Machine Learning for Physicists
Lecture 11

Summer 2017

University of Erlangen-Nuremberg
Florian Marquardt

Function/Image representation

Image classification [Handwriting recognition]

Convolutional nets

Autoencoders

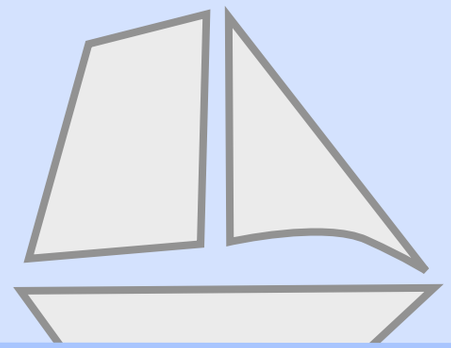
Visualization by dimensional reduction

Recurrent networks

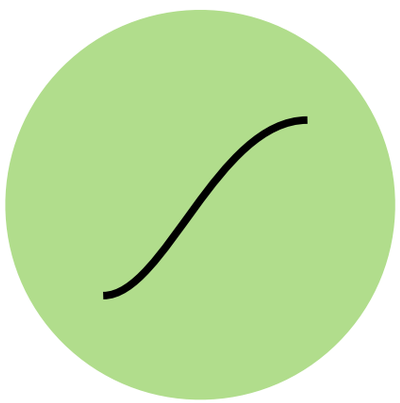
Word vectors

Reinforcement learning

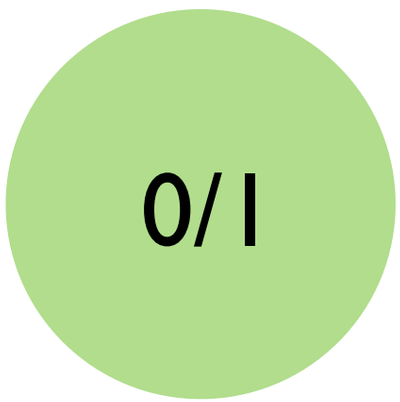
Connections to physics



Neural networks and spin models



Artifical neuron



Bit

Spin

Neural networks with stochastic transitions, and with some energy functional similar to spin models in physics; e.g. as described by Hopfield and others starting from the 80s

Modeling probability distributions

Goal: Use a neural network to generate previously unseen examples, according to the probability distribution of training samples

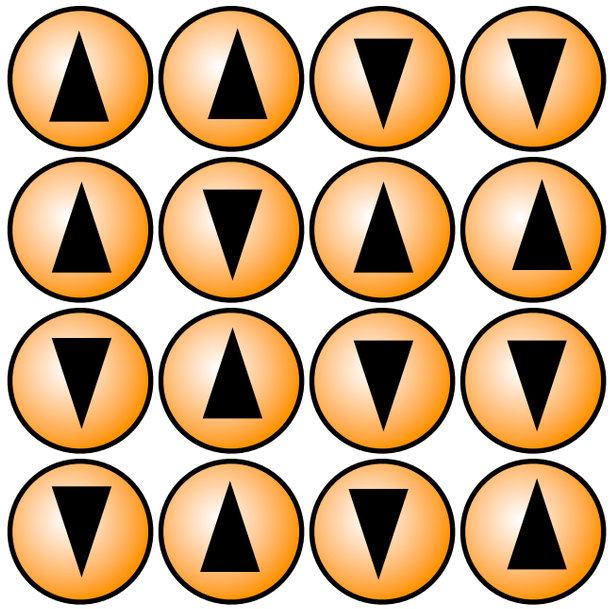
One example already mentioned in these lectures: generating new random (but kind-of reasonable) text after seeing lots of it

Example: Generate new images after looking at many, generate handwritten text

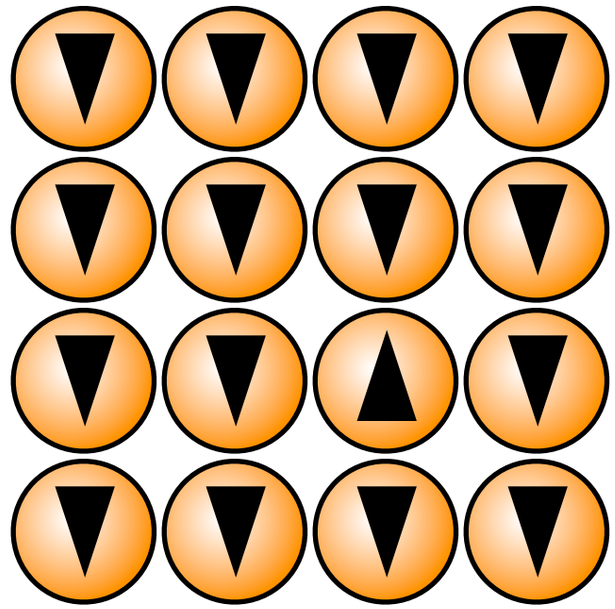
The solution will exploit the connection between neural networks and the statistical physics of spin models!

Boltzmann-Gibbs distribution

Probabilities of states of a physical system, in thermal equilibrium?



energy high: less likely



energy low: more likely

$$P(s) = \frac{1}{Z} e^{-\frac{E(s)}{k_B T}}$$

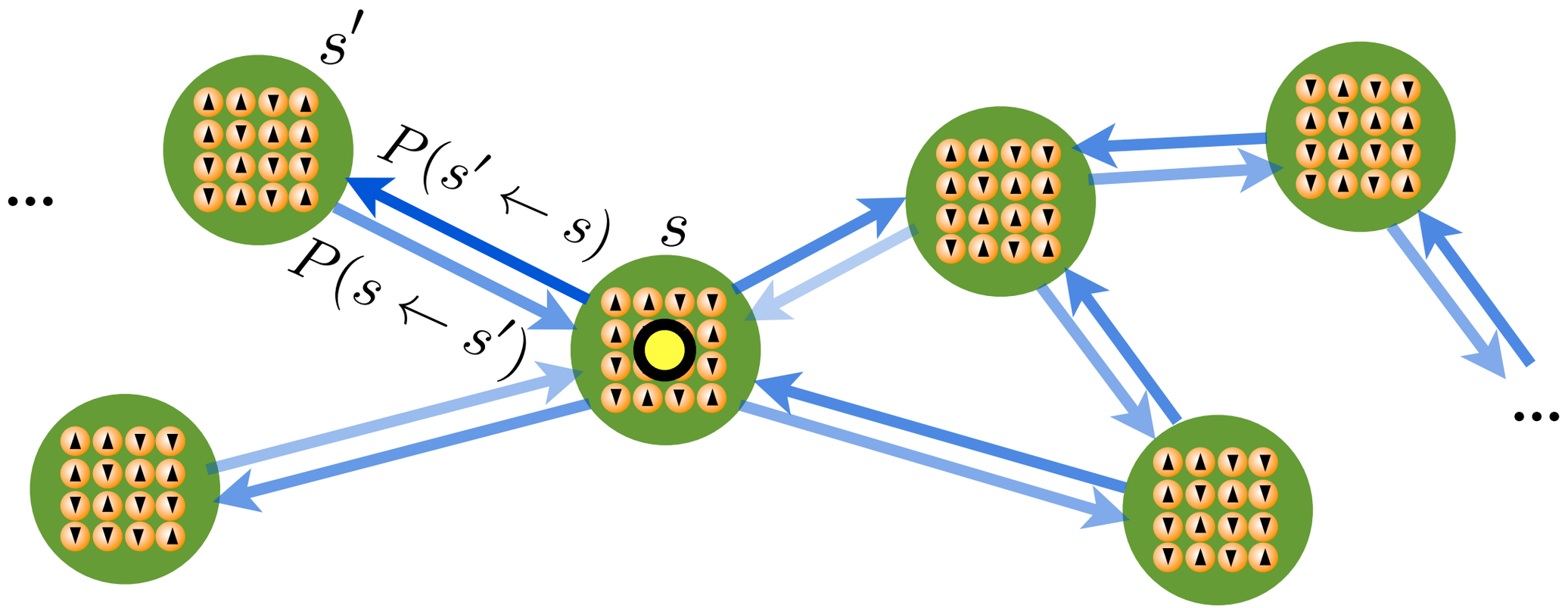
probability for state s, in thermal equilibrium

$$Z = \sum_{s'} e^{-\frac{E(s')}{k_B T}}$$

Z for normalization: "partition function"

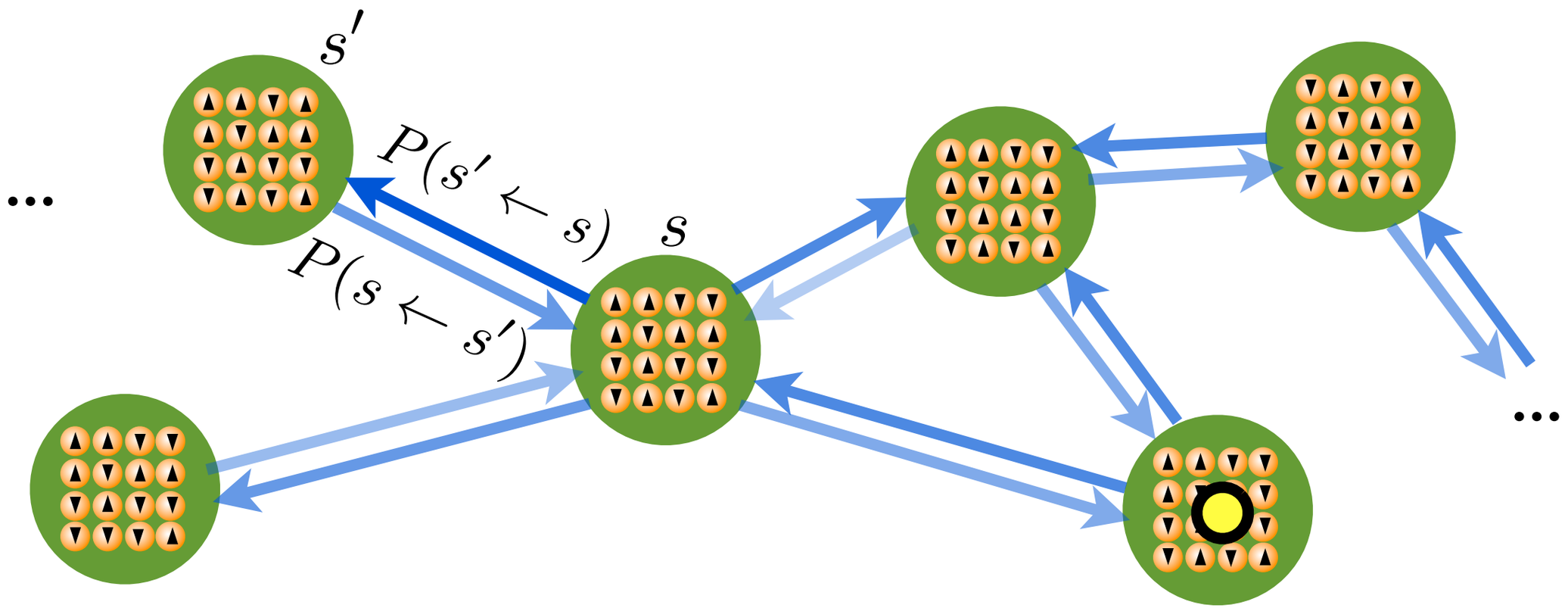
Problem: for a many-body system, exponentially many states (for example 2^N spin states). Cannot go through all of them!

