, if $\boldsymbol{y}$ is one of them, i.e. $\boldsymbol{y}=h_i$, it should be easy to learn to predict $\boldsymbol{y}$ from this representation.

<u>So, how to make representation encode relevant/salient factors?</u>



Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

marginal factor independence

erfitting

– as a regulariser, it urges the learning algorithm to discover **features that explain** ***underlying causes*** **that generate the data** (also, causal factors often remain *invariant*)

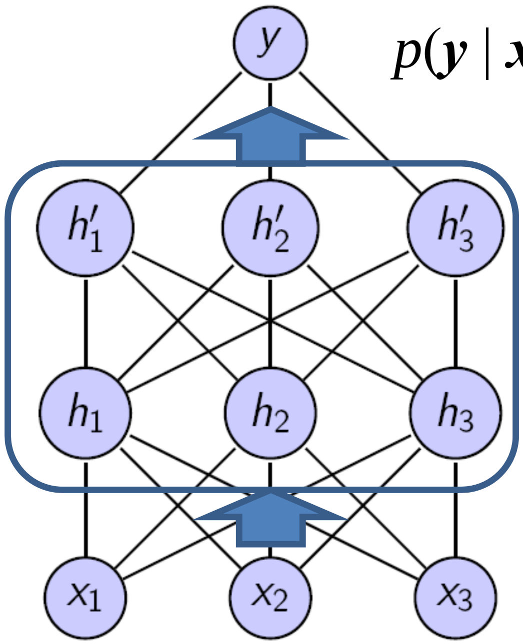
Representation learning in deep models

- The concept of layer-by-layer pretraining
  - greedy layer-wise unsupervised representation learning

So, how to make representation encode relevant/salient factors?

1) Guide unsupervised pretraining with a supervised learning signal.
2) Rely on massive representations with purely unsupervised learning.
3) Redefine the definition of salience.

erfitting



$p(\boldsymbol{y} \mid \boldsymbol{x})$

$p(\boldsymbol{h}|\boldsymbol{x})$=$p(\boldsymbol{x}|\boldsymbol{h})\ p(\boldsymbol{h})$

Ideally:

$p(\boldsymbol{h}) = \prod_i p(h_i)$

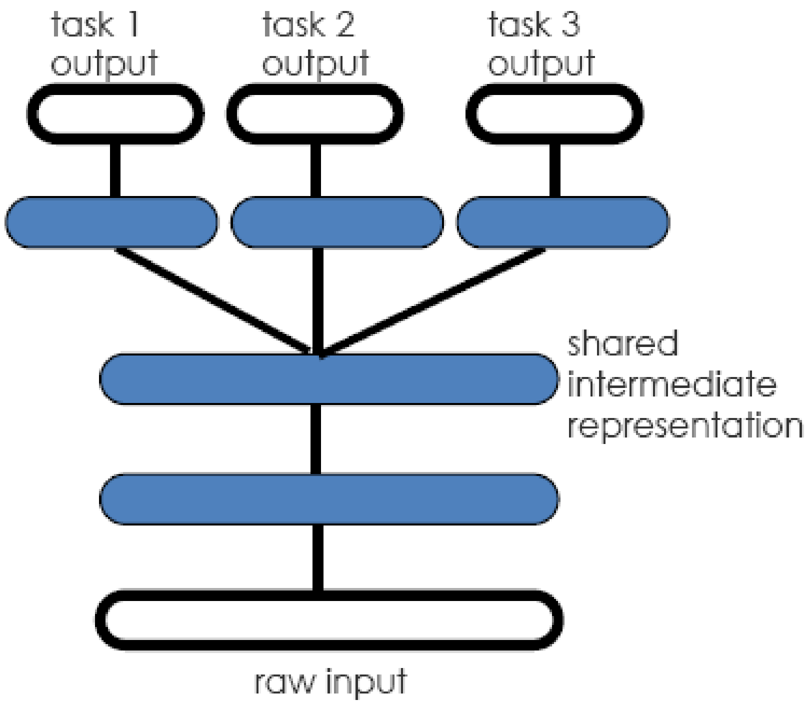
marginal factor independence

$p(\boldsymbol{x})$

– as a regulariser, it urges the learning algorithm to discover **features that explain *underlying causes* that generate the data** (also, causal factors often remain *invariant*)

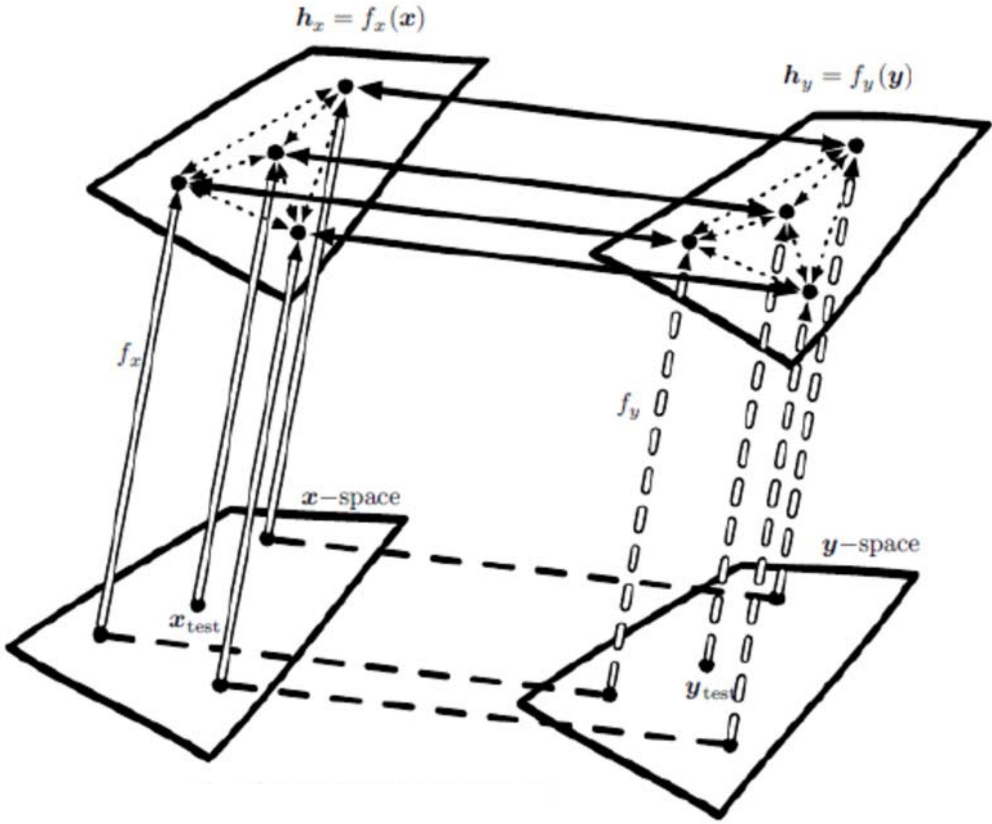
Transfer learning – sharing factors across tasks

- Assumption that factors explaining the variations in different tasks are shared/common
- Especially low-level features are expected to be the same
- The concept of *one-shot* learning



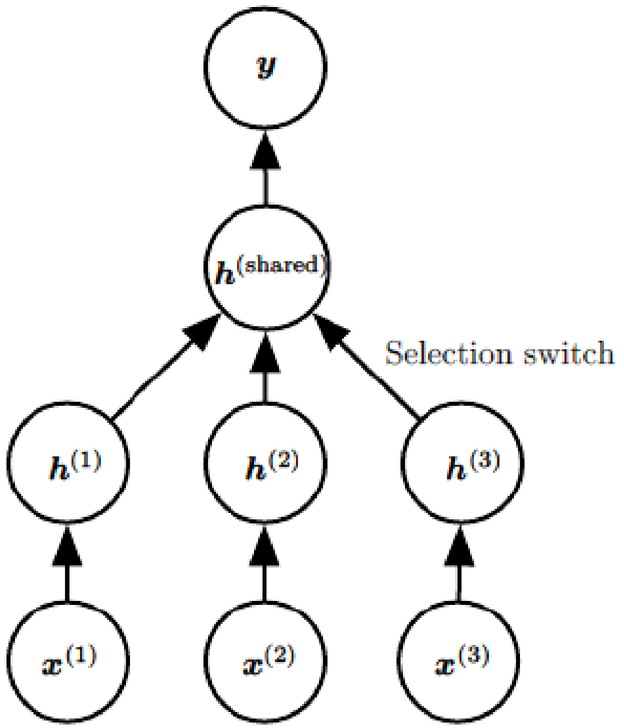
Transfer learning – sharing factors across tasks

- Assumption that factors explaining the variations in different tasks are shared/common
- Especially low-level features are expected to be the same
- The concept of *one-shot* learning
- Zero-shot learning as a specific form of *multi-modal learning* (capturing the relationship between representations in different modalities)



Transfer learning – sharing factors across tasks

- Assumption that factors explaining the variations in different tasks are shared/common
- Especially low-level features are expected to be the same
- The concept of *one-shot* learning
- Zero-shot learning as a specific form of *multi-modal learning* (capturing the relationship between representations in different modalities)
- However, sometimes the semantics of the output is shared instead, which requires *domain adaptation*



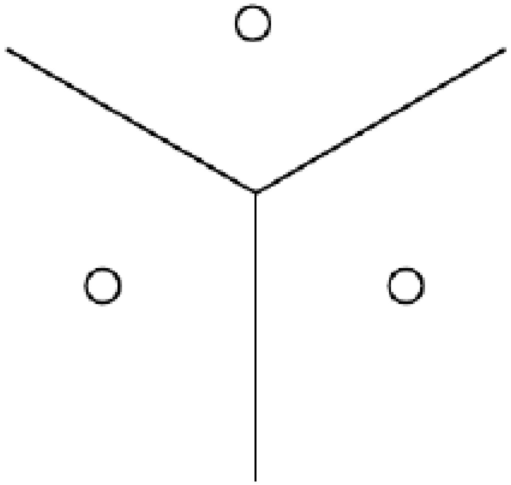
Distributed representations

Information is distributed across many units that account for information about features that are not mutually exclusive…..

Distributed representations

Information is distributed across many units that account for information about features that are not mutually exclusive…..

… unlike in clustering with distinct regions where *local generalisation* is observed.



Distributed representations

Information is distributed across many units that account for information about features that are not mutually exclusive…..

… unlike in clustering with distinct regions where *local generalisation* is observed.

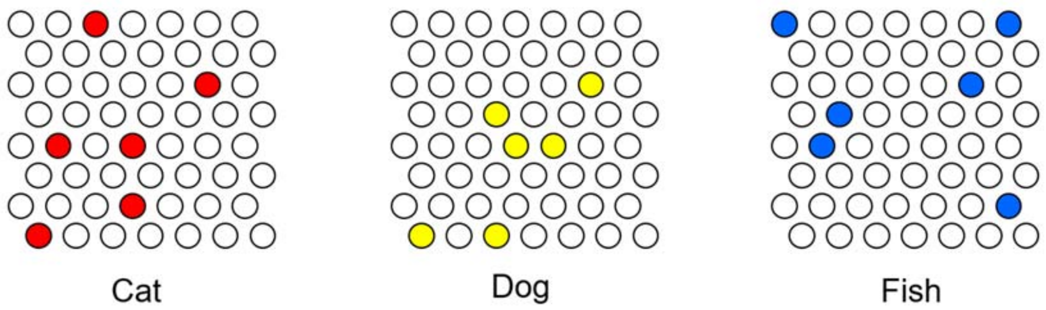
Locality in input space implies different behaviour of the learned function in different regions of data space (local or symbolic representations).

Distributed representations

Information is distributed across many units that account for information about features that are not mutually exclusive…..

… unlike in clustering with distinct regions where *local generalisation* is observed.

Locality in input space implies different behaviour of the learned function in different regions of data space (local or symbolic representations).



