Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava

Neural Networks

Lecture 12

Introduction to stochastic models

Igor Farkaš

2024

Motivation and probabilistic concepts

- Unlike discriminative models, stochastic (probabilistic) models are generative, providing an added value for some tasks (denoising, missing value restoration, sampling,…)
- Difference b/w a sample and population ($\rightarrow$ generalization)
- Probability distribution (of population) over a random vector *X*
- Avoiding a lookup table approach (no generalization)
- Graphical models – directed and undirected
- Probabilities: prior, posterior, conditional
- Bayes rule

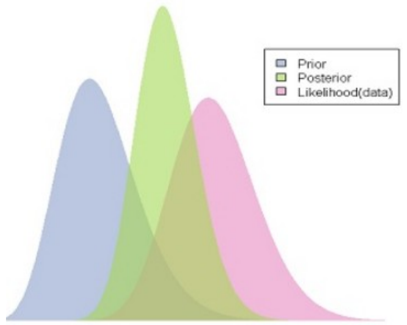
$$P(B|A)=\frac{P(A|B).P(B)}{P(A)}$$

- Machine learning: MSE and maximum likelihood estimation

Example: Bayesian estimation in linear regression

- Assume data set $\{\boldsymbol{x}^{(i)}, d^{(i)}\}$, and linear regressor with parameter vector $\boldsymbol{w}$
- Mean-squares regression: minimize MSE, i.e. $1/N \sum_{i=1}^{N} (d^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)})^2$
- probabilistic approach (in stochastic environment):

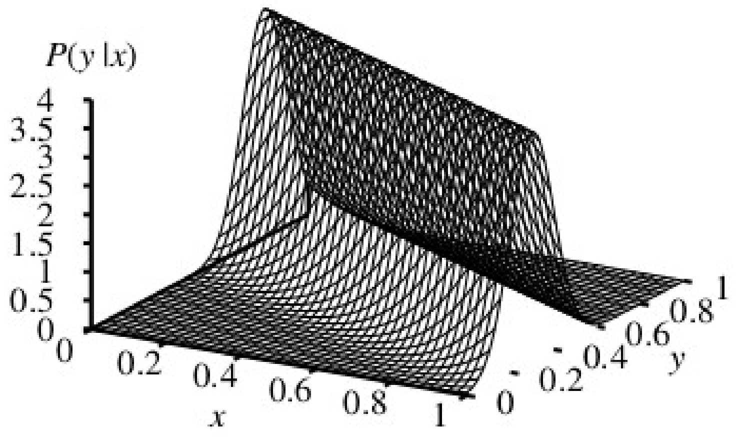
$$p(\boldsymbol{w}|d,\boldsymbol{x}) = \frac{p(d|\boldsymbol{w},\boldsymbol{x})\,p(\boldsymbol{w})}{p(d)} \qquad posterior = \frac{observation * prior}{evidence}$$