Monte Carlo approach



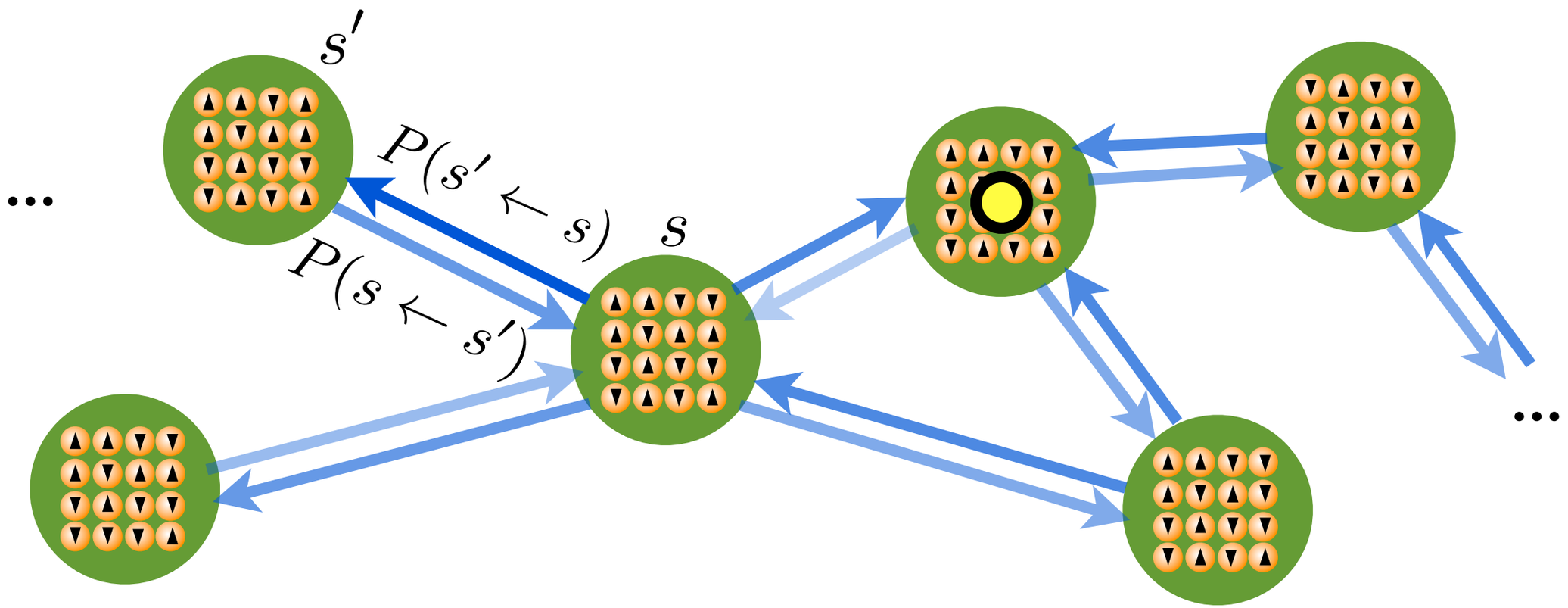
Place system in some state, make stochastic transitions to other states (with prescribed transition probabilities)

Monte Carlo approach



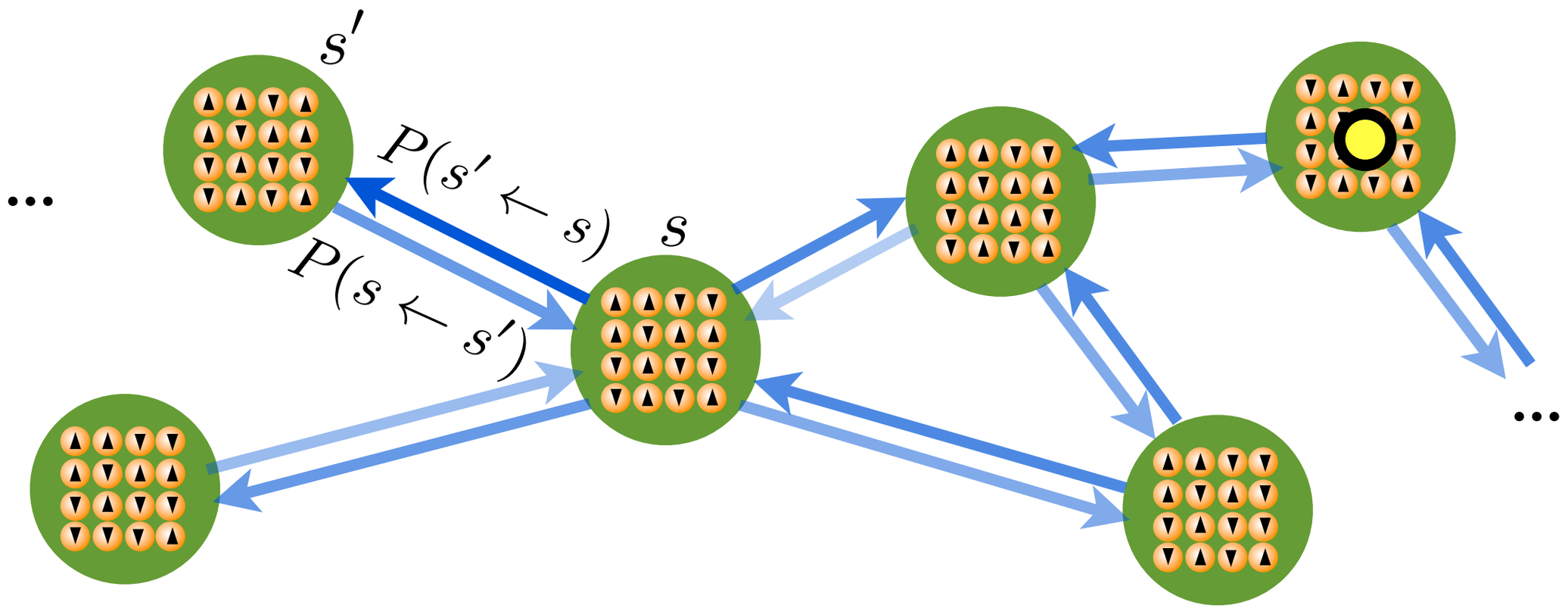
Place system in some state, make stochastic transitions to other states (with prescribed transition probabilities)

Monte Carlo approach



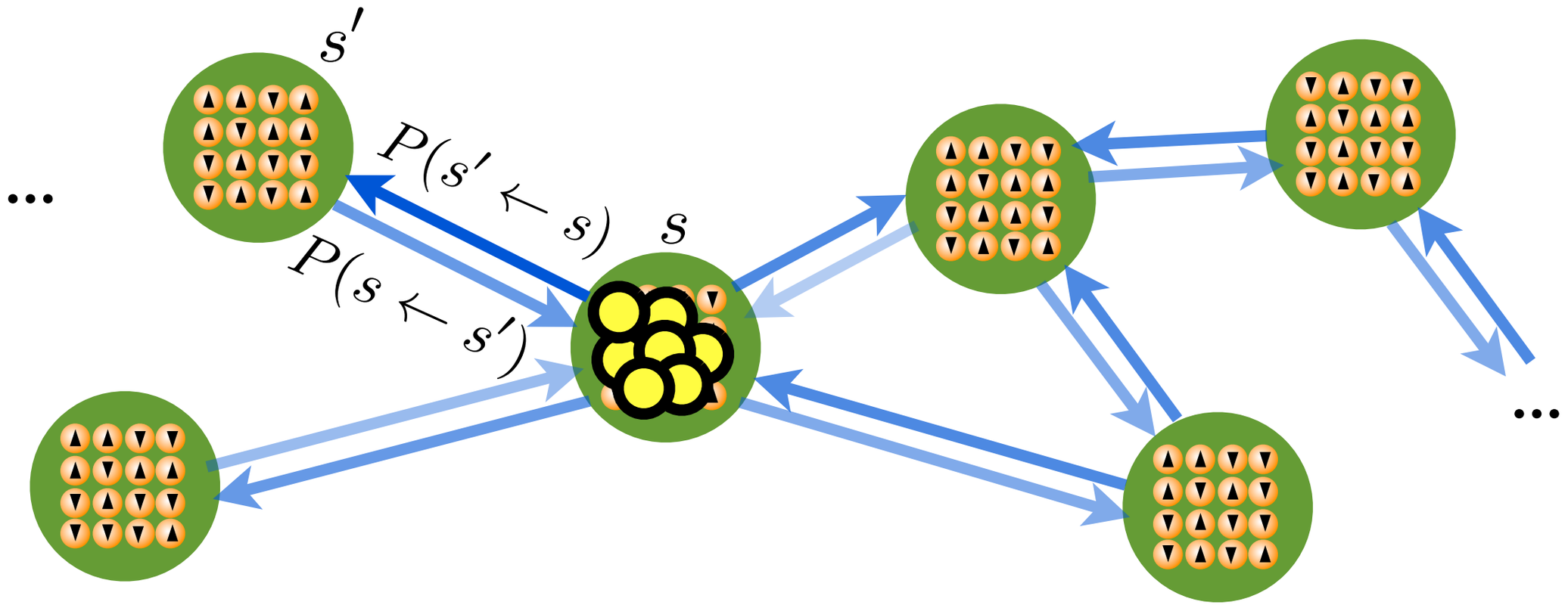
Place system in some state, make stochastic transitions to other states (with prescribed transition probabilities)

Monte Carlo approach



Place system in some state, make stochastic transitions to other states (with prescribed transition probabilities)

Monte Carlo approach



Time evolution of ensemble?