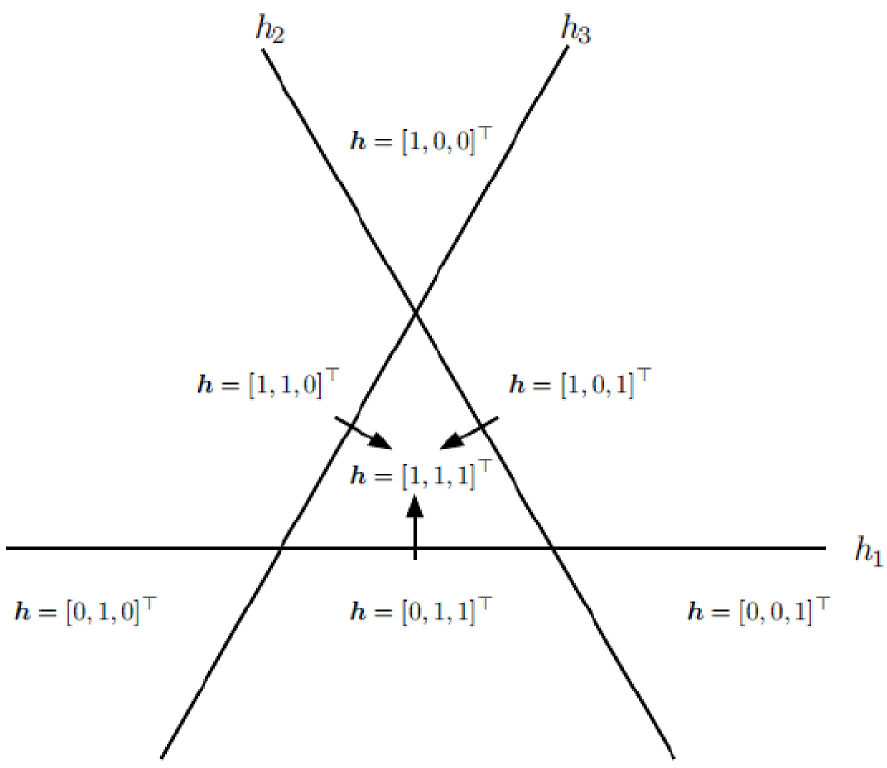
Generalisation due to shared attributes and semantic proximity.

The power of distributed representations

In summary:

- expressiveness (n features with k values each can describe k^n concepts)
- the combination of powerful distributed representations with weak classifiers could be a strong regulariser

fault tolerance



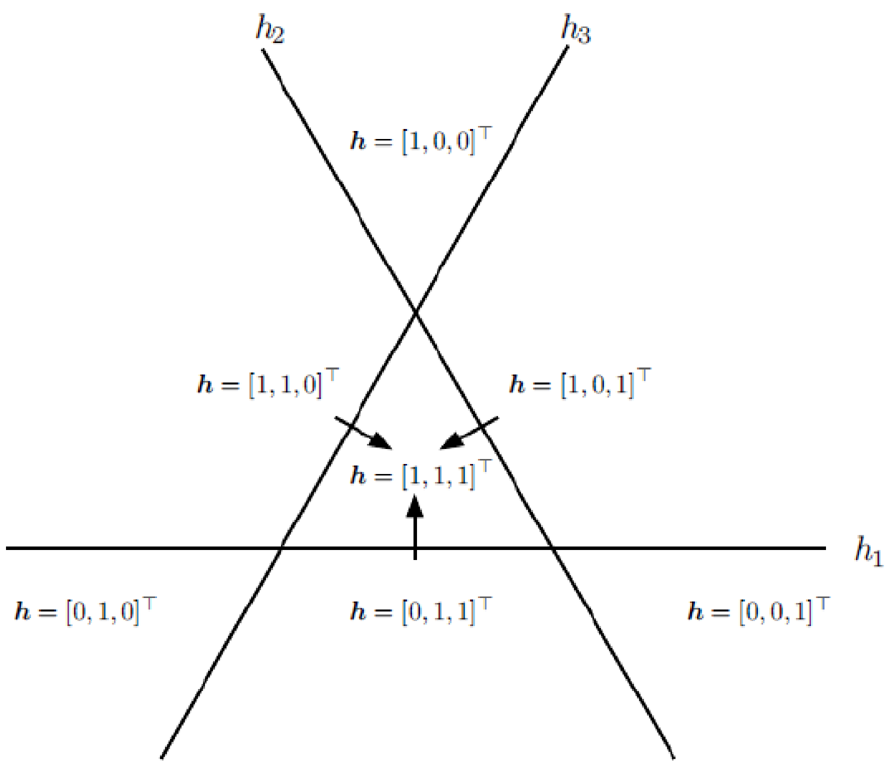
Goodfellow et al.
Bengio et al., 2009

The power of distributed representations

In summary:

- expressiveness (*n* features with *k* values each can describe k^n concepts)
- the combination of powerful distributed representations with weak classifiers could be a strong regulariser
- similarity (topological) space with a distributed code – semantically close objects are close in distance
- generalisation due to shared attributes

content addressibility

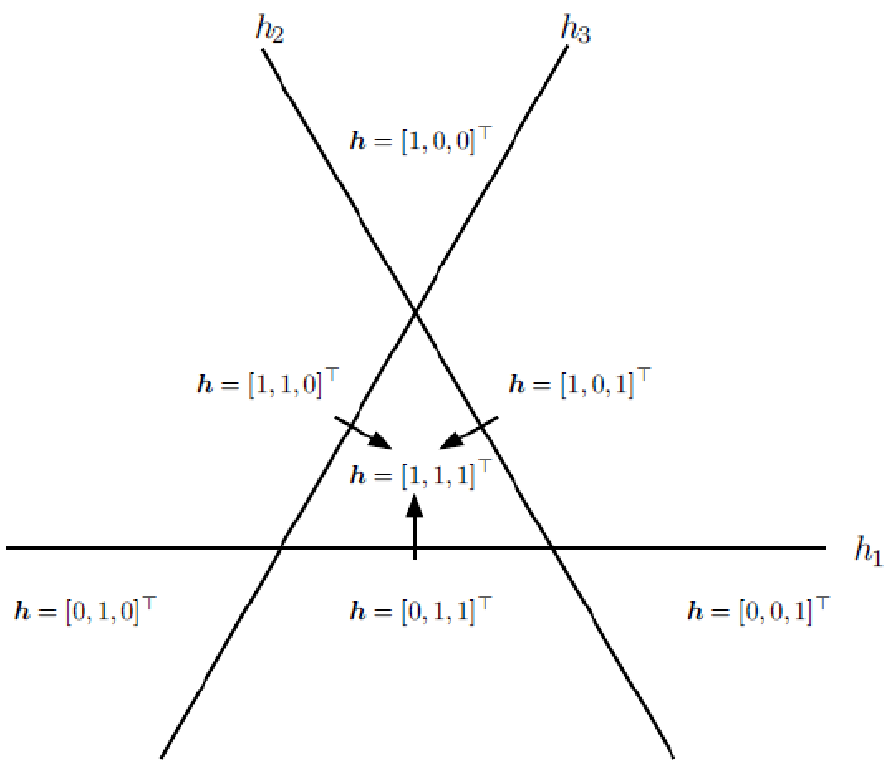


Goodfellow et al.
Bengio et al., 2009

The power of distributed representations

In summary:

- expressiveness (n features with k values each can describe k^n concepts)
- the combination of powerful distributed representations with weak classifiers could be a strong regulariser
- similarity (topological) space with a distributed code – semantically close objects are close in distance
- generalisation due to shared attributes
- in line with the idea that hidden units can learn to represent the underlying causal factors as different variables (here: directions in the representation space)



Goodfellow et al.
Bengio et al., 2009

Sparse vs dense representations

Sparse representations

- promoting memory capacity
- orthogonalisation/decorrelation
- "metabolic" efficiency
- neural selectivity (vs coarse coding with broad tuning)

sparse not distributed	not sparse distributed	sparse distributed
0 .2 0 0 0	.1 .8 .7 .5 .7	0 .8 0 .5 0
0 0 0 0 .1	.8 .9 .6 .2 .4	0 0 .6 0 .4
0 0 0 .4 0	.3 .1 .6 .3 .3	.3 0 0 .3 0

Generic regularisation strategies (Bengio et al., 2013)

How can the discovery/identification of the underlying causal factors of variation that generates the data be further supported?

(beyond application specific supervision signal)

Generic regularisation strategies (Bengio et al., 2013)

How can the discovery/identification of the underlying causal factors of variation that generates the data be further supported?

(beyond application specific supervision signal)

hierarchy of explanatory factors

multiple explanatory (and ideally, *causal*) factors

(semisupervised/unsupervised pretraining + distributed representations to untangle separate factors of variation in the representation space)

Transfer learning, multi-task learning

shared factors across tasks

Generic regularisation strategies (Bengio et al., 2013)

How can the discovery/identification of the underlying causal factors of variation that generates the data be further supported?

(beyond application specific supervision signal)

smoothness in the generalisation

coherence in space and time

hierarchy of explanatory factors

low-dim manifolds and class-dependent natural clustering

multiple explanatory (and ideally, *causal*) factors

(semisupervised/unsupervised pretraining + distributed representations to untangle separate factors of variation in the representation space)

shared factors across tasks

Generic regularisation strategies (Bengio et al., 2013)

How can the discovery/identification of the underlying causal factors of variation that generates the data be further supported?

(beyond application specific supervision signal)

smoothness in the generalisation

linearity

multiple explanatory (and ideally, *causal*) factors

(semisupervised/unsupervised pretraining + distributed representations to untangle separate factors of variation in the representation space)

hierarchy of explanatory factors

coherence in space and time

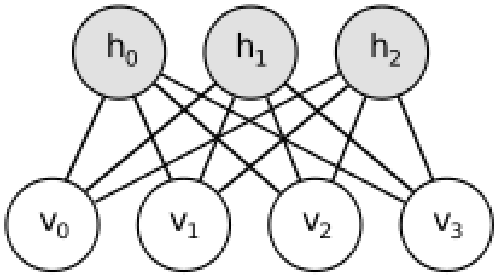
low-dim manifolds and class-dependent natural clustering

sparse and distributed

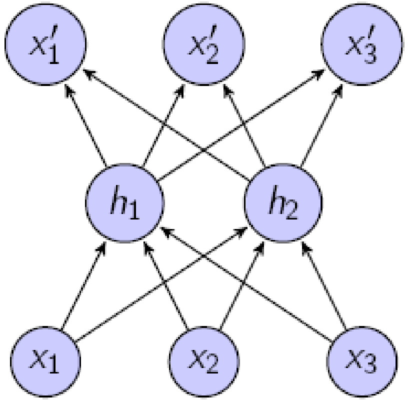
shared factors across tasks

Computational blocks for learning representations

- Two key approaches to greedy layer-wise pretraining
  - regularized Boltzmann machines (RBMs)



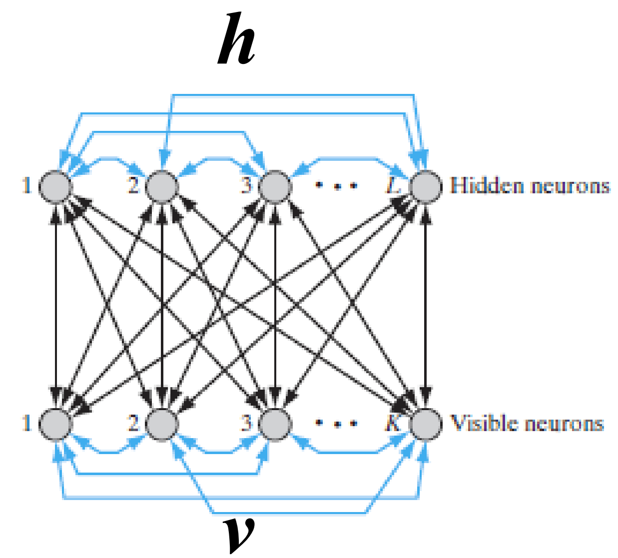
  - autoencoders



Recap on Boltzmann machine

$$E = -\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}x_i x_j$$

$$P(\vec{x}\,|\,\mathbf{W}) = \frac{e^{-E}}{Z} = \frac{1}{Z(\mathbf{W})}\exp\left(\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x}\right)$$