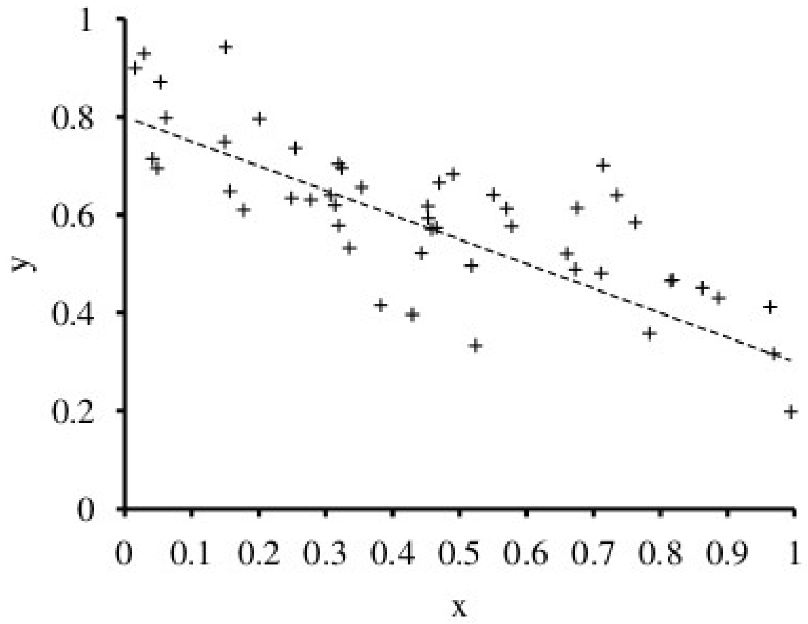


- Observation density is commonly reformulated as likelihood function, i.e. $l(\boldsymbol{w}|d,\boldsymbol{x}) = p(d|\boldsymbol{w},\boldsymbol{x})$
- maximum likelihood (ML) estimate (of $\boldsymbol{w}$): $\boldsymbol{w}_{ML} = \arg\max_{\boldsymbol{w}} l(\boldsymbol{w}|d,\boldsymbol{x})$
- maximum a posteriori (MAP) estimate: $\boldsymbol{w}_{MAP} = \arg\max_{\boldsymbol{w}} \{l(\boldsymbol{w}|d,\boldsymbol{x}) \,.\, p(\boldsymbol{w})\}$
- in general, MAP is more accurate (since it includes priors), but computationally more costly
- for large data sets (i.e. many observations), ML often works well

(Haykin, 2009, sec. 2.3)

Link to linear regression





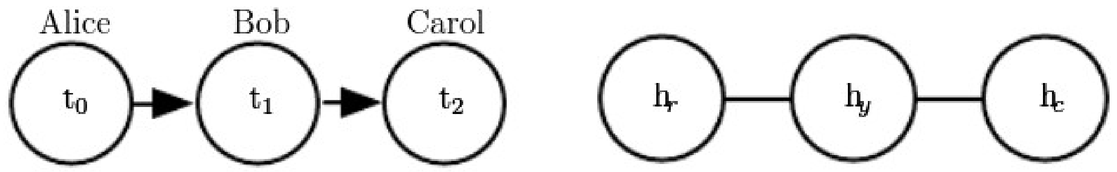
Maximizing $$P(y|x)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-(y-w_1x-w_2)^2}{2\sigma^2}\right)$$ w.r.t. w_1, w_2

= minimizing $$E=\sum_{i=1}^{N}\left(y^{(i)}-w_1x^{(i)}-w_2\right)^2$$ y = *target*

Minimizing the sum of squared errors yields the ML solution for a linear fit assuming Gaussian noise of fixed variance.

Structured probabilistic models

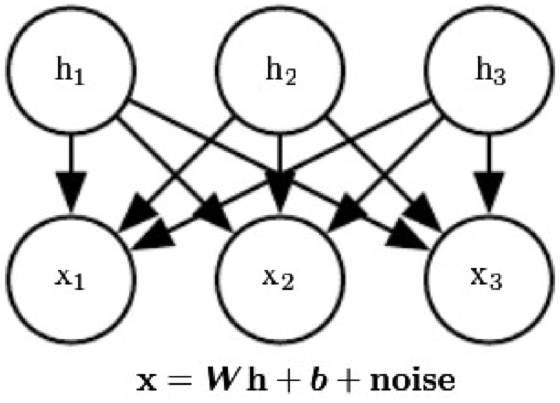
- Graphical (probabilistic) models – directed or undirected



(Goodfellow et al, 2015)

- Structured probabilistic models provide a formal framework for modeling only direct interactions between random variables
- Learning probabilistic distributions yields various advantages
- partition function = normalizing constant = sum (or integral) of all probabilities
- discriminative vs. generative models
- energy based models

Linear factor models


h_1
h_2
h_3
x_1
x_2
x_3
$\mathbf{x} = \boldsymbol{W}\mathbf{h} + \boldsymbol{b} + \text{noise}$

- the simplest directed graphical models
- first we sample latent variables $\mathbf{h} \sim p(\boldsymbol{h})$
- then we sample observables $\boldsymbol{x}$
- they build $p_{\text{model}}(\boldsymbol{x}) = \langle p_{\text{model}}(\boldsymbol{x}|\boldsymbol{h})\rangle_{\boldsymbol{h}}$
- Probabilistic PCA: $\mathbf{x} \sim N(\boldsymbol{x}; \boldsymbol{b}, \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I})$, i.e. $\boldsymbol{x} = \mathbf{W}\boldsymbol{h} + \boldsymbol{b} + \sigma \boldsymbol{z}$,
- $\mathbf{z} \sim N(z, \mathbf{0}, \mathbf{I})$ is Gaussian noise;
- parameters $\mathbf{W}$, σ^2 can be estimated iteratively (e.g. by EM alg.)
- probabilistic PCA becomes (deterministic) PCA as $\sigma \rightarrow 0$
- manifold interpretation of PCA
- nonlinear analogy: probabilistic (generative) SOM (GTM)