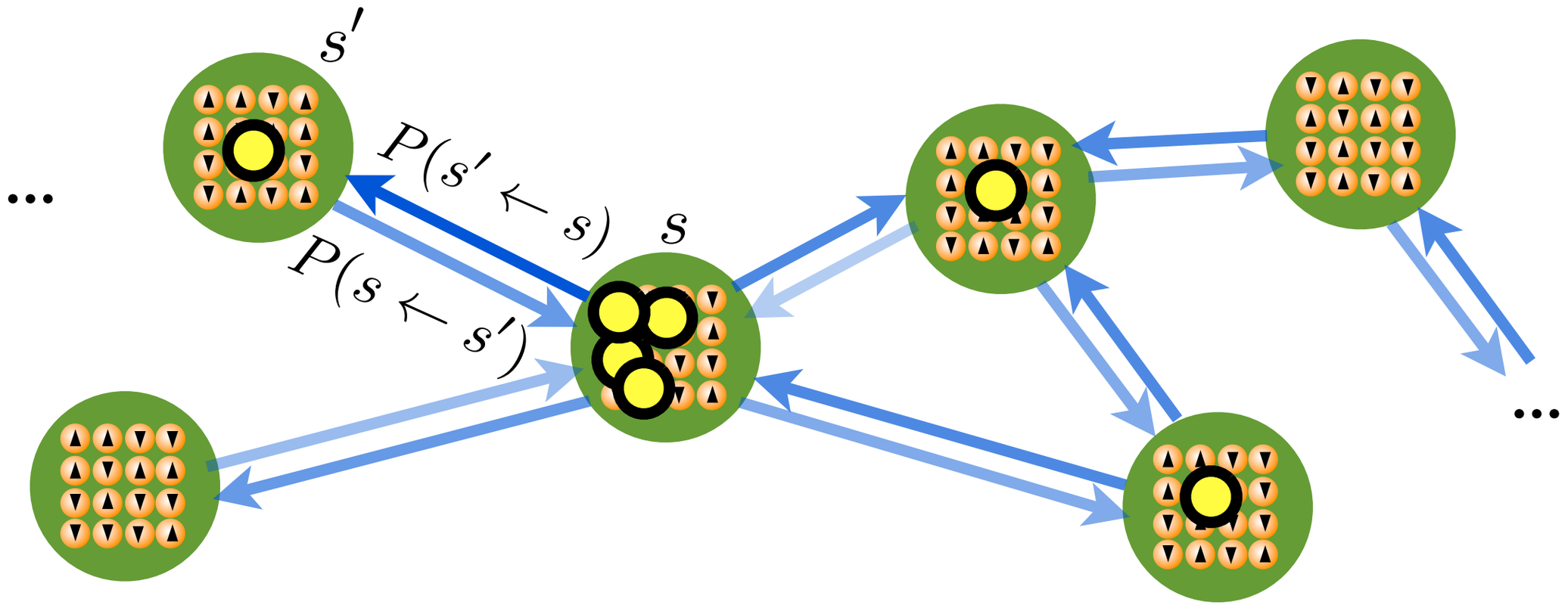
$$\Delta P(s) = \sum_{s'} P(s \leftarrow s')P(s') - P(s' \leftarrow s)P(s)$$

"IN" "OUT"

change in one time-step

P(s) = probability to find the system in state s (or: fraction of ensemble in this state)

Monte Carlo approach



Time evolution of ensemble?

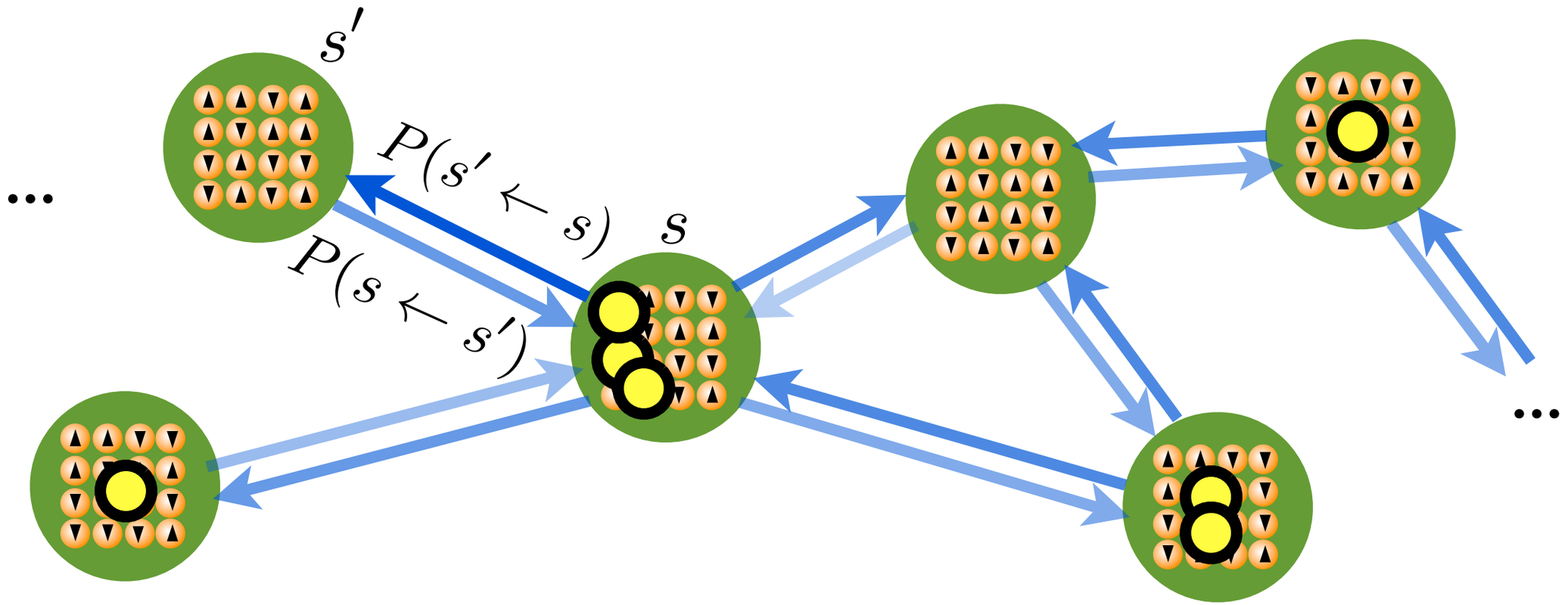
$$\Delta P(s) = \sum_{s'} P(s \leftarrow s')P(s') - P(s' \leftarrow s)P(s)$$

"IN" "OUT"

change in one time-step

P(s) = probability to find the system in state s (or: fraction of ensemble in this state)

Monte Carlo approach



Time evolution of ensemble?

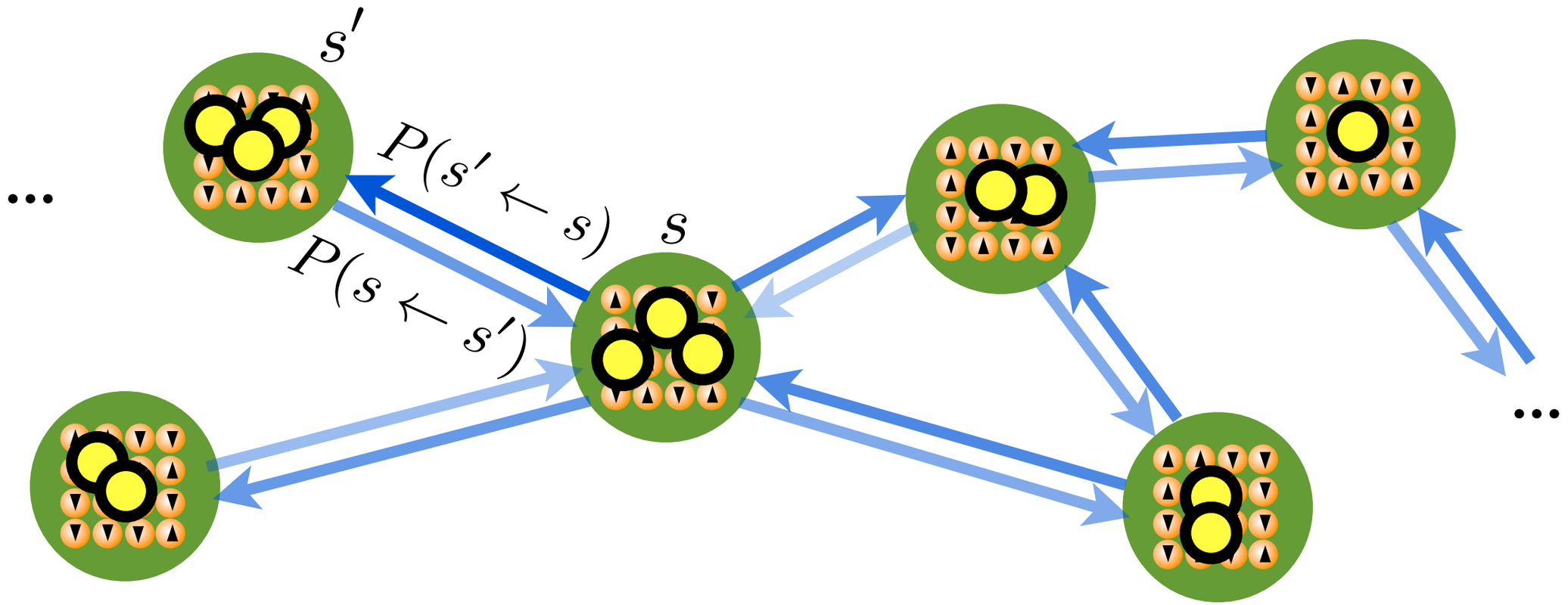
$$\Delta P(s) = \sum_{s'} P(s \leftarrow s')P(s') - P(s' \leftarrow s)P(s)$$