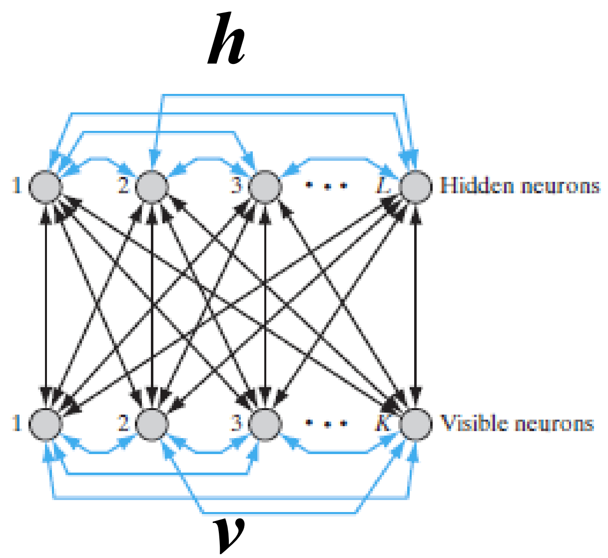


$$\boldsymbol{v}^{(p)} = \boldsymbol{x}^{(p)}$$
$$\Downarrow$$
$$\boldsymbol{y}^{(p)} = [\boldsymbol{x}^{(p)}, \boldsymbol{h}]$$

Recap on Boltzmann machine

$$E = -\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}x_i x_j$$

$$P(\vec{x} \mid \mathbf{W}) = \frac{e^{-E}}{Z} = \frac{1}{Z(\mathbf{W})}\exp\left(\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x}\right)$$



$$\boldsymbol{v}^{(p)} = \boldsymbol{x}^{(p)}$$
$$\Downarrow$$
$$\boldsymbol{y}^{(p)} = [\boldsymbol{x}^{(p)}, \boldsymbol{h}]$$

The idea is to maximise log-likelihood,

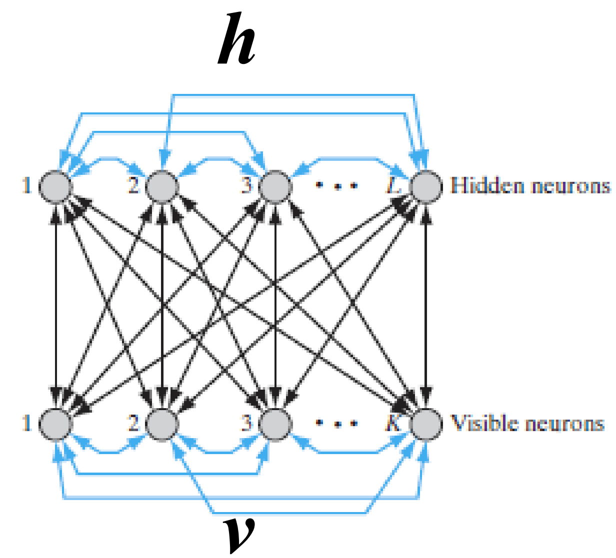
$$L(\mathbf{W}) = \log\left(P(\mathbf{X})|\mathbf{W}\right)$$

$$\Delta w_{ji} = \varepsilon\frac{\partial L(\mathbf{W})}{\partial w_{ji}} \propto \left\langle y_i y_j \right\rangle_{\text{data}} - \left\langle y_i y_j \right\rangle_{\text{model}}$$

Recap on Boltzmann machine

$$E = -\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}x_i x_j$$

$$P(\vec{x}\,|\,\mathbf{W}) = \frac{e^{-E}}{Z} = \frac{1}{Z(\mathbf{W})}\exp\left(\frac{1}{2}\vec{x}^{\mathrm{T}}\mathbf{W}\vec{x}\right)$$



$$\boldsymbol{v}^{(p)} = \boldsymbol{x}^{(p)}$$

$$\Downarrow$$

$$\boldsymbol{y}^{(p)} = [\boldsymbol{x}^{(p)}, \boldsymbol{h}]$$

The idea is to maximise log-likelihood,

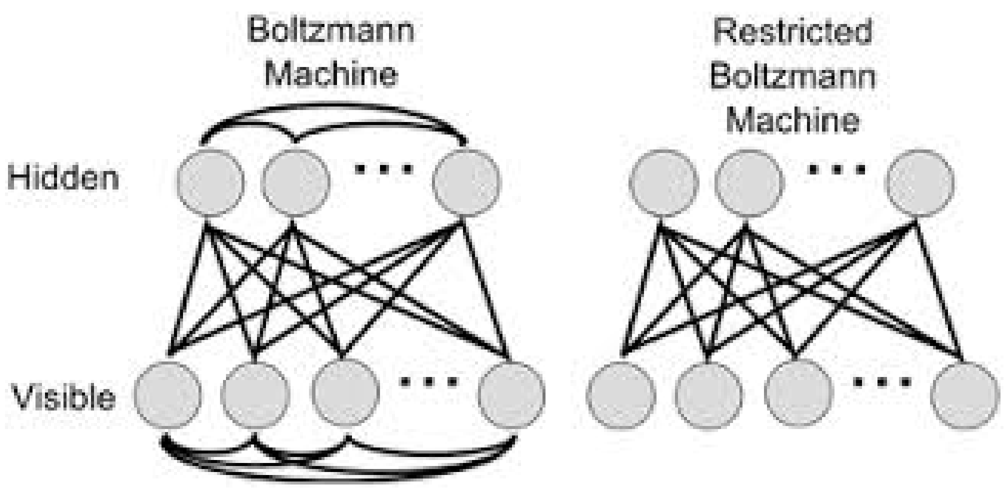
$$L(\mathbf{W}) = \log\left(P(\mathbf{X})|\mathbf{W}\right)$$

$$\Delta w_{ji} = \varepsilon\frac{\partial L(\mathbf{W})}{\partial w_{ji}} \propto \boxed{\langle y_i y_j \rangle_{\text{data}}} - \boxed{\langle y_i y_j \rangle_{\text{model}}}$$

positive: *"Hebbian learning"*

negative: *"Hebbian forgetting"*

Restricted Boltzmann machine (RBM)

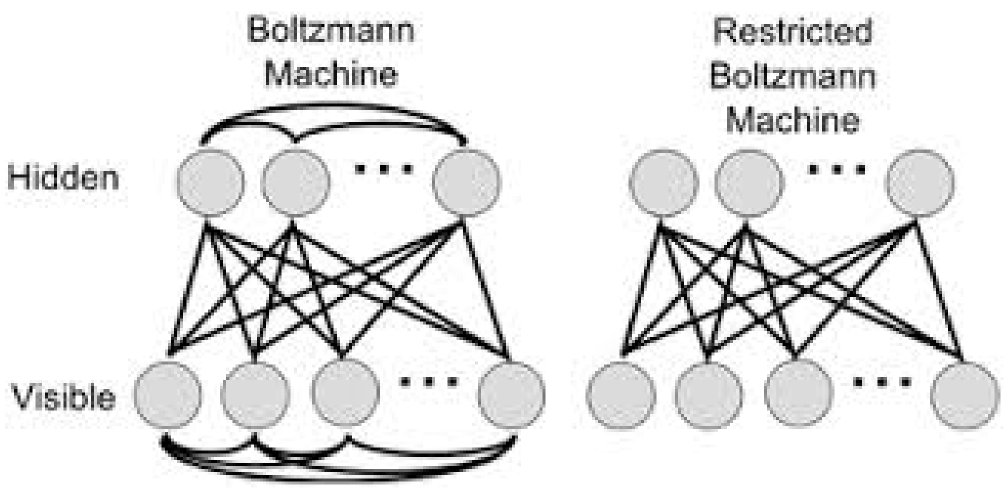


Visible and hidden units are conditionally independent given one another

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i p(h_i \mid \boldsymbol{v})$$

$$p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_j p(v_j \mid \boldsymbol{h})$$

Restricted Boltzmann machine (RBM)



Visible and hidden units are conditionally independent given one another

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_i p(h_i \mid \boldsymbol{v})$$

$$p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_j p(v_j \mid \boldsymbol{h})$$

Following the same principle of maximising log likelihood by means of gradient ascent, one obtains:

$$\Delta w_{ji} = \varepsilon \frac{\partial L(\mathbf{W})}{\partial w_{ji}} = \varepsilon \left(\left\langle v_j h_i \right\rangle_{\text{data}} - \left\langle v_j h_i \right\rangle_{\text{model}} \right)$$

Restricted Boltzmann machine (RBM)

Visible and hidden units are conditionally independent given one another

$$P(h_i = 1 | \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 | \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:} \boldsymbol{h}\right)}$$

$\Leftarrow$

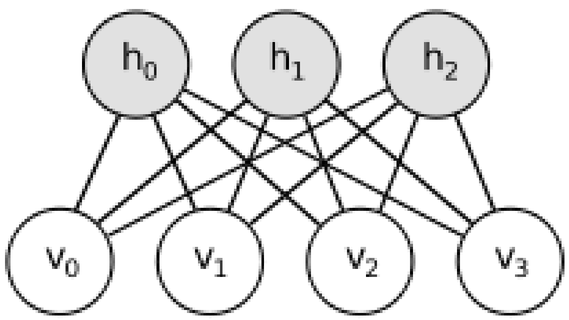
$$p(\boldsymbol{h} | \boldsymbol{v}) = \prod_i p(h_i | \boldsymbol{v})$$

$$p(\boldsymbol{v} | \boldsymbol{h}) = \prod_j p(v_j | \boldsymbol{h})$$

Following the same principle of maximising log likelihood by means of gradient ascent, one obtains:

$$\Delta w_{ji} = \varepsilon \frac{\partial L(\mathbf{W})}{\partial w_{ji}} = \varepsilon \left(\left\langle v_j h_i \right\rangle_{\mathrm{data}} - \left\langle v_j h_i \right\rangle_{\mathrm{model}} \right)$$

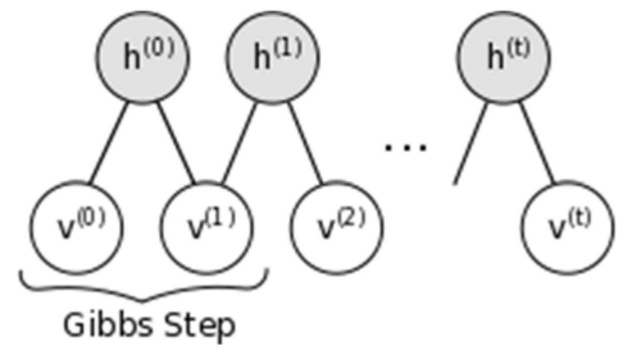
RBM learning with Contrastive Divergence (CD)