(Goodfellow et al, 2015)

Introduction of physics' concepts

- Statistical mechanics formally studies macroscopic equilibrium properties of large systems of elements that are subject to the microscopic laws of mechanics.
- Entropy: The more ordered the system, or the more concentrated the underlying probability distribution, the smaller the entropy will be.
- Energy: States of low energy have a higher probability of occurrence than states of high energy.
- As the temperature *T* is reduced, the probability is concentrated on a smaller subset of low-energy states.

$$P(\mathbf{S}=\mathbf{s})=\frac{1}{Z}\exp\left(-\frac{E(\mathbf{s})}{T}\right) \qquad Z=\sum_{\mathbf{s}'}\exp\left(-\frac{E(\mathbf{s}')}{T}\right)$$ = partition function

$$p_i=\frac{1}{Z}\exp\left(-\frac{E_i}{T}\right) \qquad Z=\sum_{j}\exp\left(-\frac{E_j}{T}\right)$$

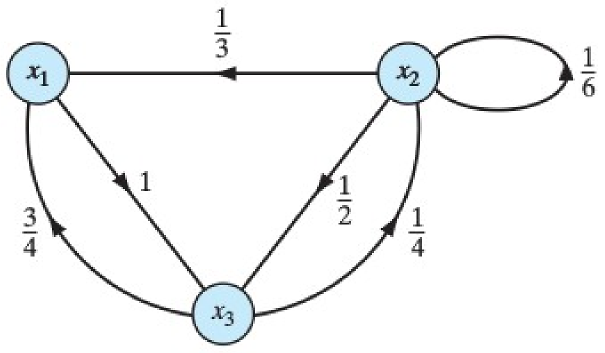
(Haykin, 2009, ch.11)

Free energy and entropy

- Helmholtz' free energy of a physical system: $F = -T \log Z$
- Average energy: $\langle E\rangle = \Sigma_i\, p_i\, E_i$
- Hence $\langle E\rangle - F = -T\, \Sigma_i\, p_i \log p_i = TH$, where H is the entropy
- Minimal free energy: … of a stochastic system with respect to variables of the system is achieved at thermal equilibrium, when the system is governed by the Gibbs (Boltzmann) distribution (Landau & Lifshitz, 1980).
  - and the principle of detailed balance holds ($\pi_i\, p_{ij} = \pi_j\, p_{ji}$)
- Nature likes to follow minimum free energy principle in a closed system (according to the 2nd law of thermodynamics)
- In modeling stochastic phenomena, generative models are useful (e.g. Markov chains)

Markov chains

- Assume a system, as a stochastic process $\{X_n, n=1,2,\ldots\}$ described by a vector of random variables, operating in discrete time
- $P(X_{n+1} = x_{n+1} \mid X_n=x_n, \ldots, X_1=x_1) = P(X_{n+1} = x_{n+1} \mid X_n=x_n)$ (Markov property)
- (square) matrix $\mathbf{P}$ of transition probabilities $p_{ij} = P(X_{n+1} = j \mid X_n = i) \geq 0$
- subject to $\Sigma_j \, p_{ij} = 1$
- If $\mathbf{P}$ is fixed, the MC is homogeneous in time
- MC is ergodic, if ∃ a limit of $\mathbf{P}^n$, for $n \to \infty$ where $\boldsymbol{\pi}^{(n)} = \mathbf{P}^n \, \boldsymbol{\pi}^{(0)}$
- $\boldsymbol{\pi}^{(n)}$ is a state distribution vector at time *n,* example:



$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{2} \\ \frac{3}{4} & \frac{1}{4} & 0 \end{bmatrix}$$

$$\pi_1 = \frac{1}{3}\pi_2 + \frac{3}{4}\pi_3$$
$$\pi_2 = \frac{1}{6}\pi_2 + \frac{1}{4}\pi_3$$
$$\pi_3 = \pi_1 + \frac{1}{2}\pi_2$$