"IN" "OUT"

change in one time-step

P(s) = probability to find the system in state s (or: fraction of ensemble in this state)

Monte Carlo approach



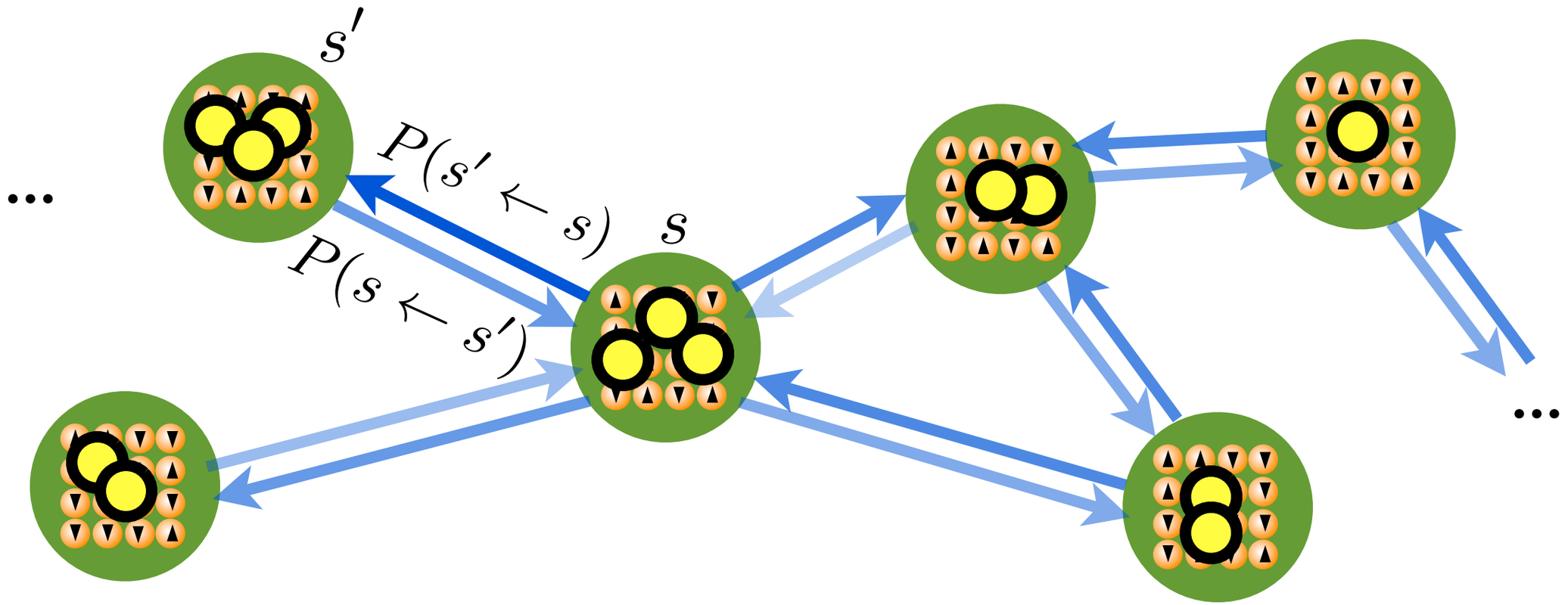
At long times: stable steady state distribution

If we have "detailed balance", i.e. if there exists a distribution P(s), such that for any pair of states:

$$\frac{P(s \leftarrow s')}{P(s' \leftarrow s)} = \frac{P(s)}{P(s')}$$

then P(s) is the long-time distribution!

Monte Carlo approach

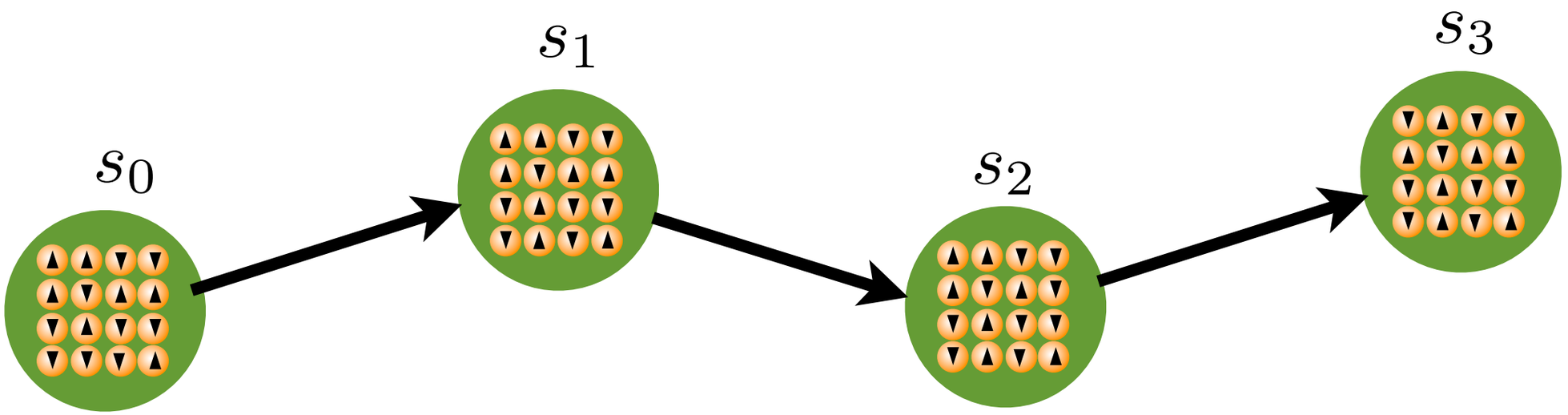


Monte Carlo for thermal equilibrium: choose transition probabilities such that P(s) will be the Boltzmann distribution!

$$\frac{P(s \leftarrow s')}{P(s' \leftarrow s)} = e^{\frac{E(s')-E(s)}{k_B T}}$$

example Metropolis algorithm: pick random spin, calculate energy change for spin flip. Do the flip if it lowers the energy. If the energy increases, only flip with probability $\exp(-\Delta E/k_B T)$

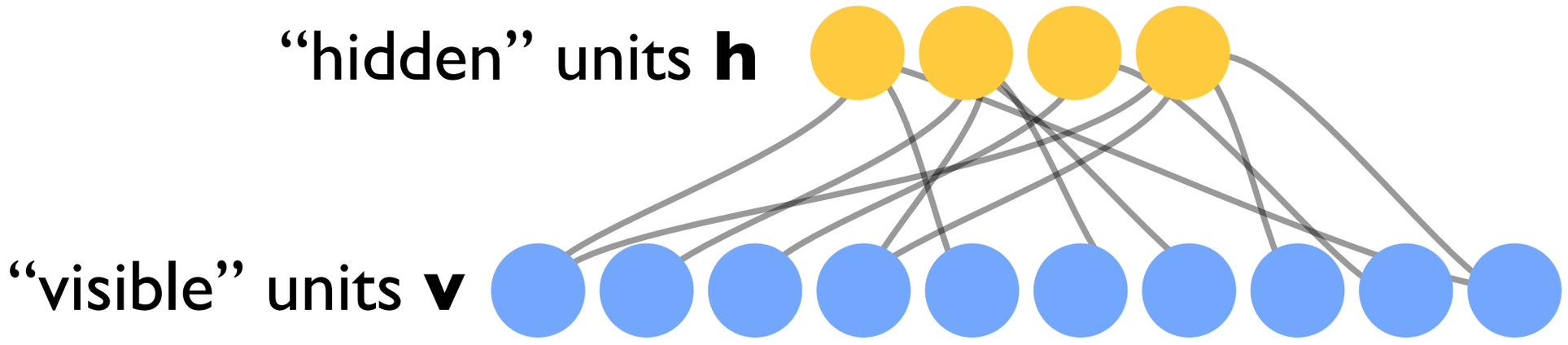
Markov chain



The sequence of visited states forms a so-called “Markov chain”

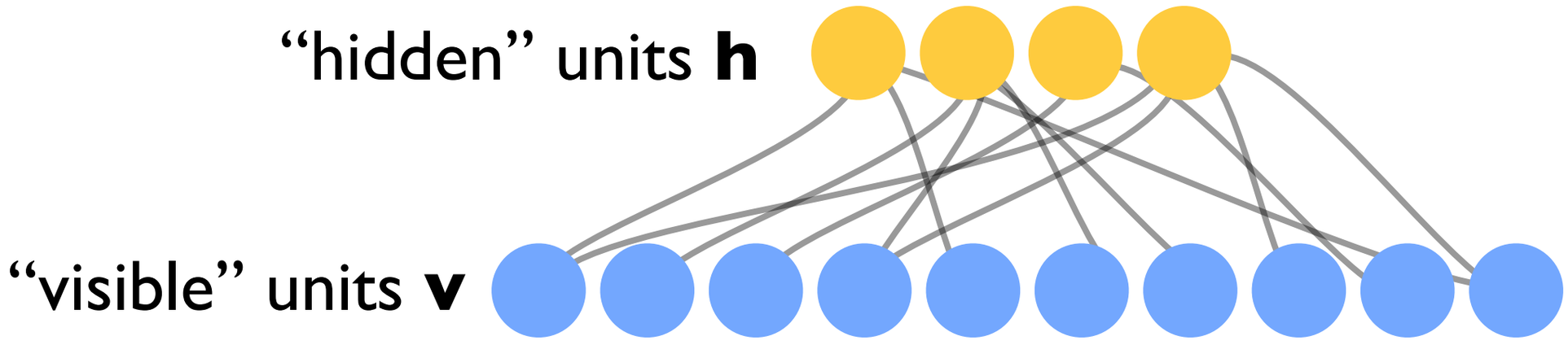
Markov = transitions without memory

Restricted Boltzmann Machine



Each "unit" is like a spin (or a bit) that can be 0 or 1

Restricted Boltzmann Machine



Each "unit" is like a spin (or a bit) that can be 0 or 1

Define "energy" (we will set $k_B T = 1$)