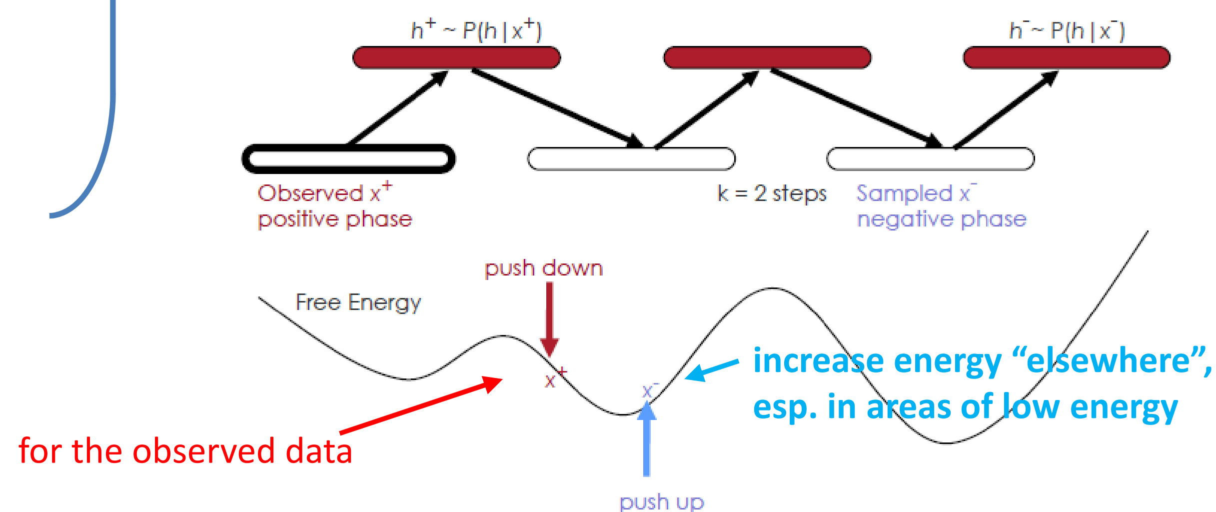


$$P(h_i = 1 \mid \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 \mid \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:} \boldsymbol{h}\right)}$$

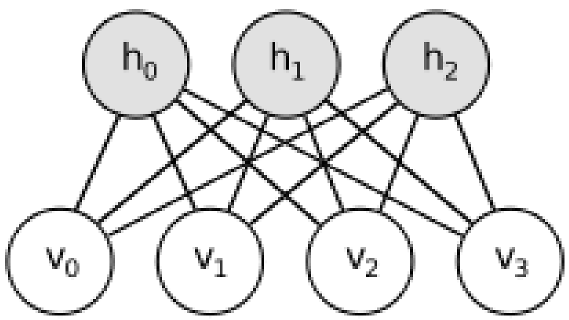
Gibbs sampling





Hinton, 2003

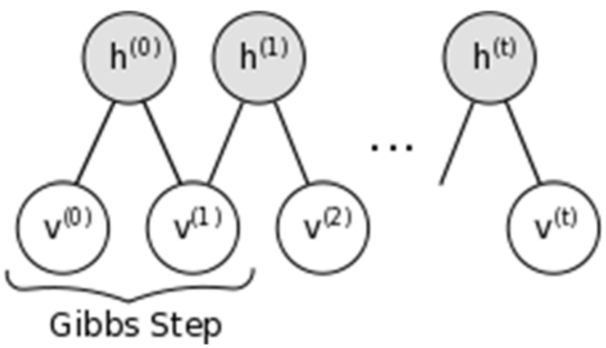
RBM learning with Contrastive Divergence (CD)

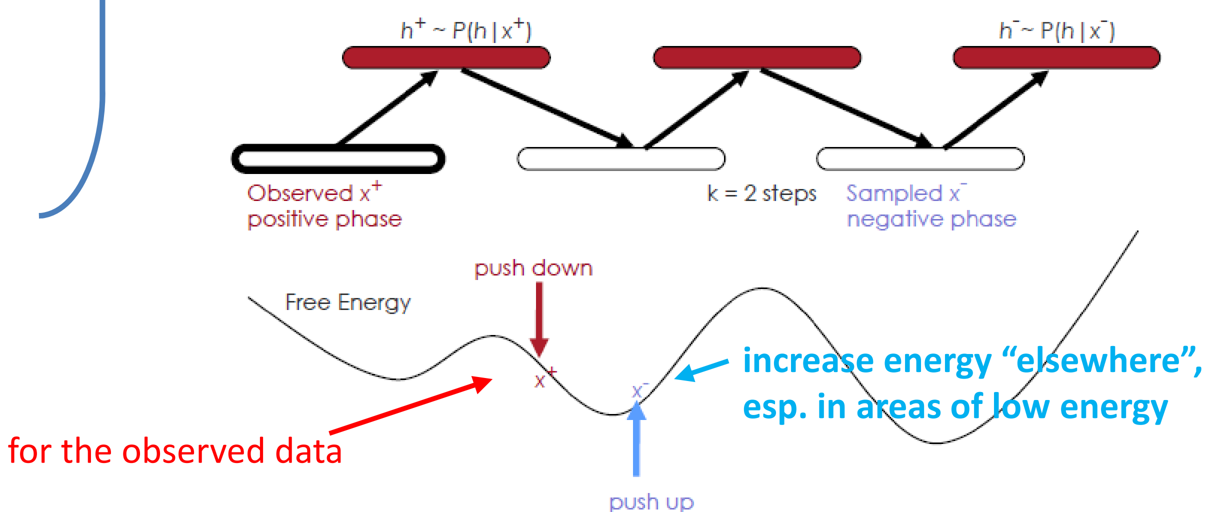


$$P(h_i = 1 \mid \boldsymbol{v}) = \frac{1}{1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)}$$

$$P(v_j = 1 \mid \boldsymbol{h}) = \frac{1}{1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:} \boldsymbol{h}\right)}$$

Gibbs sampling





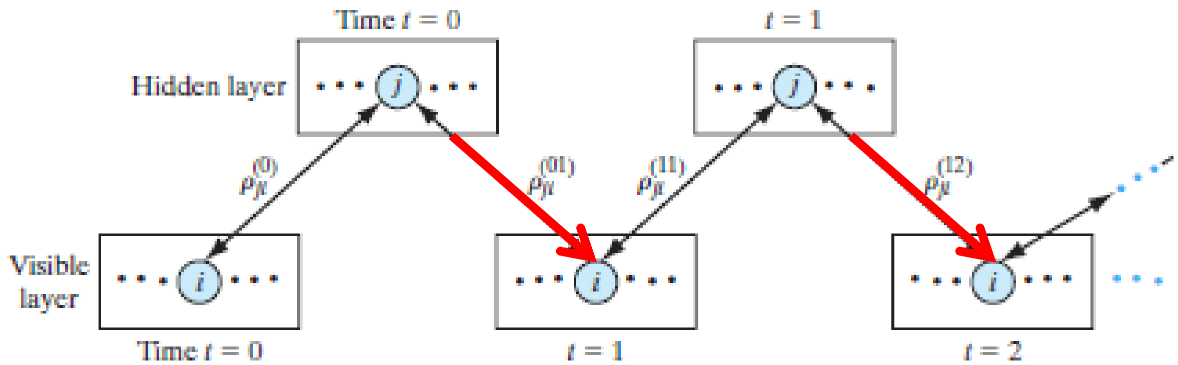
Hinton, 2003

<u>GOOD TO KNOW:</u>

Contrastive Divergence does not optimise the likelihood but it works effectively!

CD$_k$ recipe for training RBM

Gibbs sampling



1) Clamp the visible units with an input vector and update hidden units.

$$P(h_i = 1 \mid \boldsymbol{v}) = \left(1 + \exp\left(-bias_{h_i} - \boldsymbol{v}^{\mathrm{T}} \mathbf{W}_{:,i}\right)\right)^{-1}$$

2) Update all the visible units in parallel to get a **reconstruction**.

$$P(v_j = 1 \mid \boldsymbol{h}) = \left(1 + \exp\left(-bias_{v_j} - \mathbf{W}_{j,:}\, \boldsymbol{h}\right)\right)^{-1}$$

3) Collect the statistics for correlations after *k* steps using mini-batches and update weights:

$$\Delta w_{j,i} = \frac{1}{N} \sum_{n=1}^{N} \left(v_j^{(n)} h_i^{(n)} - \hat{v}_j^{(n)} \hat{h}_i^{(n)} \right)$$

From RBM to Gaussian-Bernoulli RBM

Bernoulli-Bernoulli (binary-binary)

$$p(v_i = 1|\mathbf{h}) = g\left(\sum_j W_{ij} h_j + b_i\right)$$

$$p(h_j = 1|\mathbf{v}) = g\left(\sum_i W_{ij} v_i + a_j\right)$$

Gaussian-Bernoulli (real/cont.-binary)

$$p(v_i = x|\mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{\left(x - b_i - \sigma_i \sum_j h_j W_{ij}\right)^2}{2\sigma_i^2}\right),$$

$$p(h_j = 1|\mathbf{v}) = g\left(b_j + \sum_i W_{ij} \frac{v_i}{\sigma_i}\right),$$

Visible units are real-valued whereas hidden units remain binary.

Salakhutdinov, 2015

From RBM to Gaussian-Bernoulli RBM

Bernoulli-Bernoulli (binary-binary)

$$p(v_i = 1|\mathbf{h}) = g\left(\sum_j W_{ij}h_j + b_i\right)$$

$$p(h_j = 1|\mathbf{v}) = g\left(\sum_i W_{ij}v_i + a_j\right)$$

Gaussian-Bernoulli (real/cont.-binary)

$$p(v_i = x|\mathbf{h}) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{\left(x - b_i - \sigma_i \sum_j h_j W_{ij}\right)^2}{2\sigma_i^2}\right),$$

$$p(h_j = 1|\mathbf{v}) = g\left(b_j + \sum_i W_{ij}\frac{v_i}{\sigma_i}\right),$$

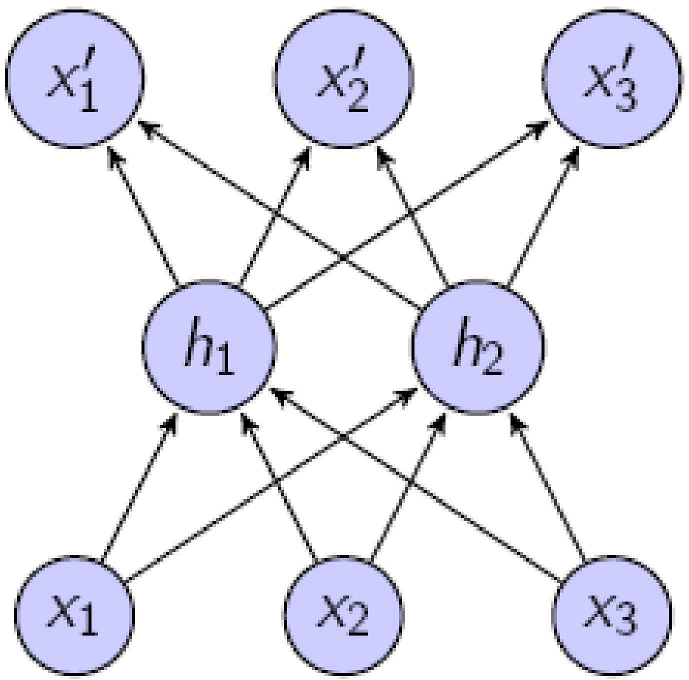
Visible units are real-valued whereas hidden units remain binary.