$$\pi_1 = 0.3953$$
$$\pi_2 = 0.1395$$
$$\pi_3 = 0.4652$$

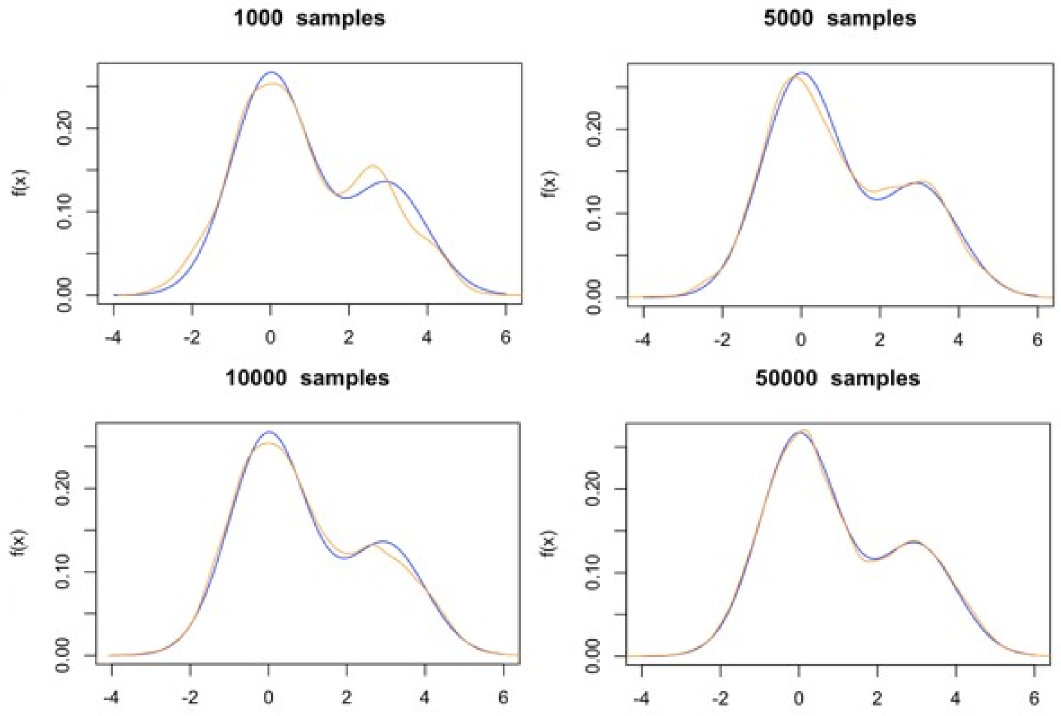
Markov chain Monte Carlo methods

- a class of algorithms for sampling from a probability distribution (e.g. a training set)
- MCMC is any method that produces an ergodic Markov chain whose own stationary distribution is unknown (Robert and Casella, 1999), avoids chicken-and-egg problem
- repeatedly samples $i \rightarrow j$ using $p_{ij} = P(X_{n+1} = j \mid X_n = i)$
- By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by observing the chain after a number of steps.
- The more steps there are, the more closely the distribution of the sample matches the actual desired distribution.
- Examples: Metropolis algorithm, simulated annealing,...

Metropolis algorithm

- is a stochastic algorithm simulating the evolution of a physical system to thermal equilibrium (Metropolis et al., 1953)
- Example of a MCMC method
- Transition to a new state ($s_i \rightarrow s_j$) accepted if $\Delta E = E(s_j) - E(s_i) < 0$, otherwise allowed if $d < \exp(-\Delta E/T)$, where $d = rnd(0,1)$.

- Metropolis algorithm generates a Markov chain, whose transition probabilities converge to a unique and stable Boltzmann distribution (Beckerman, 1997).



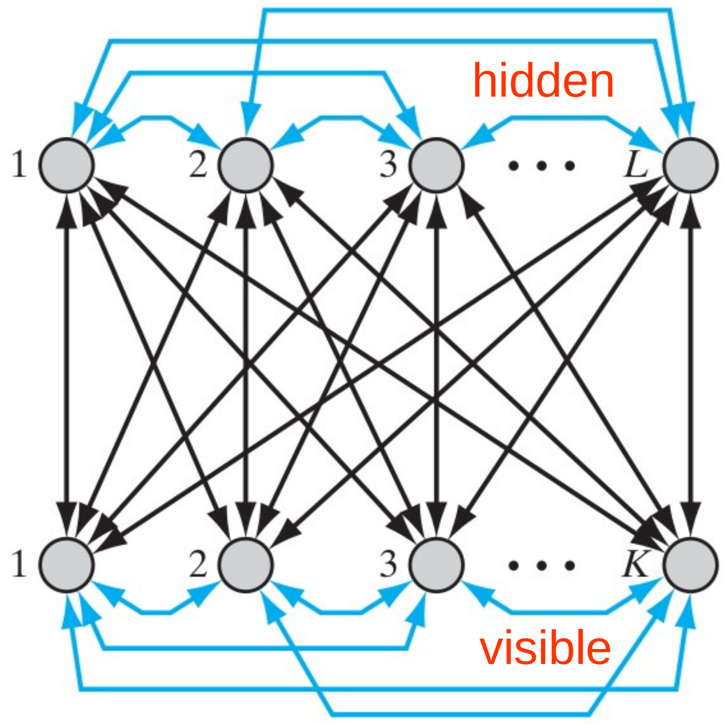
(wiki MCMC)