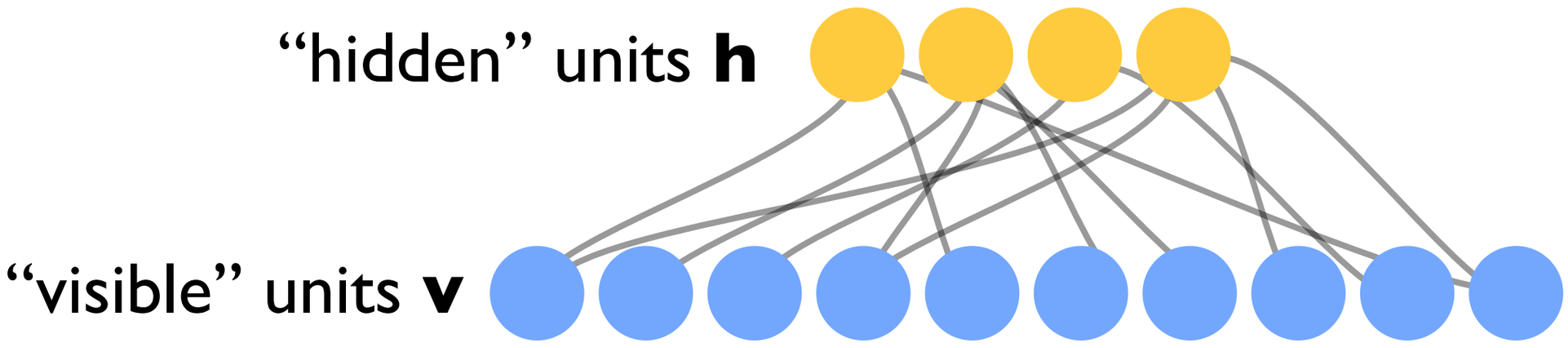
$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$

"restricted": no coupling v-v or h-h

w: couplings (weights)

see G. Hinton's guide: http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf

Restricted Boltzmann Machine



Each "unit" is like a spin (or a bit) that can be 0 or 1

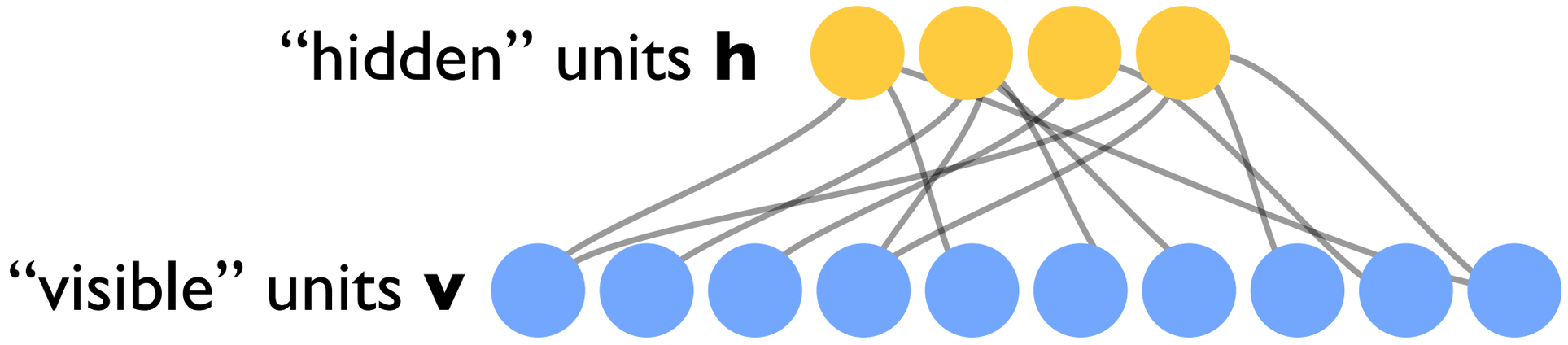
$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \qquad Z = \sum_{v,h} e^{-E(v,h)}$$

$$P(v) = \sum_{h} P(v,h)$$

Goal: adapt weights (and biases), such that the probability distribution of a set of training examples is approximately reproduced by P(v)

$$P(v) \approx P_0(v)$$ —— from training samples

Restricted Boltzmann Machine



Each "unit" is like a spin (or a bit) that can be 0 or 1

Interpretation: the 'hidden units' represent categories of data (e.g. "dog+white+big")

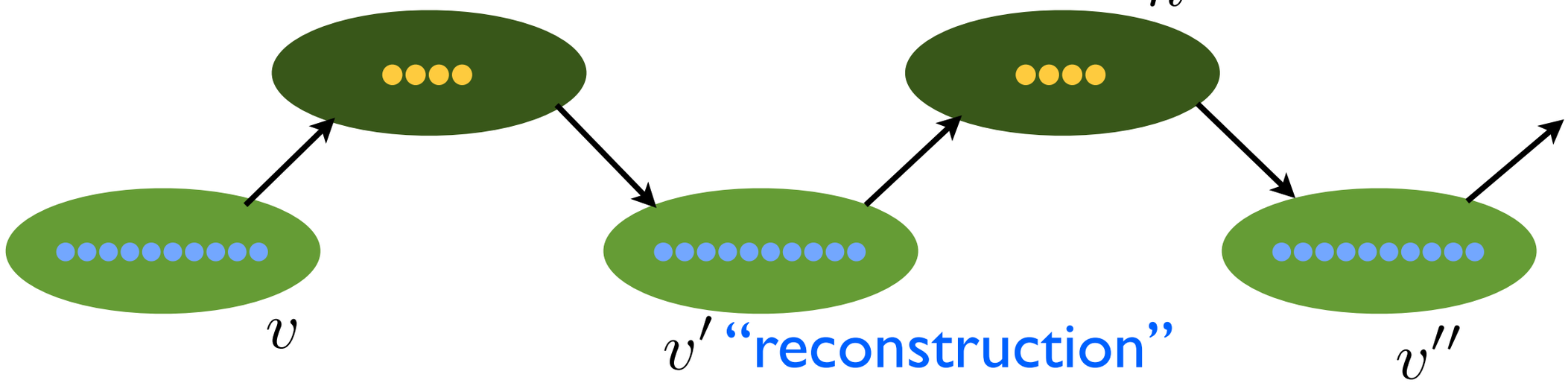
Building a Markov chain

Instead of the full state s=(v,h): Consider alternating transitions between v and h states

Set:

$$P(h \leftarrow v) = P(h|v) = \frac{P(v,h)}{P(v)}$$

$$P(v \leftarrow h) = P(v|h) = \frac{P(v,h)}{P(h)}$$



These transition probabilities fulfill detailed balance!

$$\frac{P(h \leftarrow v)}{P(v \leftarrow h)} = \frac{P(h)}{P(v)}$$

Thus: P(v) [and P(h)] are the steady-state distributions!

Building a Markov chain

$$ZP(v) = \sum_h e^{-E(v,h)} = \sum_h e^{\sum_i a_i v_i + \sum_j b_j h_j + \sum_{i,j} v_i h_j w_{ij}}$$

$$= e^{\sum_i a_i v_i} \Pi_j (1 + e^{z_j})$$

with: $z_j = b_j + \sum_i v_i w_{ij}$

where we used: $e^{\sum_j X_j} = \Pi_j e^{X_j}$ $\sum_h \cdots = \sum_{h_0=0,1} \sum_{h_1=0,1} \sum_{h_2=0,1} \cdots$

Therefore:

$$P(h|v) = \frac{e^{-E(v,h)}}{ZP(v)} = \Pi_j \frac{e^{z_j h_j}}{1 + e^{z_j}}$$

Product of probabilities! All the h_j are independently distributed, with probabilities:

$$P(h_j = 1|v) = \frac{e^{z_j}}{1 + e^{z_j}} = \sigma(z_j)$$

sigmoid

$$P(h_j = 0|v) = 1 - \sigma(z_j)$$

Building a Markov chain

Given some visible-units state vector v, calculate the probabilities

$$P(h_j = 1|v) = \frac{e^{z_j}}{1 + e^{z_j}} = \sigma(z_j)$$

sigmoid

Then assign 1 or 0, according to these probabilities, to obtain the new hidden state vector h

Similarly, go from h to a new v', using:

$$P(v'_i = 1|h) = \sigma(z'_i)$$

$$z'_i = a_i + \sum_j w_{ij} h_j$$

Updating the weights

Goal: adapt weights (and biases), such that the probability distribution of a set of training examples is approximately reproduced by P(v)

$$P(v) \approx P_0(v)$$ —— from training samples

Minimize the categorical cross-entropy

$$C = -\sum_v P_0(v) \ln P(v)$$

But now (unlike earlier examples), there are exponentially many values for v, so we cannot simply have a network output P(v) for all v. Still, let us take the derivative of C with respect to the weights w!