The derivative of the log-likelihood:

$$\frac{\partial \log P(\mathbf{v};\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{\text{data}}}\left[\frac{1}{\sigma_i}v_i h_j\right] - \mathbb{E}_{P_{\text{model}}}\left[\frac{1}{\sigma_i}v_i h_j\right]$$

Salakhutdinov, 2015

Autoencoders – principles



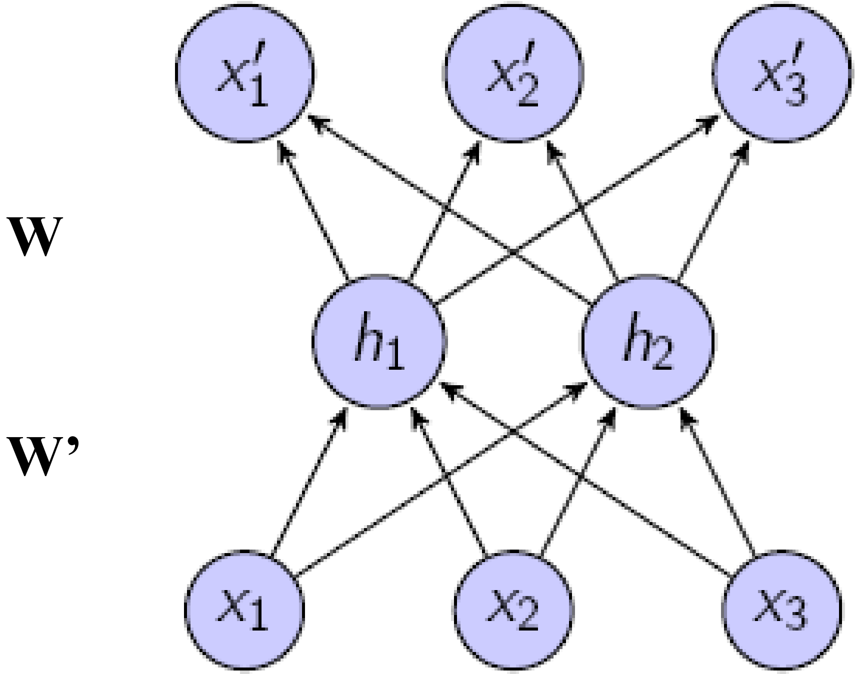
Decoder: $x' = f(x)$

Encoder: $h = g\left(f(x)\right)$

The idea is to minimise the loss function, L:

$$L\left(x, g\left(f(x)\right)\right)$$

Autoencoders



Decoder: $x' = \sigma(W'h + d)$

Encoder: $h = \sigma(Wx + b)$

Encourage h to give small reconstruction error:

- e.g. $Loss = \sum_m ||x^{(m)} - DECODER(ENCODER(x^{(m)}))||^2$
- Reconstruction: $x' = \sigma(W'\sigma(Wx + b) + d)$

Different types of autoencoders

- Undercomplete autoencoders
  - hidden layer is smaller than the input dimensionality
- Overcomplete regularised autoencoders
  - Larger hidden layer size with the regularisation (to avoid overfitting and copying input to the output)

$$L\left(x, g\left(f(x)\right)\right) + \Omega(h), \quad h = f(x)$$

  - Sparse autoencoders, denosing autoencoders
  - Deep autoencoders

Sparse autoencoders

- Penalizing non-sparse solutions can be seen as adding latent variables with a prior and maximising likelihood

$$\log p_{\text{model}}(\boldsymbol{h},\boldsymbol{x}) = \log p_{\text{model}}(\boldsymbol{h}) + \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h})$$

for example: $$p_{\text{model}}(\boldsymbol{h}) = \prod_i \frac{\lambda}{2} e^{-\lambda |h_i|} \Rightarrow \quad \Omega(\boldsymbol{h}) = \lambda \sum |h_i|$$

Denoising autoencoders

$$L\left(\boldsymbol{x}, g\left(f\left(\tilde{\boldsymbol{x}}\right)\right)\right), \quad h = f\left(\tilde{\boldsymbol{x}}\right)$$

Corrupted copy of x

Autoencoders have to undo this corruption beyond simply coping the input.

Denoising autoencoders

$$L\left(\boldsymbol{x}, g\left(f\left(\tilde{\boldsymbol{x}}\right)\right)\right), \quad h = f\left(\tilde{\boldsymbol{x}}\right)$$

Corrupted copy of x

Autoencoders have to undo this corruption beyond simply coping the input.

1. A training sample is sampled from the training data.
2. A corrupted version of the sample ***x*** is drawn from some corruption process
$$C(\tilde{\boldsymbol{x}} \mid \boldsymbol{x} = \boldsymbol{s})$$
3. $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is used as a training sample to estimate the autoencoder's reconstruction distribution $p_{reconstruction}(\tilde{\boldsymbol{x}} \mid \boldsymbol{x}) = p_{decoder}(\boldsymbol{x} \mid \boldsymbol{h}), \quad \boldsymbol{h} = f(\tilde{\boldsymbol{x}})$

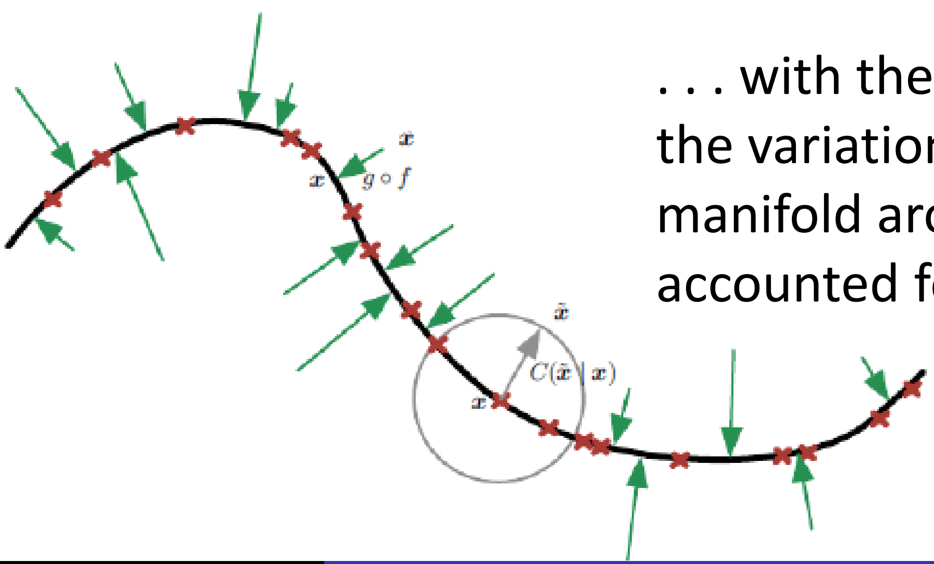
Denoising autoencoders

$$L\left(\boldsymbol{x}, g\left(f\left(\tilde{\boldsymbol{x}}\right)\right)\right), \quad h = f\left(\tilde{\boldsymbol{x}}\right)$$

Corrupted copy of x

Autoencoders have to undo this corruption beyond simply coping the input.

Learning a *vector field* around a *low-dimensional manifold . . .*



. . . with the principle that only the variations tangent to the manifold around ***x*** should be accounted for by changes in ***h***

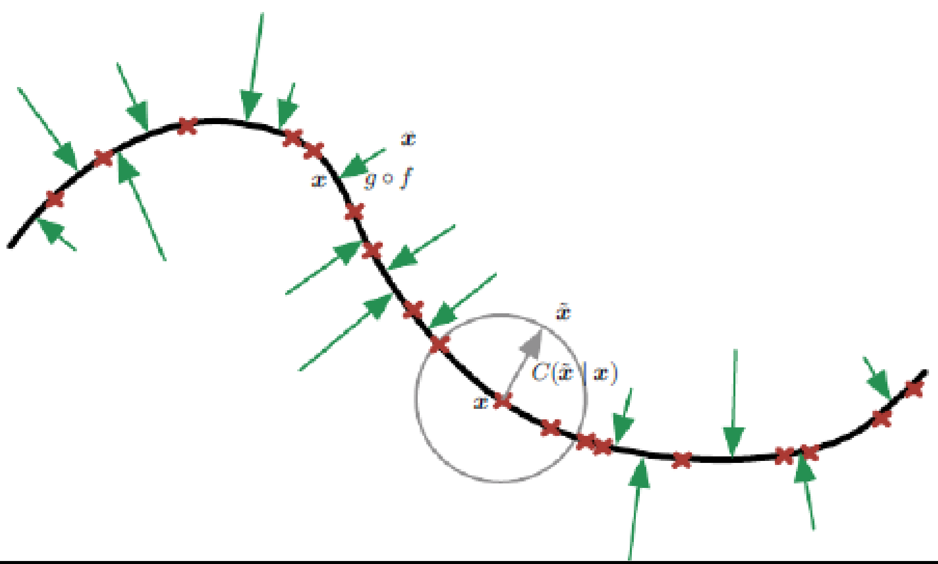
Goodfellow et al.

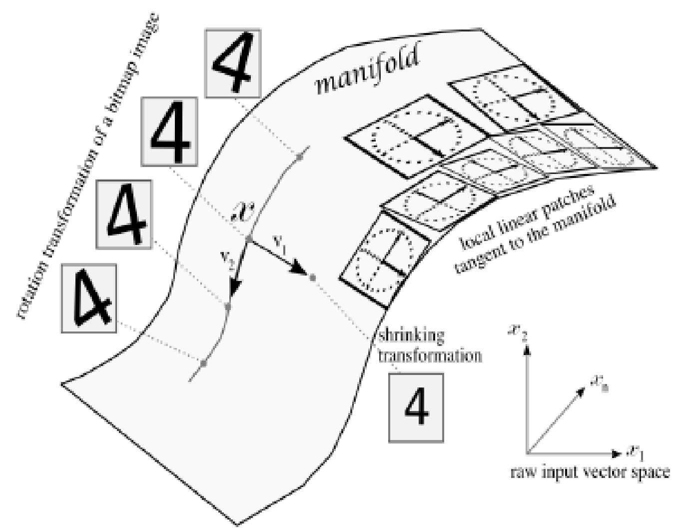
Denoising autoencoders

When training autoencoders there is a **compromise**

I. Need to approximately recover ***x*** – *reconstruction* force

II. Need to satisfy the regularization term – *regularisation* force.

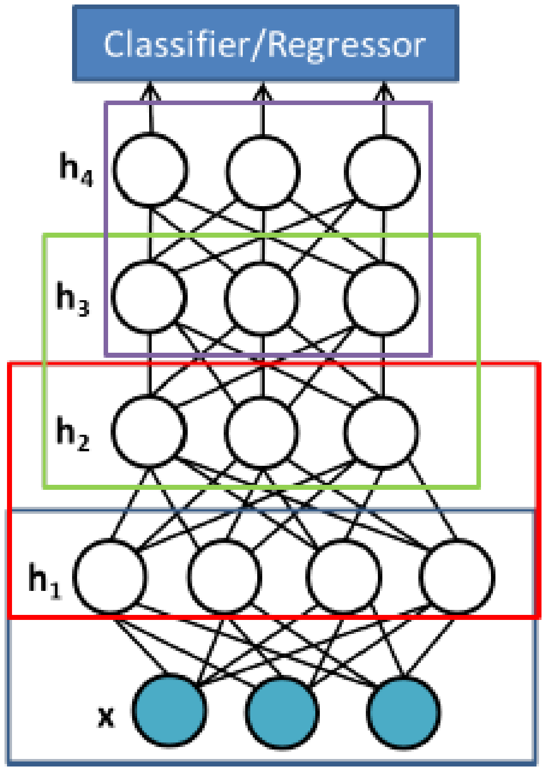




Goodfellow et al.