Simulated annealing

- Consider the problem of finding a low-energy system whose states are ordered in a Markov chain.
- For $T \to 0$, free energy $F \to \langle E \rangle$ which leads to a global minimum of average energy.
- SA = combination of two related ingredients:
  - schedule that determines the rate at which temperature is lowered;
  - Metropolis algorithm that iteratively finds the equilibrium distribution at each new temperature in the schedule by using the final state of the system at previous temperature as the starting point for new temperature.
- Kirkpatrick et al. (1983), Černý (1985)

Boltzmann machine

- fully connected recurrent two-layer network – stochastic binary model
- two different types of units:
- visible units – interface to the environment, can be clamped during training (positive phase)
- hidden units – always operate freely (negative phase)



- symmetric weight matrices
- Goal: Learn input patterns according to Boltzmann distribution.
- Two assumptions: (1) Each input pattern persists long enough to permit the network to reach thermal equilibrium. (2) There is no sequential structure in input patterns.

(Hinton, 1985)

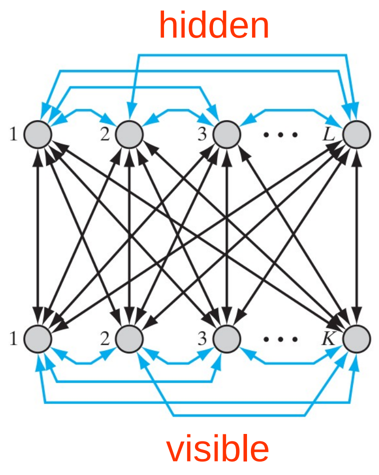
Learning in the Boltzmann machine

- Pattern completion: When a subset of visible neurons are clamped (to a desired pattern), the network can retrieve the activations of the remaining visible neurons, provided that it has learned the training distribution properly.
- State vector $\boldsymbol{s} = [\boldsymbol{s}_v , \boldsymbol{s}_h]$ (visible & hidden components)
- State vector $\boldsymbol{x}$ is the realization of the random variable $\boldsymbol{X}$
- Energy of BM: $E(\boldsymbol{s}) = -\frac{1}{2} \sum_i \sum_j w_{ij} s_i s_j$
- Probability of state $\boldsymbol{s}$: $P(\boldsymbol{S}=\boldsymbol{s}) = \frac{1}{Z} \exp(-\frac{E(\boldsymbol{s})}{T})$ $\quad Z = \sum_{\boldsymbol{s}'} \exp(-\frac{E(\boldsymbol{s}')}{T})$
- Two phases alternated during training: positive and negative
- Criterion for learning: maximize marginal probability over data

Learning rule in the BM

- Training patterns are assumed statistically independent
- Cost function: $L(\boldsymbol{w}) = \log \prod_{\boldsymbol{s}_v \in trn} P(\boldsymbol{s}_v) = \sum_{\boldsymbol{s}_v \in trn} \log P(\boldsymbol{s}_v)$
- Probability of finding visible neurons in state $\boldsymbol{s}_v$, for any $\boldsymbol{s}_h$, is

$$P(\boldsymbol{s}_v) = \frac{1}{Z} \sum_{\boldsymbol{s}_h} \exp\left(-\frac{E(\boldsymbol{s})}{T}\right) \qquad Z = \sum_{\boldsymbol{s}} \exp\left(-\frac{E(\boldsymbol{s})}{T}\right)$$



- Then (maximize *L*)

$$L(\boldsymbol{w}) = \sum_{\boldsymbol{s}_v \in trn} \left(\log \sum_{\boldsymbol{s}_h} \exp\left(-\frac{E(\boldsymbol{s})}{T}\right) - \log \sum_{\boldsymbol{s}} \exp\left(-\frac{E(\boldsymbol{s})}{T}\right) \right)$$

$$\frac{\partial L(\boldsymbol{w})}{\partial w_{ij}} = \frac{1}{T} \sum_{\boldsymbol{s}_v \in trn} \left(\sum_{\boldsymbol{s}_h} P(\boldsymbol{s}_h | \boldsymbol{s}_v) s_i s_j - \sum_{\boldsymbol{s}} P(\boldsymbol{s}) s_i s_j \right) \propto \Delta w_{ij}$$

$$\rho_{ij}^{+} = \langle s_i s_j \rangle^{+} \qquad \rho_{ij}^{-} = \langle s_i s_j \rangle^{-}$$

$$\Delta w_{ij} = \eta(T)(\rho_{ij}^{+} - \rho_{ij}^{-})$$

(Haykin, 2009)