Updating the weights

$$C = -\sum_v P_0(v) \ln P(v)$$

$$\frac{\partial}{\partial w_{ij}} \ln P(v) = \frac{\frac{\partial}{\partial w_{ij}} \sum_h P(v,h)}{\sum_h P(v,h)}$$

$$Z = \sum_{v',h'} e^{-E(v',h')}$$

$$= \frac{\frac{\partial}{\partial w_{ij}} \sum_h e^{-E(v,h)}}{\sum_h e^{-E(v,h)}} - \frac{\frac{\partial}{\partial w_{ij}} \frac{1}{Z}}{\frac{1}{Z}}$$

$$= \frac{\sum_h v_i h_j e^{-E(v,h)}}{\sum_h e^{-E(v,h)}} - \frac{\sum_{v',h'} v'_i h'_j e^{-E(v',h')}}{Z}$$

overall:

$$\sum_v P_0(v) \frac{\partial}{\partial w_{ij}} \ln P(v) = \sum_{v,h} v_i h_j P(h|v) P_0(v) - \sum_{v',h'} v'_i h'_j P(v',h')$$

Updating the weights

$$\sum_v P_0(v) \frac{\partial}{\partial w_{ij}} \ln P(v) = \sum_{v,h} v_i h_j P(h|v) P_0(v) - \sum_{v',h'} v'_i h'_j P(h'|v') P(v')$$

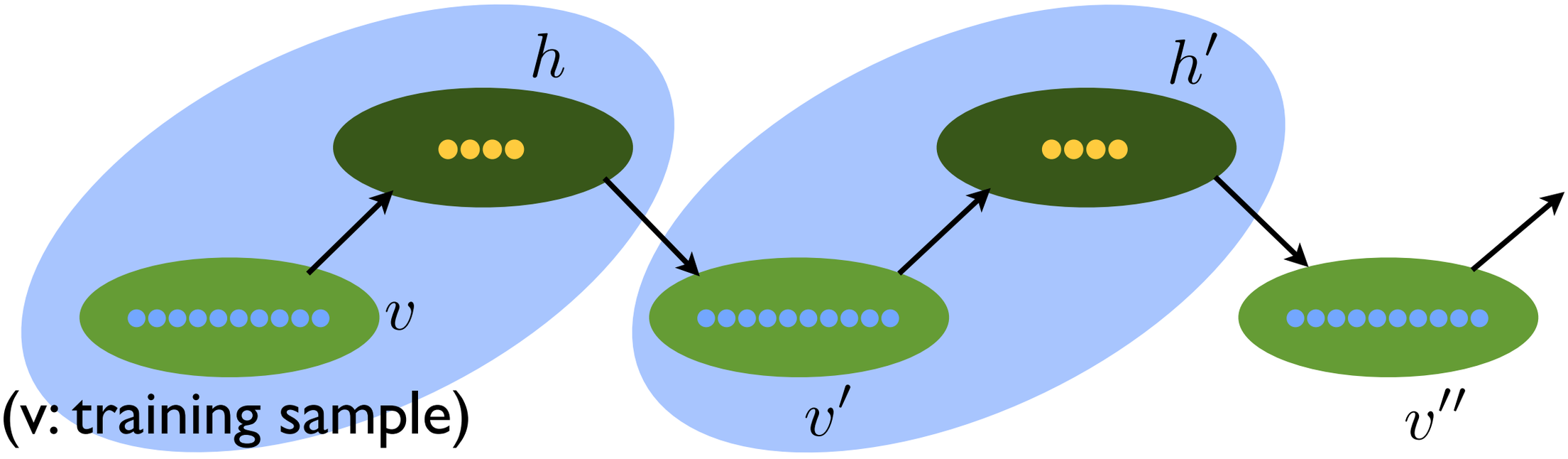
easy: draw one training sample v, then do one Markov chain step from v to h; average over all samples v

hard: need to average over the correct distribution P(v) belonging to the Boltzmann machine!

Updating the weights

$$\sum_v P_0(v) \frac{\partial}{\partial w_{ij}} \ln P(v) = \sum_{v,h} v_i h_j P(h|v) P_0(v) - \sum_{v',h'} v'_i h'_j P(h'|v') P(v')$$

Could obtain P(v) by running the Markov chain for really long times! Very expensive!



(v: training sample)

Rough approximation, used in practice: Just take v',h' from the second pair of the chain! [For better approx.: can take a pair further down the chain]

$$\Delta w_{ij} = \eta(\langle v_i h_j \rangle - \langle v'_i h'_j \rangle)$$

(averaged over a batch of training samples v starting the chain)

Updating the weights

$$\Delta w_{ij} = \eta(\langle v_i h_j \rangle - \langle v'_i h'_j \rangle)$$

(averaged over a batch of training samples v starting the chain)

"Contrastive Divergence" (CD) algorithm by G. Hinton

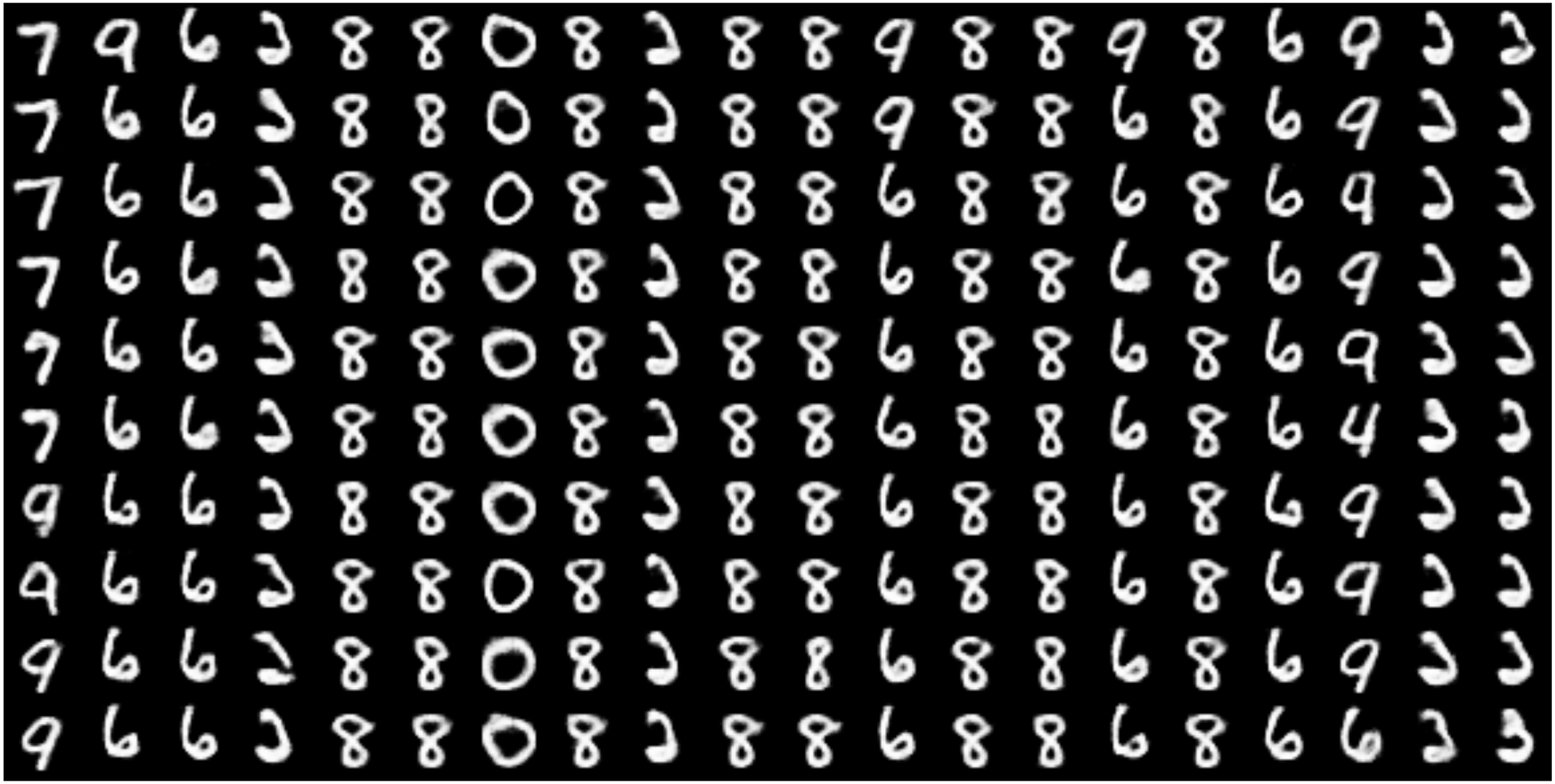
Note: At least we can claim that $P_0(v) = P(v)$ would be a fixed point of this update rule, since then the two averages on the right-hand-side yield identical results. Of course, usually the restricted Boltzmann machine will not be able to reach this point, since it cannot represent arbitrary P(v).