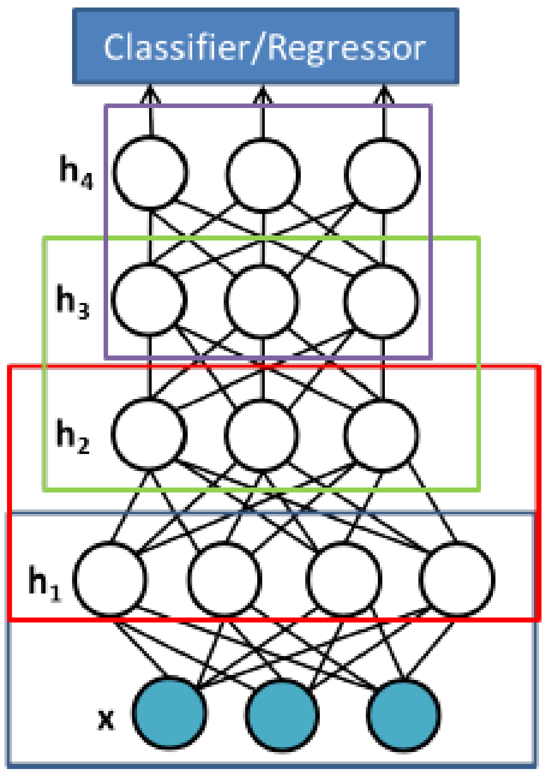
Deep belief nets

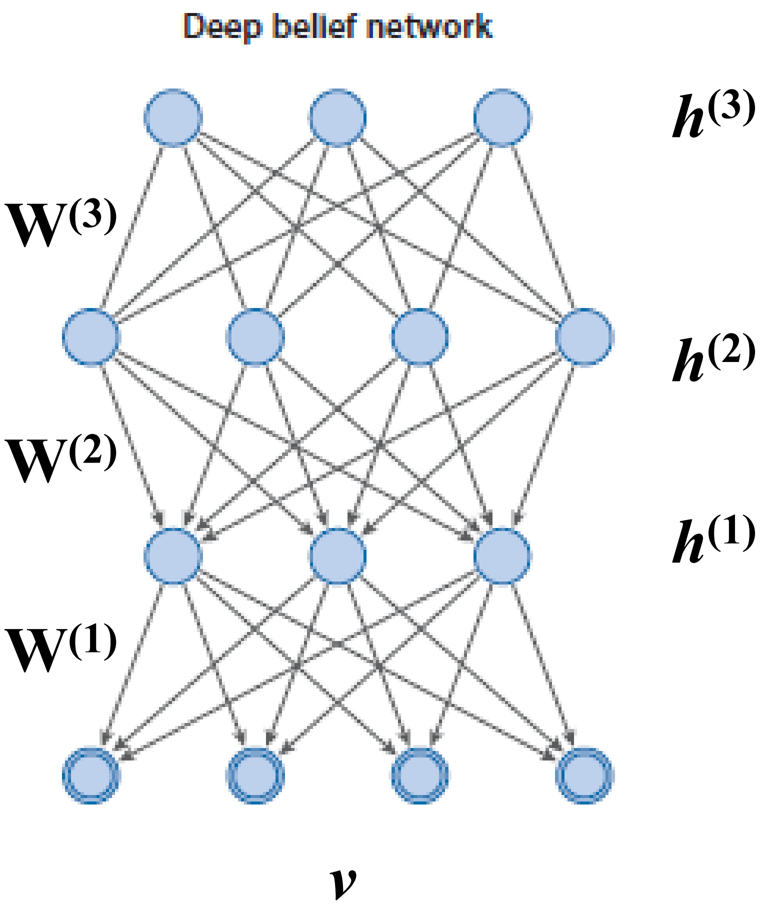
Greedy layer-wise training approach with the use of RBMs



Deep belief nets

Greedy layer-wise training approach with the use of RBMs

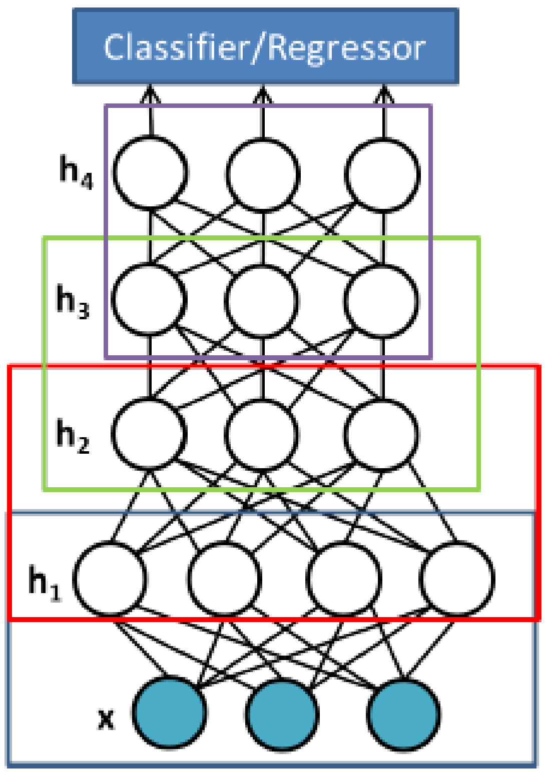


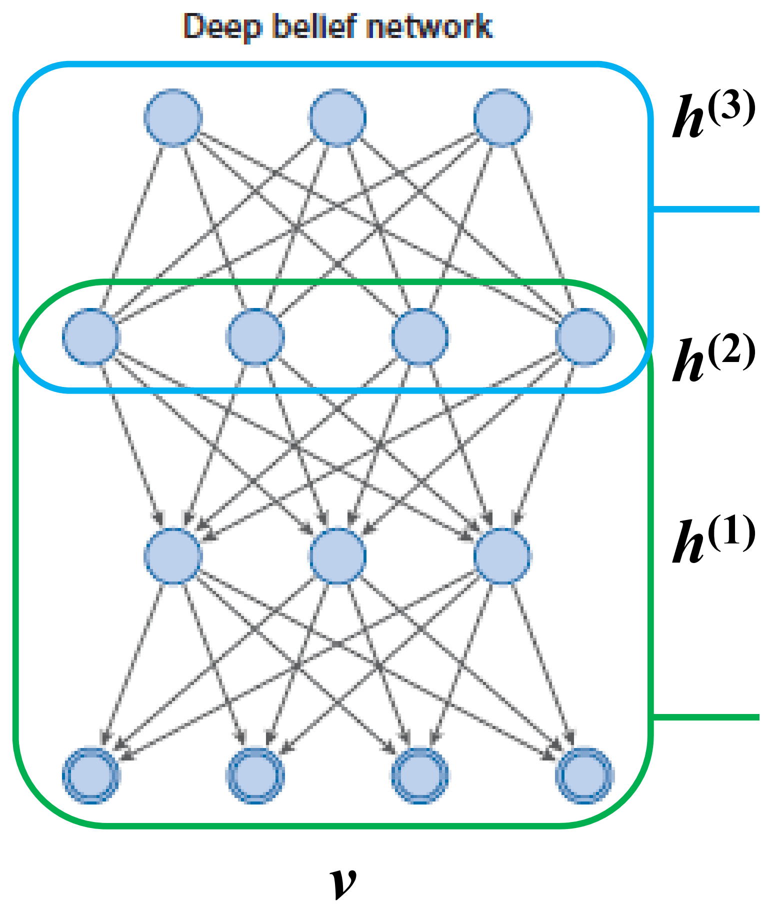


Salakhutdinov, 2015

Deep belief nets

Greedy layer-wise training approach with the use of RBMs





undirected part of the network (bipartite graph of RBM)

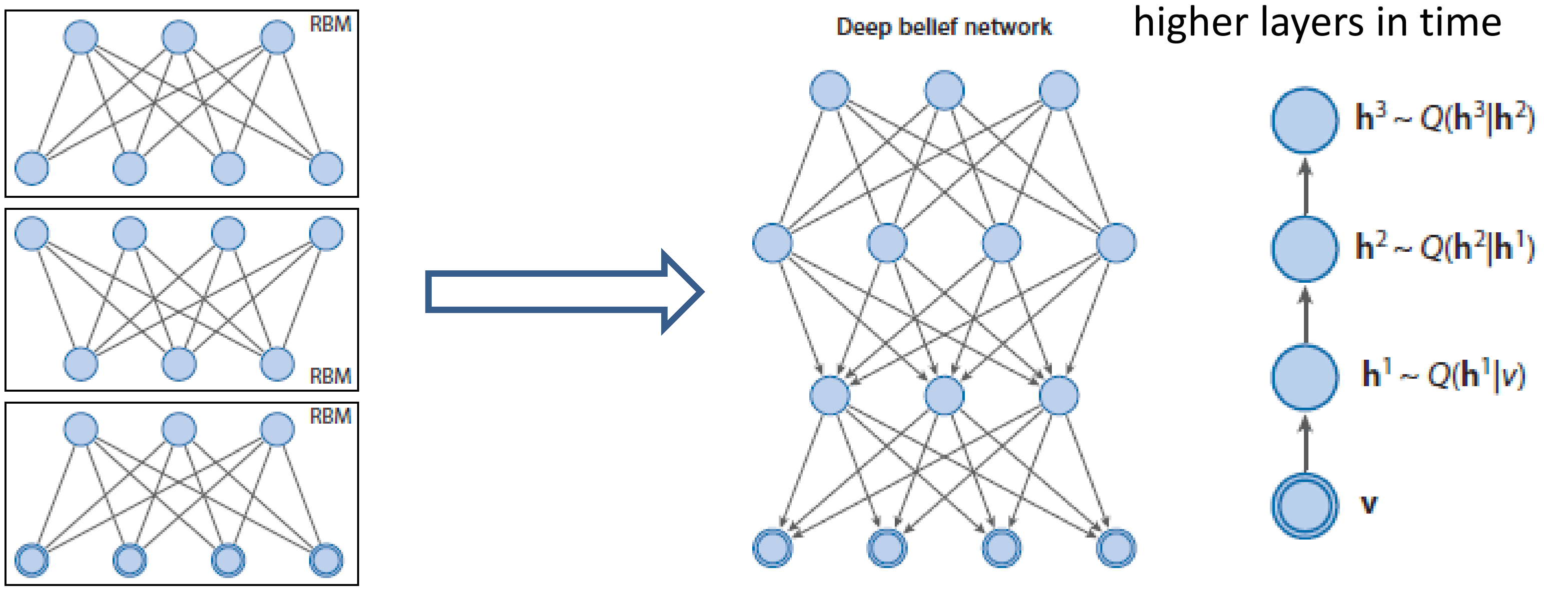
directed part of the network (sigmoid belief network)

Salakhutdinov, 2015

Deep belief nets

Approach 1

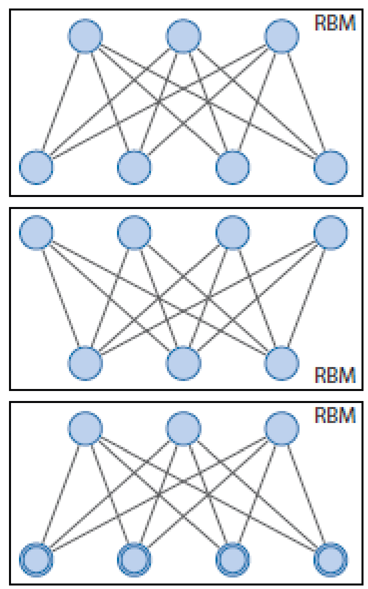
Bottom-up pass by stochastically activating higher layers in time

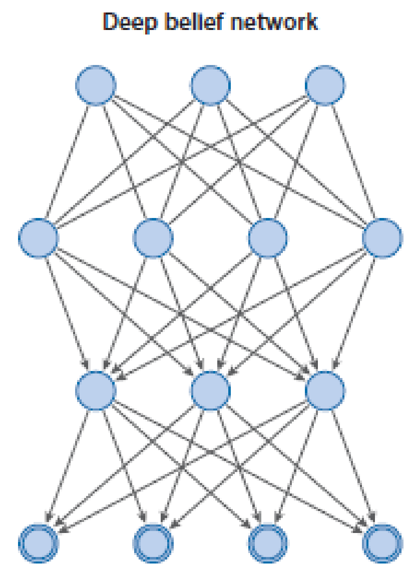


1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathrm{h}^{(1)}$ from $Q(\mathrm{h}^{(1)}|\mathrm{v}) = P(\mathrm{h}^{(1)}|\mathrm{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.
3: Fix the parameters $W^{(2)}$ that define the second layer of features, and use the samples $\mathrm{h}^{(2)}$ from $\boxed{Q(\mathrm{h}^{(2)}|\mathrm{h}^{(1)}) = P(\mathrm{h}^{(2)}|\mathrm{h}^{(1)}, W^{(2)})}$ as the data for training the third layer of binary features.
4: Proceed recursively for the next layers.