Deep Belief Networks

- Based on restricted Boltzmann Machine (RBM) – no lateral connections, alternating:
- RBMs can be stacked and trained in a greedy manner, on a layer-by-layer basis
- Update the hidden states all in parallel, given the (clamped) visible states (sigma logistic f.)

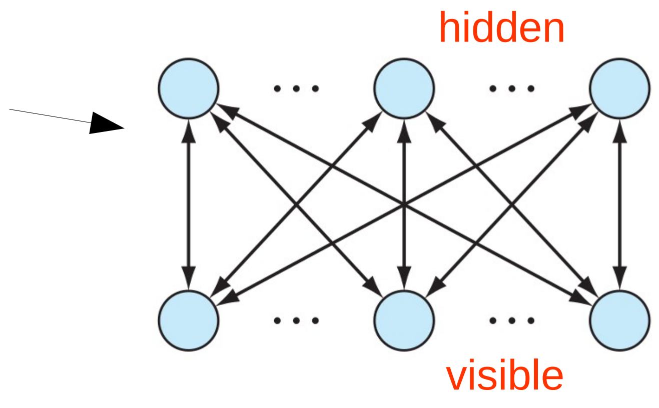
$$p(h_j{=}1|\mathbf{s}_v) = \sigma(b_j + \textstyle\sum_i v_i w_{ij})$$

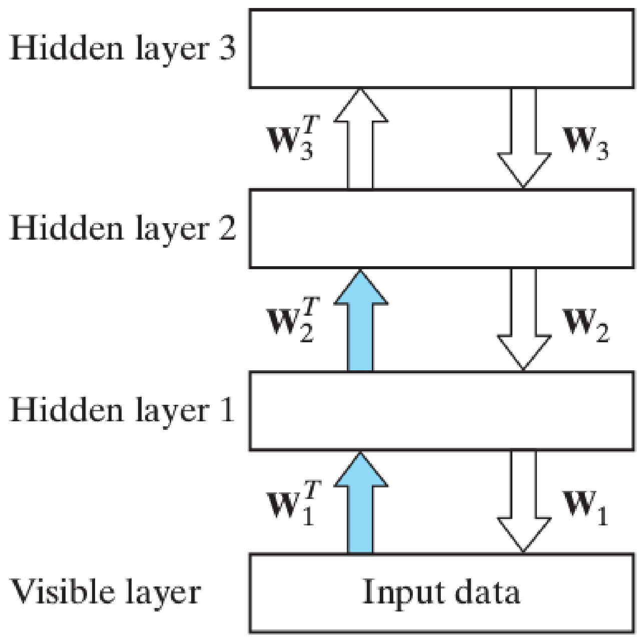
- Update the visible states all in parallel, given the (clamped) hidden states

$$p(v_i{=}1|\mathbf{s}_h) = \sigma(a_i + \textstyle\sum_j h_j w_{ij})$$

- Learning rules: (data – reconstruction)

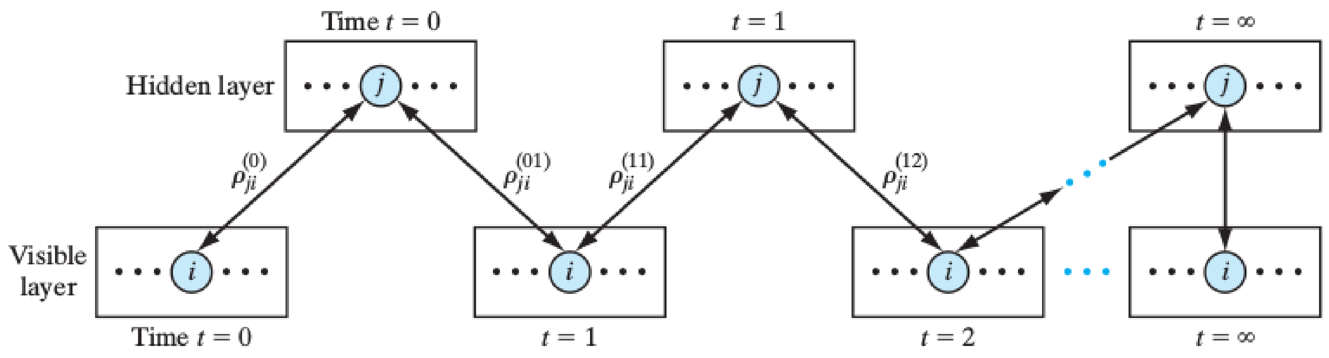
$$\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstr}$$