$$\Delta a_i = \eta(\langle v_i \rangle - \langle v'_i \rangle)$$

$$\Delta b_j = \eta(\langle h_j \rangle - \langle h'_j \rangle)$$

Restricted Boltzmann Machine for MNIST

example from http://deeplearning.net/tutorial/rbm.html

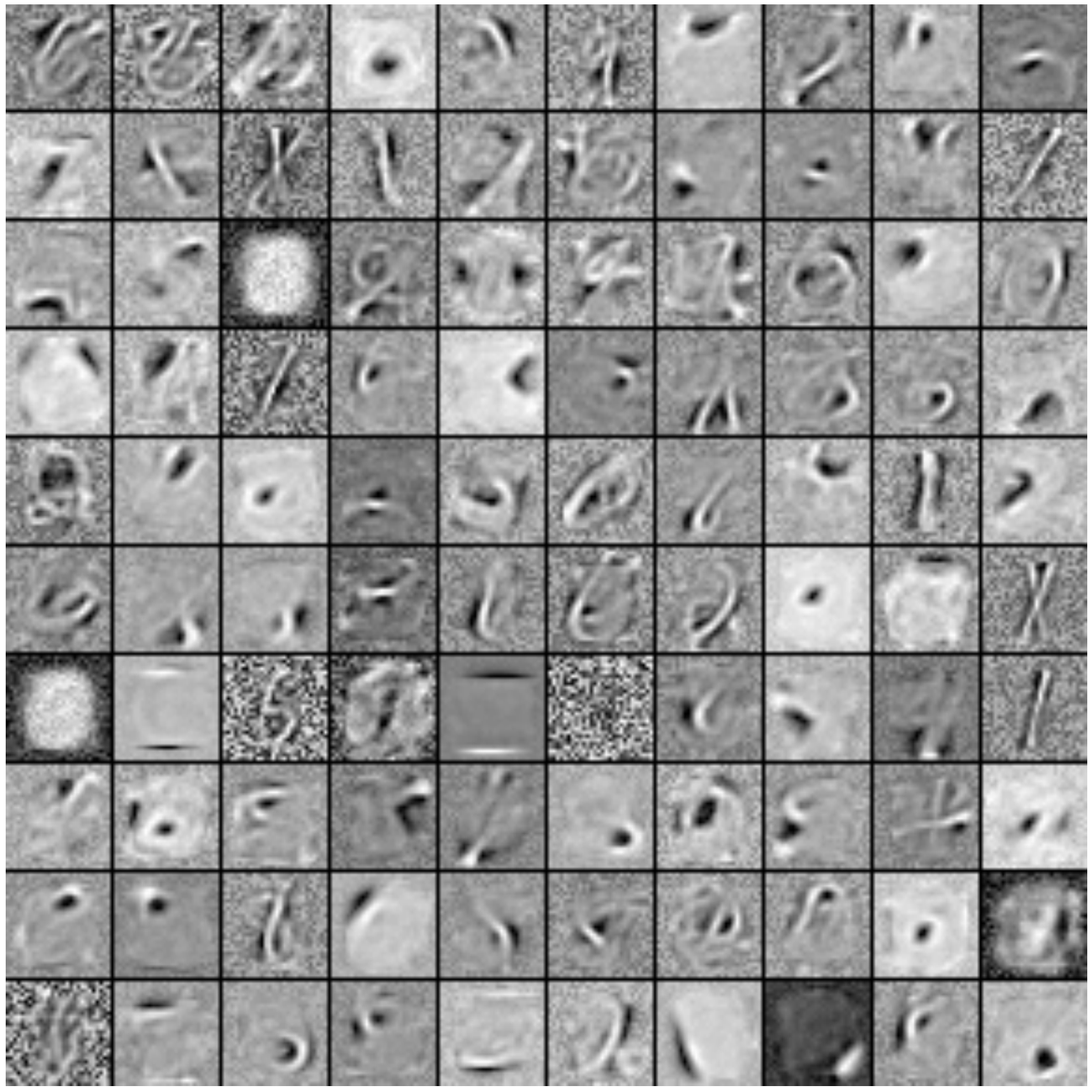


Each column: a different, independent Markov chain

Restricted Boltzmann Machine for MNIST

example from http://deeplearning.net/tutorial/rbm.html

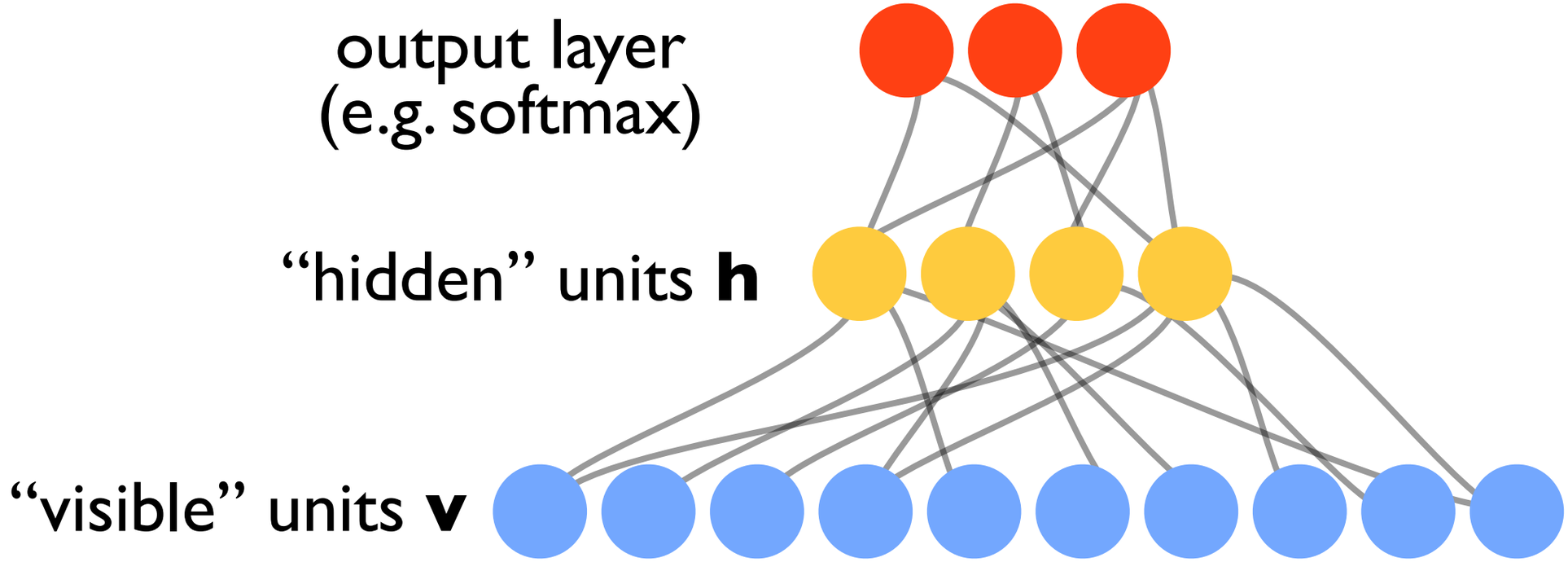
The learned weights for the 100 hidden units



RBM as a starting point

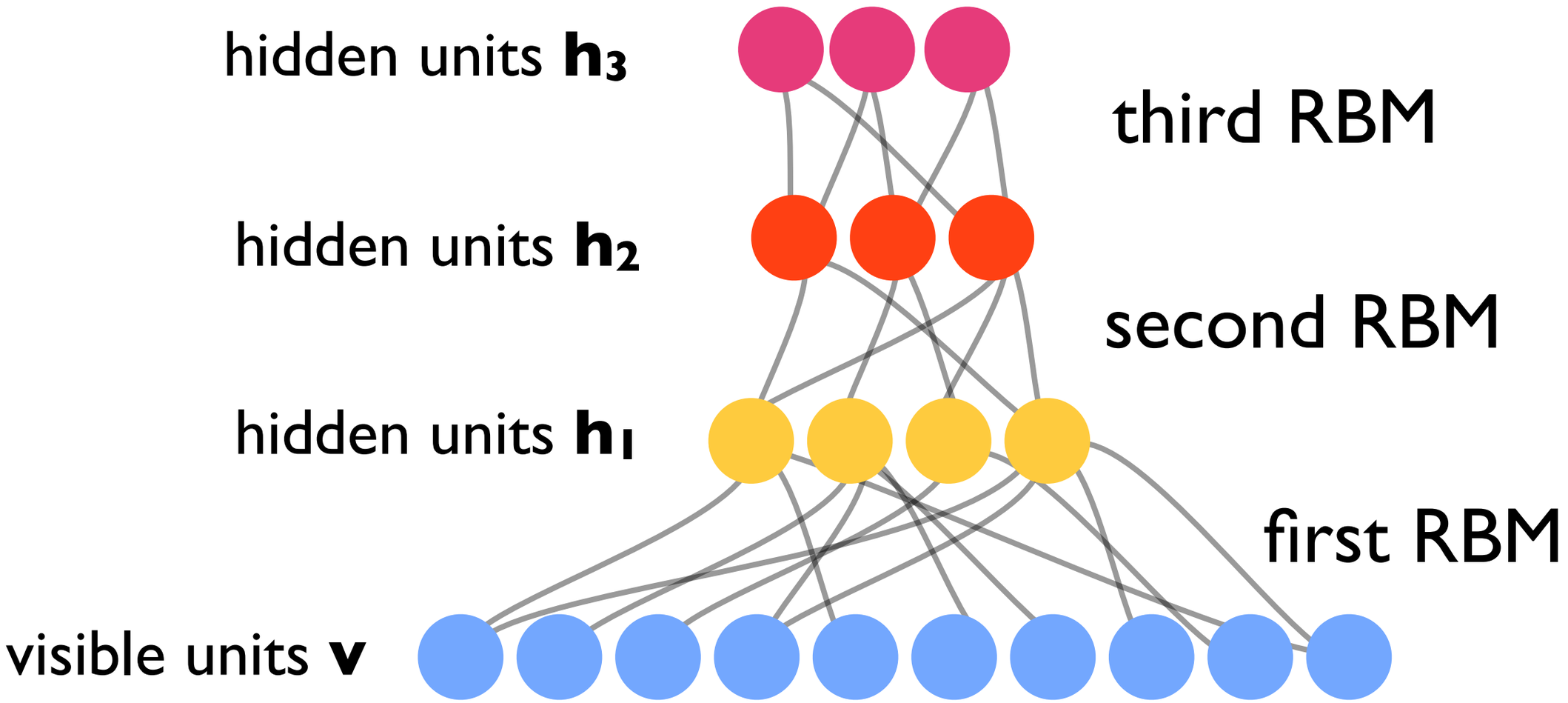
First train RBM, then connect hidden layer to some output layer for supervised learning of classification

Idea: RBM provides unsupervised learning of important features in the training set (pre-training)



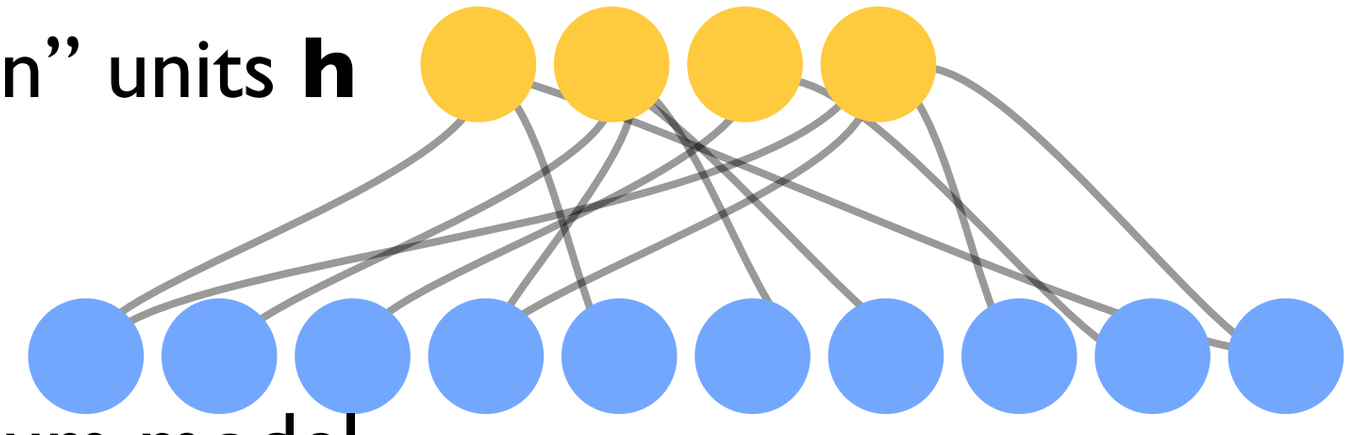
Deep belief networks

Stack RBMs: First train a simple RBM, then use its hidden units as input to another RBM, and so on