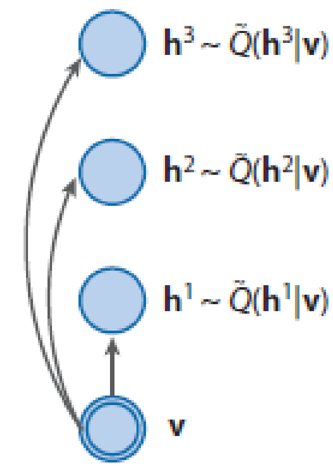
Deep belief nets

Approach 2





Bottom-up pass by stochastically activating higher layers in time



Assumption about fully factorised approximating distribution

$$\tilde{Q}(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = \prod_{l=1}^{L} \tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}) \Rightarrow$$

$$\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v}) = \prod_j q(h_j^{(1)}|\mathbf{v}),$$

$$\tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}) = \prod_j q(h_j^{(l)}|\mathbf{v}),$$

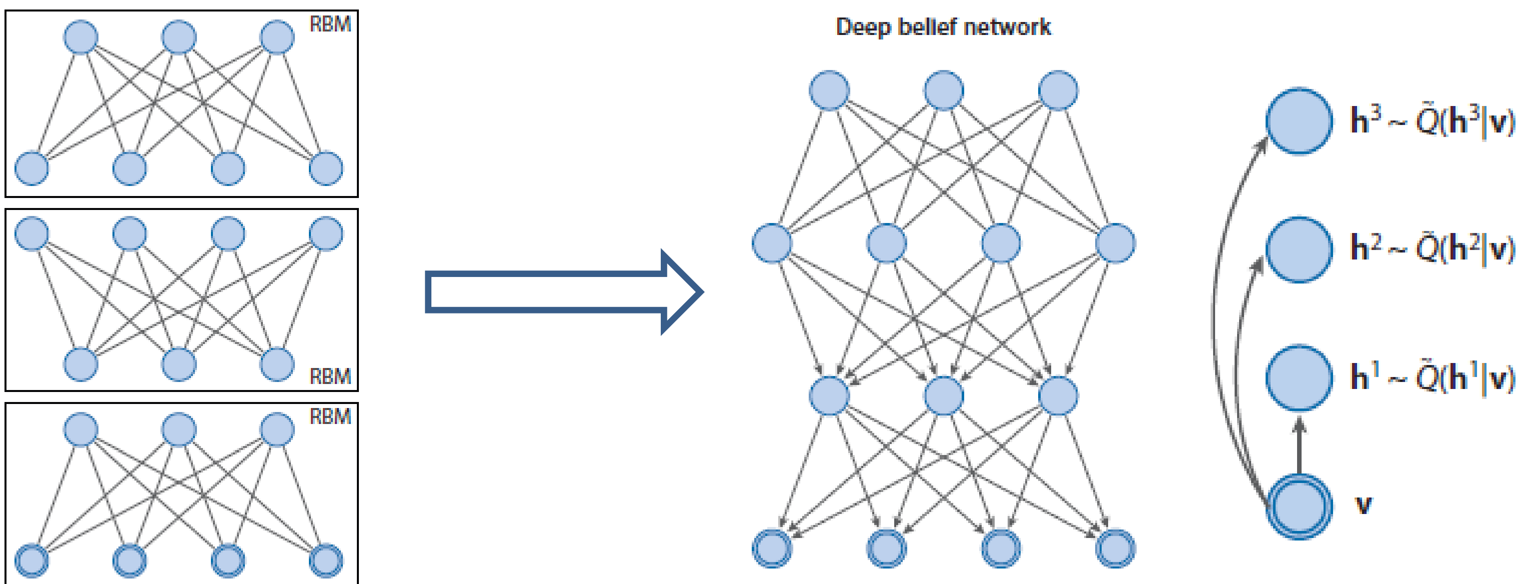
where:

$$q(h_j^{(1)} = 1|\mathbf{v}) = g\left(\sum_i W_{ij}^{(1)} v_i + a_j^{(1)}\right), \text{ and}$$

$$q(h_j^{(l)} = 1|\mathbf{v}) = g\left(\sum_i W_{ij}^{(l)} q(h_i^{(l-1)} = 1|\mathbf{v}) + a_j^{(l)}\right),$$

Deep belief nets

Approach 2

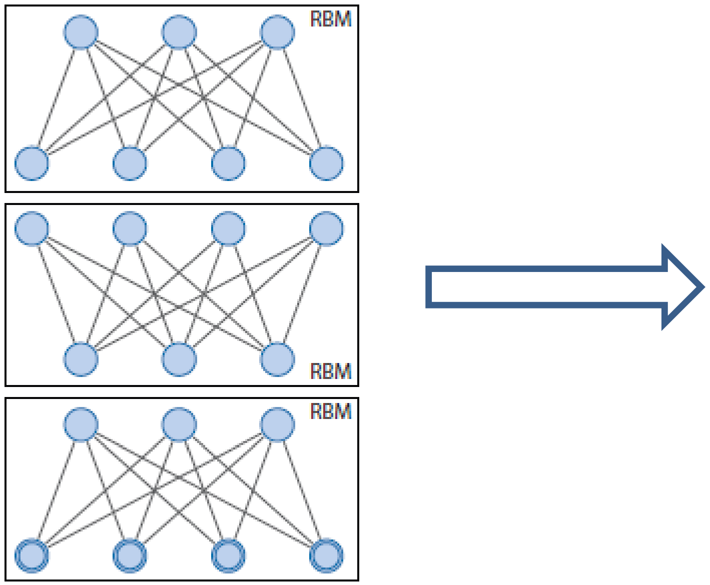


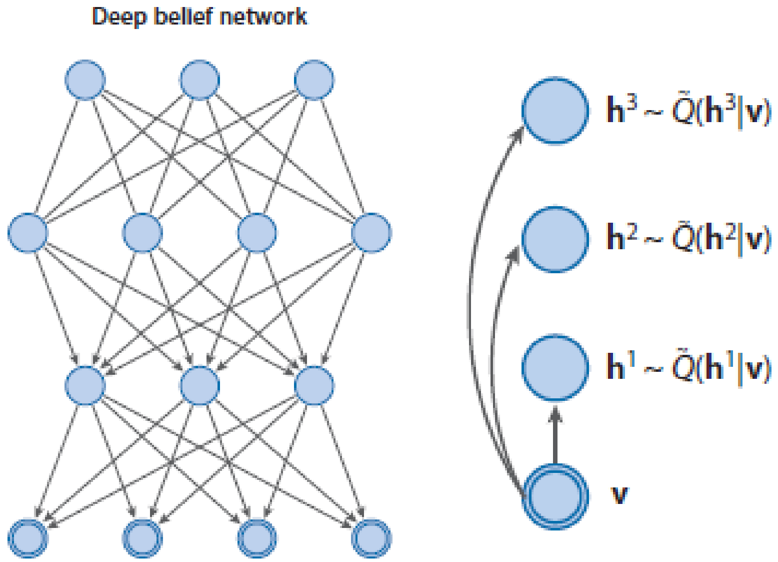
$$\tilde{Q}(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = \prod_{l=1}^{L} \tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}),$$

1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathbf{h}^{(1)}$ from $\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v}) = P(\mathbf{h}^{(1)}|\mathbf{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.
3: Fix the parameters $W^{(2)}$ that define the second layer of features, and use the samples $\mathbf{h}^{(2)}$ from $\tilde{Q}(\mathbf{h}^{(2)}|\mathbf{v})$ as the data for training the third layer of binary features.
4: Proceed recursively for the next layers.

Deep belief nets

Approach 2





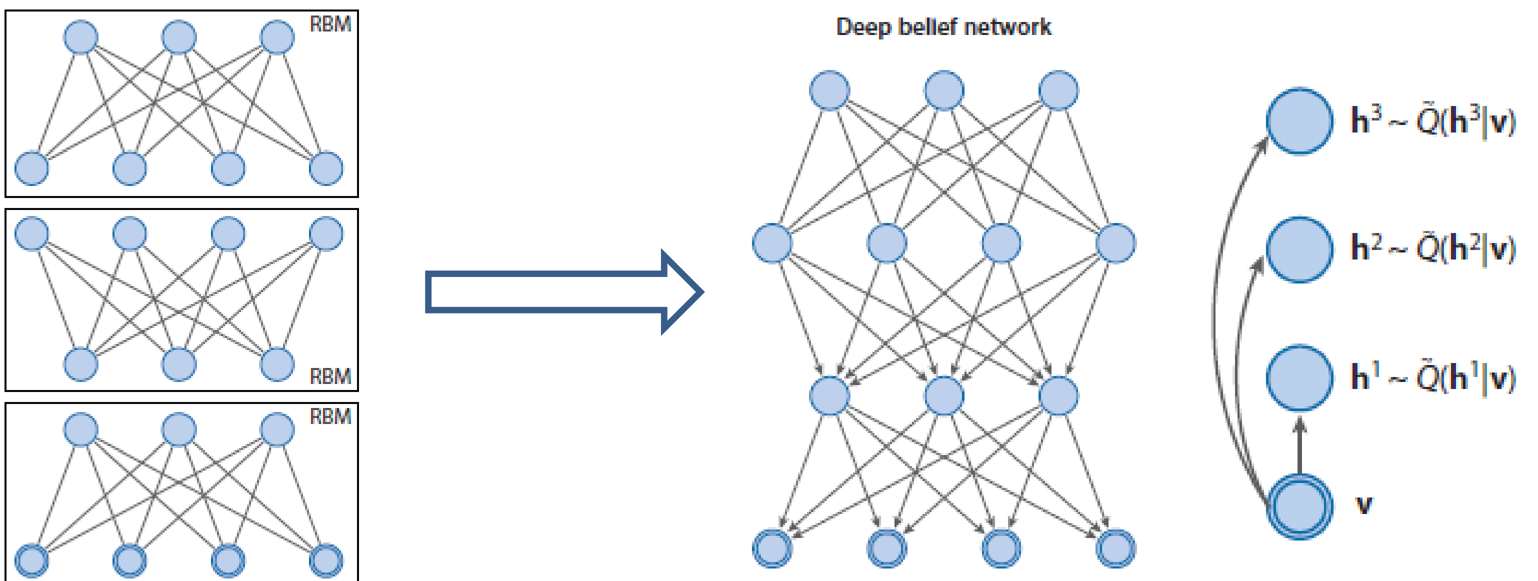
$$\tilde{Q}(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = \prod_{l=1}^{L} \tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}),$$

1: Fit the parameters $W^{(1)}$ of the first-layer RBM to data.
2: Fix the parameter vector $W^{(1)}$, and use samples $\mathbf{h}^{(1)}$ from $\tilde{Q}(\mathbf{h}^{(1)}|\mathbf{v}) = P(\mathbf{h}^{(1)}|\mathbf{v}, W^{(1)})$ as the data for training the next layer of binary features with an RBM.

The difference lies in how we obtain sampling distribution to generate input for greedy training another layer.