(Hinton, 2006)

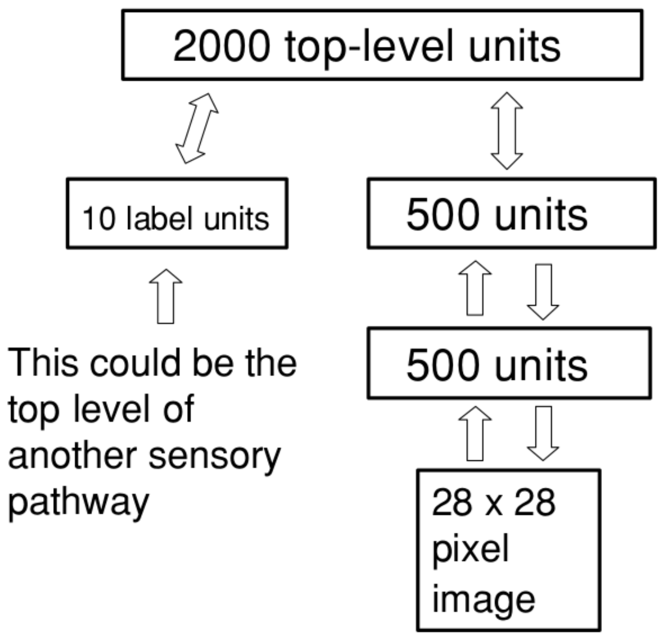
DBN (ctd)



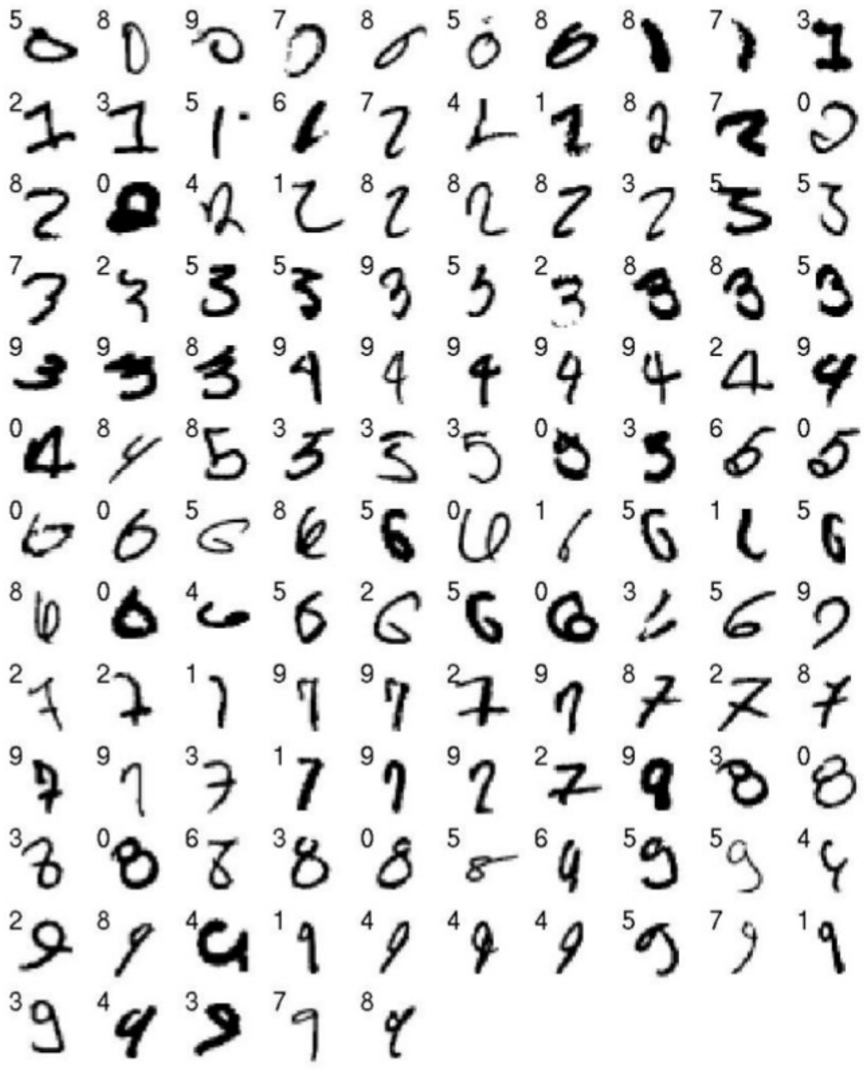
- Contrastive divergence as an approximation to ML learning
- RBM is trained directly on the input data, allowing stochastic neurons in the hidden layer to capture important features
  - after enough iterations, the visible and hidden vectors are sampled from the stationary distribution
- trained features are then treated as “input data,” for 2nd RBM => learning features of features
- DBNs learns joint distribution b/w observed data and hidden layers