Deep belief nets

Approach 2



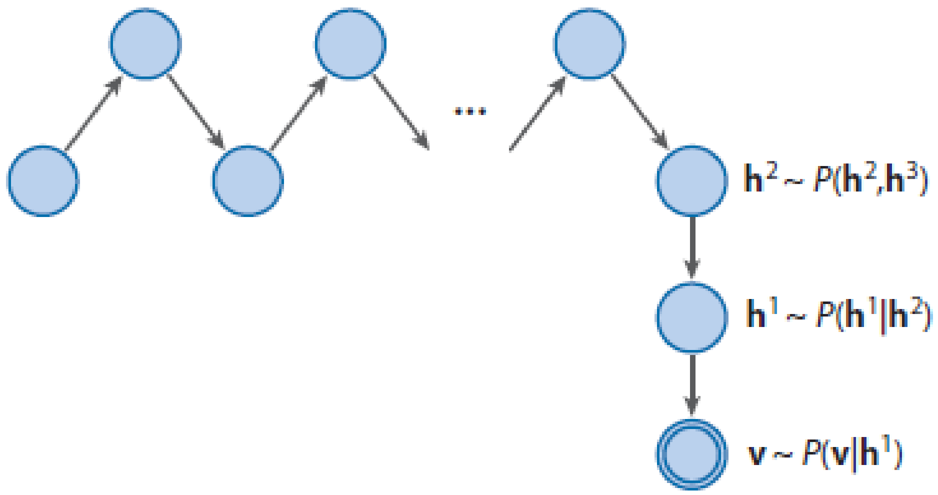
$$\tilde{Q}(\mathbf{h}^{(1)}, \ldots, \mathbf{h}^{(L)}|\mathbf{v}) = \prod_{l=1}^{L} \tilde{Q}(\mathbf{h}^{(l)}|\mathbf{v}),$$

For a fully factorised model , $\tilde{Q}(\boldsymbol{h}^{(l)} \mid \boldsymbol{v})$, needed for sampling data at the next level, a single deterministic bottom-up pass can be executed on real-valued probabilities.

$$q(b_j^{(l)} = 1|\mathbf{v}) = g\left(\sum_i W_{ij}^{(l)} q(b_i^{(l-1)} = 1|\mathbf{v}) + a_j^{(l)}\right),$$

Approximate sampling from DBN

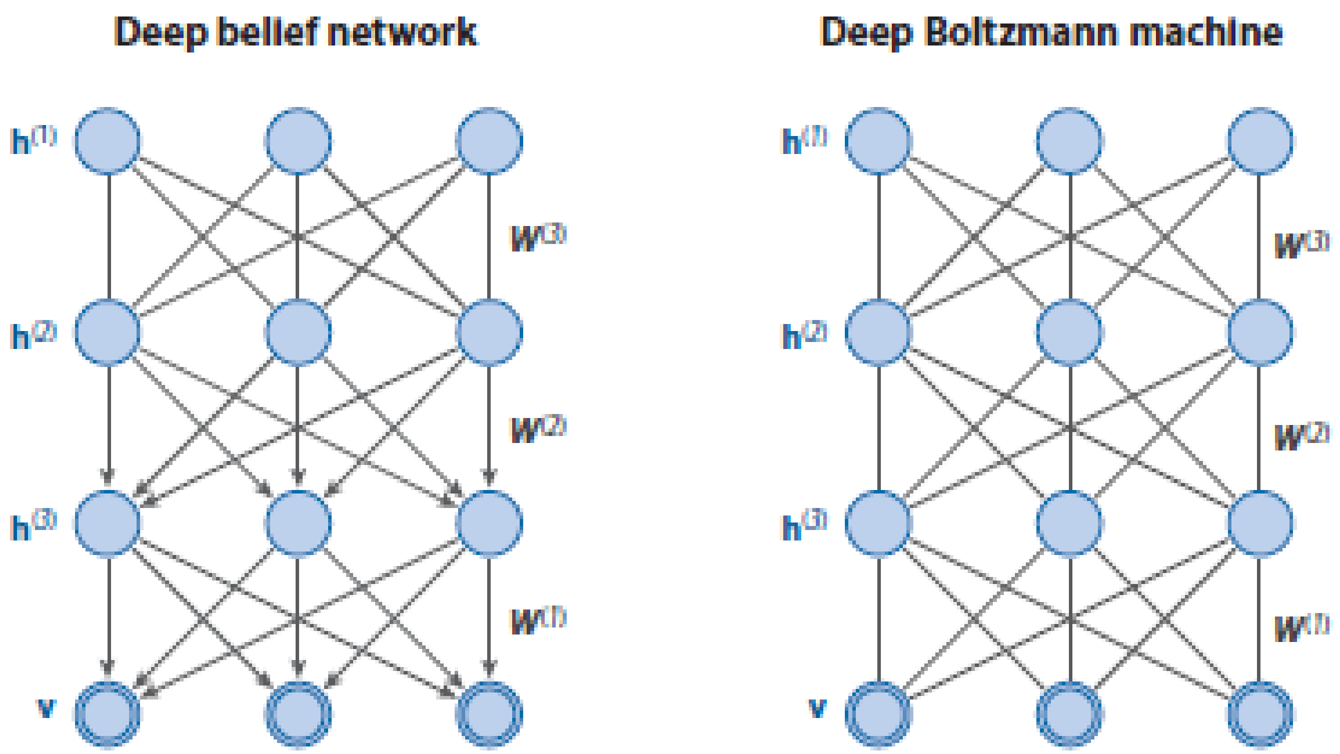
Gibbs sampling chain in the RBM part



RBM part
(undirected part of the graph)

single-run sampling
(through the directed graph)

DBN vs DBM



Deep belief network
h(1)
h(2)
h(3)
v
W(3)
W(2)
W(1)
Deep Boltzmann machine
h(1)
h(2)
h(3)
v
W(3)
W(2)
W(1)

DBN vs DBM

Greedy layer-wise pre-training approach*

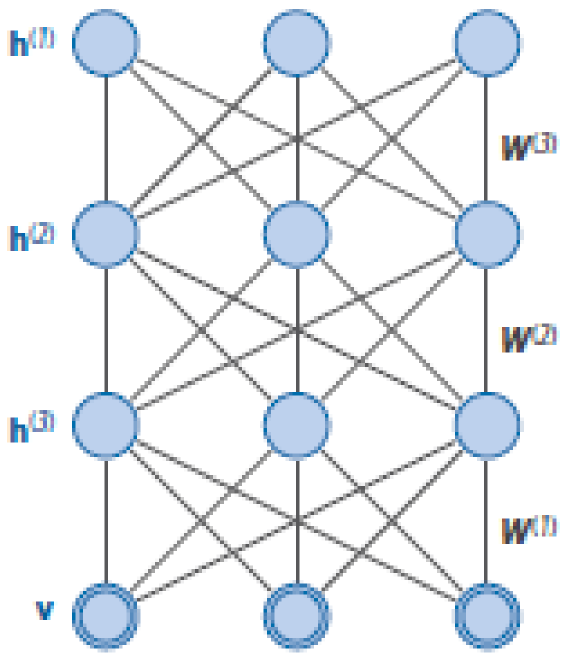
$$p(h_j^{(1)} = 1|\mathbf{v}, \mathbf{h}^{(2)}) = g\left(\sum_i W_{ij}^{(1)} v_i + \sum_m W_{jm}^{(2)} h_m^{(2)}\right),$$

$$p(h_m^{(2)} = 1|\mathbf{h}^{(1)}, \mathbf{h}^{(3)}) = g\left(\sum_j W_{jm}^{(2)} h_j^{(1)} + \sum_l W_{ml}^{(3)} h_l^{(3)}\right),$$

$$p(h_l^{(3)} = 1|\mathbf{h}^{(2)}) = g\left(\sum_m W_{ml}^{(3)} h_m^{(2)}\right),$$

$$p(v_i = 1|\mathbf{h}^{(1)}) = g\left(\sum_j W_{ij}^{(1)} h_j^{(1)}\right).$$

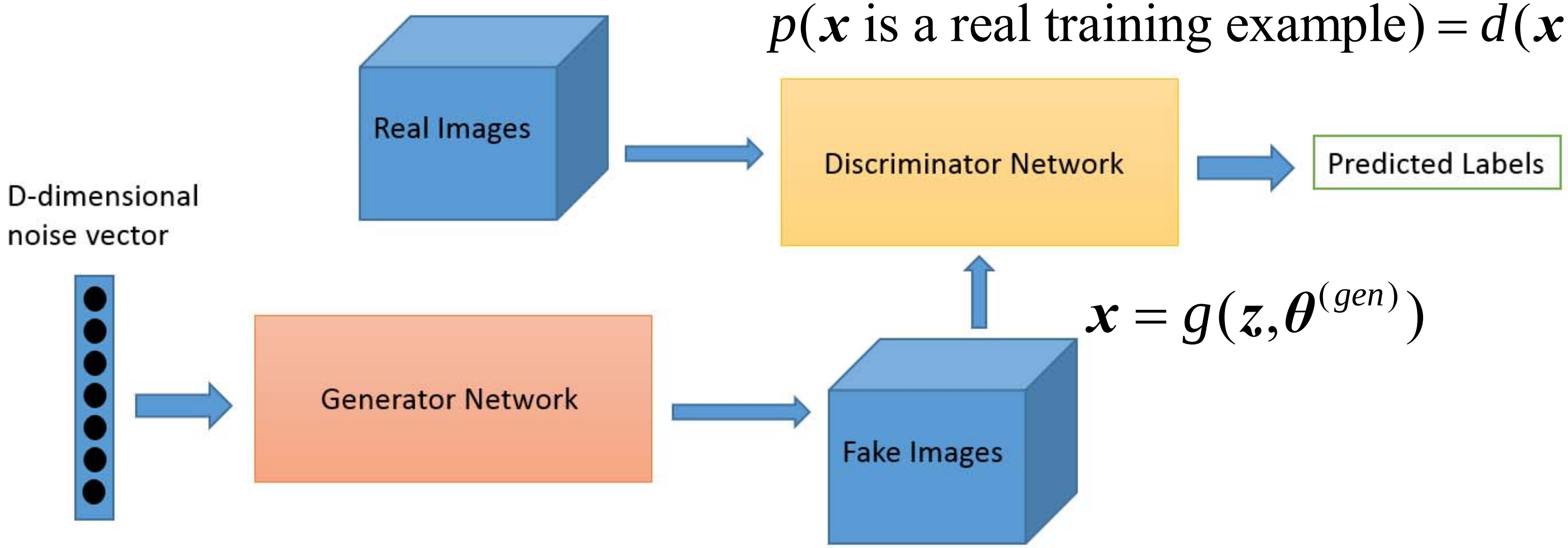
Deep Boltzmann machine



*Salakhutdinov and Hinton, 2009

Generative adversarial networks (GANs)

$$p(\boldsymbol{x} \text{ is a real training example}) = d(\boldsymbol{x}, \boldsymbol{\theta}^{(discr)})$$



$$\boldsymbol{x} = g(\boldsymbol{z}, \boldsymbol{\theta}^{(gen)})$$

Discriminator network received some payoff v and the generator receives $-v$, so it is a zero-sum game. Both attempt to maximise their own payoff, so at the convergence:

$$g^* = \arg\min_g \max_d v(g,d)$$

$$v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x}\sim p_{\text{model}}} \log(1 - d(\boldsymbol{x}))$$

Summary

1. Central role of data representations (sparse distributed code in the cortex).
2. Deep learning is about learning features that constitute representations (string link to hierarchical processing in the brain).
3. RBMs and autoencoders are key computational blocks for learning representations and building deep generative models – let them learn/extract features.
4. The earlier popularity of greedy layer-wise pretraining, today we rather rely on dropout and ReLU units (rather than sigmoidal).
5. Generative power of deep models.

Recommended reading

- Goodfellow, I., Bengio, Y., & Courville, A. *Deep learning*, chapters 6, 14, 15, 20 .
- Salakhutdinov, R. (2015) Learning deep generative models. *Annual Reviews of Statistics and Its Application*, 2, p.361–385.
- Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence,* 35 (8), p.1798-1828.
- Hinton, G. (2010). A Practical Guide to Training Restricted Boltzmann Machines. Technical report UTML TR 2010–003.
- Bengio, Y. (2009) Learning deep architectures for AI. *Foundations and trends® in Machine Learning,* 2.1. p.1-127.
- Salakhutdinov ,R., & Hinton, G. (2009). Deep Boltzmann machines. *Proc. 12th Int. Conf. Artif. Intell. Stat.*, Clearwater Beach, FL, pp. 448–55. Brookline, MA: Microtome.