DBN application

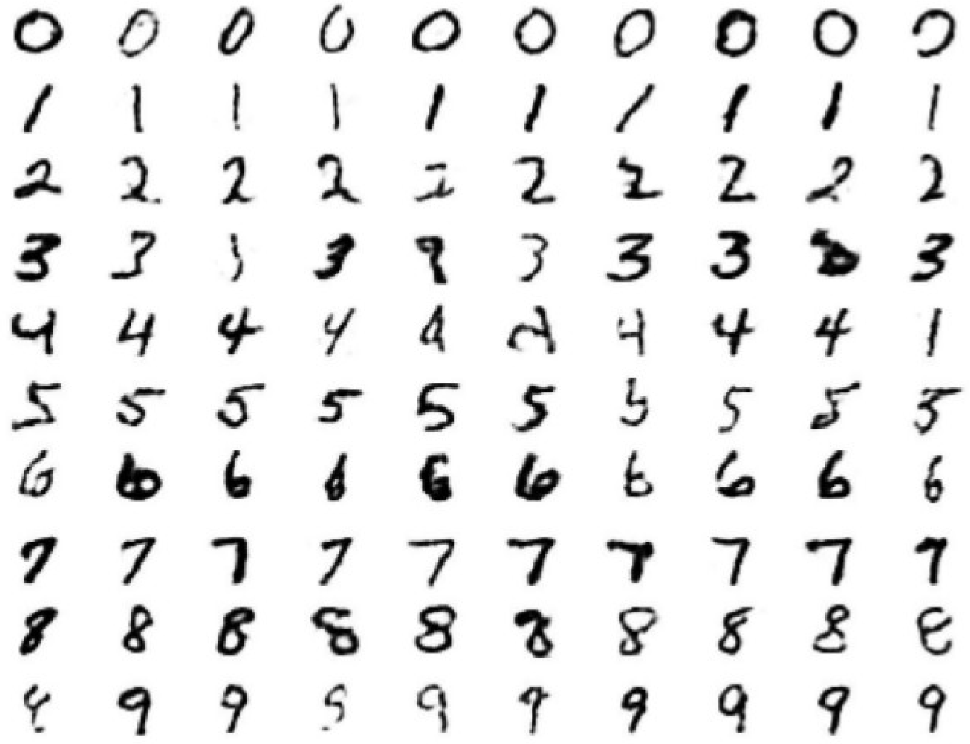
- Top layer learns the joint distribution of handwritten digit images and their labels
- 44000 training images, 10000 testing
- 1.25% error on testing data (best of the time)
- Input units are real valued, hidden units binary (stochastic)
- Labels provided during training the top layer
- triggered the boom of deep learning



DBN on MNIST dataset



Errors: Each case is labeled by the network’s guess.



10 samples for each class from the generative model with a particular label clamped on.

Summary

- Training generative models with hidden units is a powerful way to make models represent the world given by the training data.
- By learning a model $p_{\text{model}}(\boldsymbol{x})$ and representation $p_{\text{model}}(\boldsymbol{x}|\boldsymbol{h})$, a generative model can provide answers to many inference problems about the relationships between input variables in $\boldsymbol{x}$, and can offer different ways of representing $\boldsymbol{x}$ by taking expectations of $\boldsymbol{h}$ at different layers of the hierarchy.
- Generative models hold the promise to provide AI systems with a framework for all the many different intuitive concepts they need to understand, giving them the ability to reason about these concepts in the face of uncertainty.