Afterwards, fine-tune weights, e.g. by supervised learning

Application to Quantum Physics



Try to solve a quantum many-body problem (quantum spin model) using the following **variational ansatz** for the wave function amplitudes:

$$\Psi(\mathcal{S}) = \sum_h e^{\sum_j a_j \sigma_j^z + \sum_i b_i h_i + \sum_{ij} h_i \sigma_j^z W_{ij}}$$

Carleo & Troyer, Science 2017

$\mathcal{S} = (\sigma_1^z, \sigma_2^z, \ldots, \sigma_N^z)$ one basis state in the many-body Hilbert space

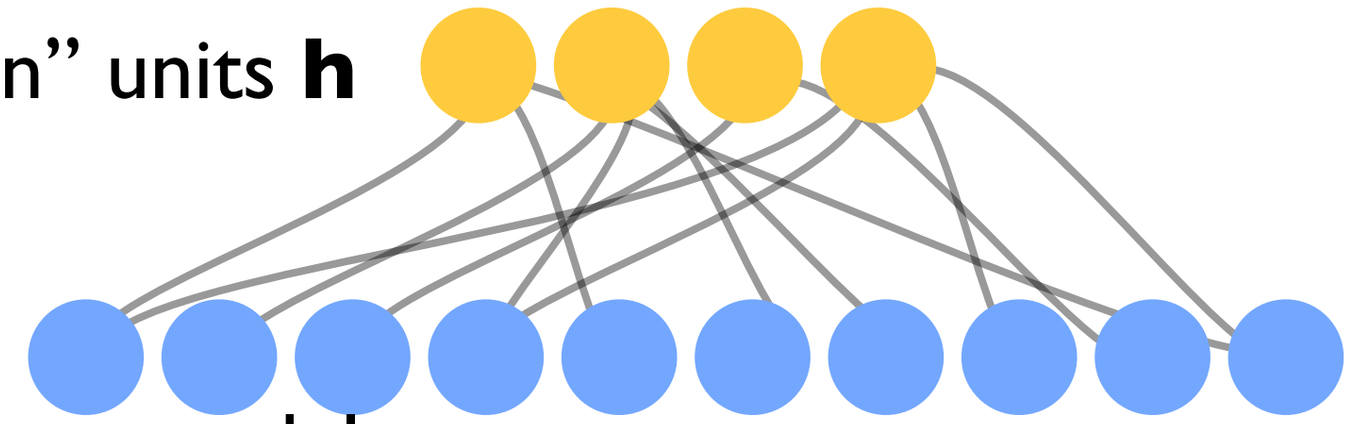
$\sigma_j^z = \pm 1$

$h_i = \pm 1$

(in general, a,b,W may be complex)

This is exactly (proportional to) the RBM representation for P(v) [with v=S]!

Application to Quantum Physics



Minimize the energy $\dfrac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}$

by adapting the weights W and biases a and b!

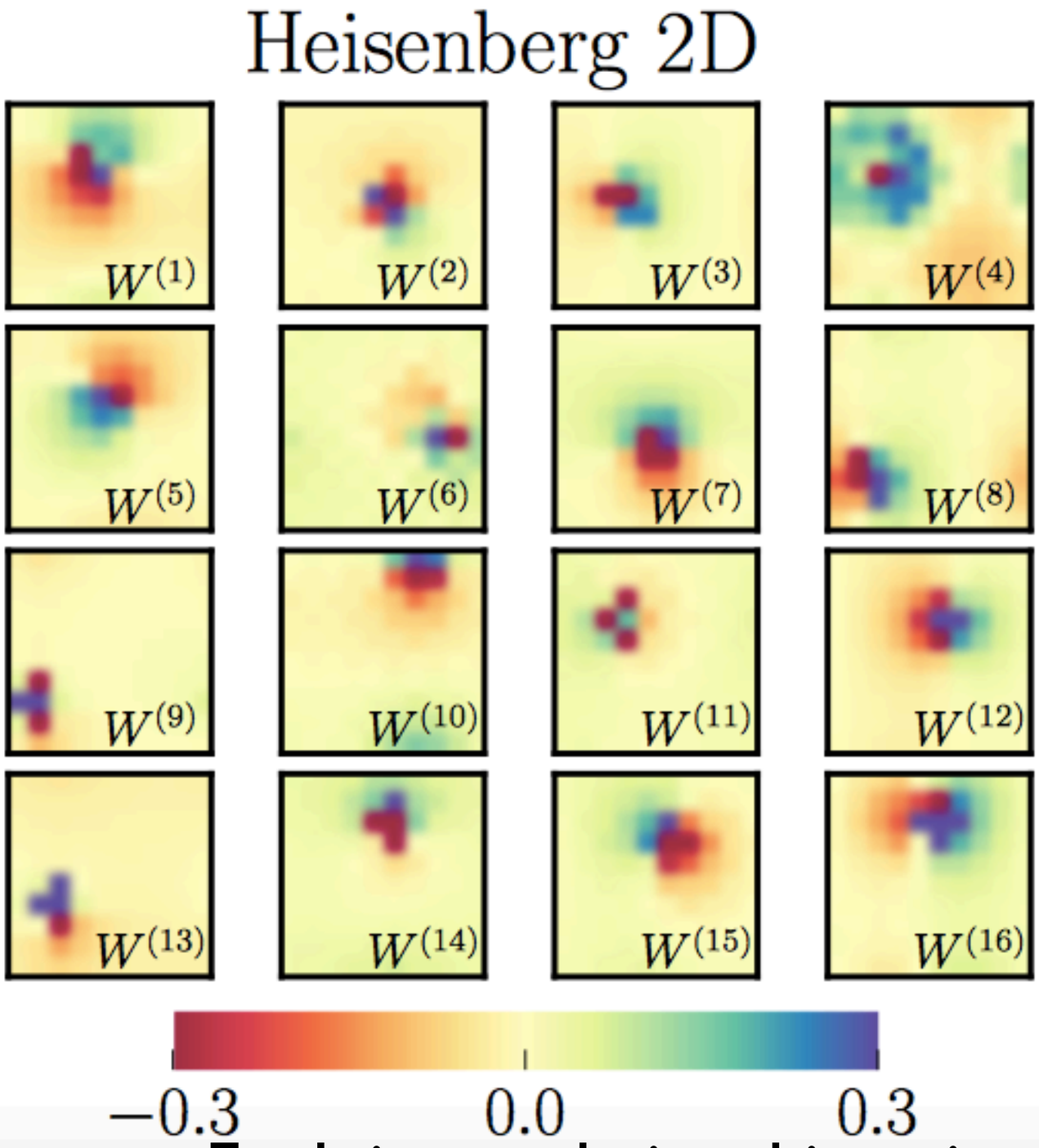
[requires additional Monte Carlo simulation, to obtain a stochastic sampling of the gradient with respect to these parameters]

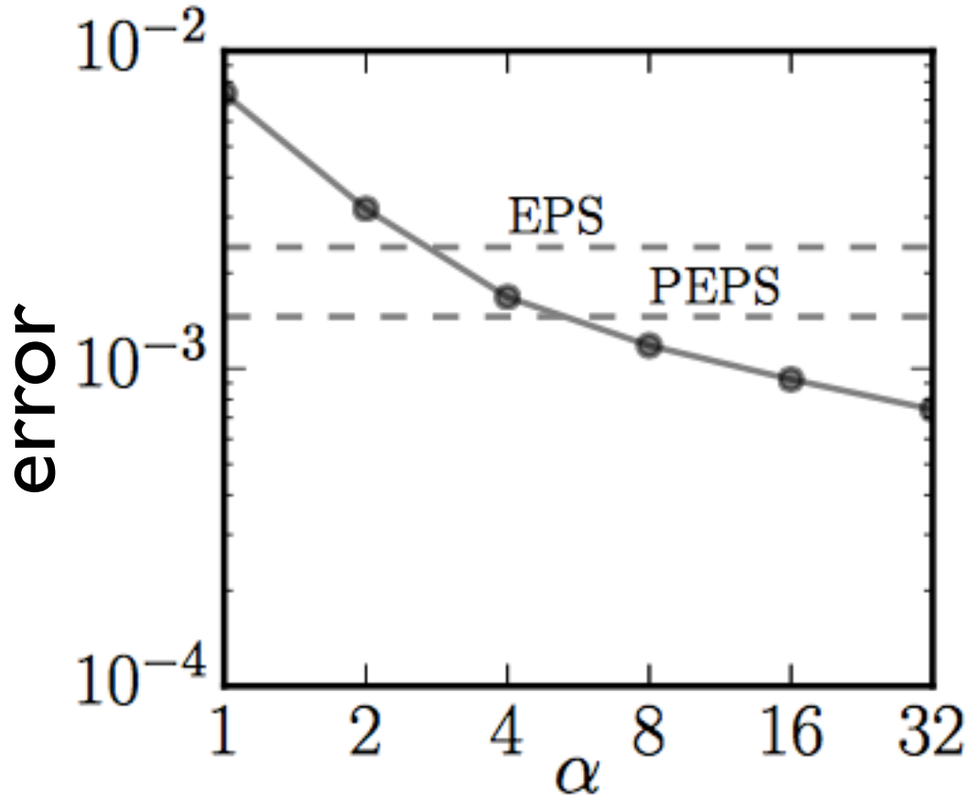
For example: sample probabilities by using Metropolis algorithm, with transition probabilities

$$P(\mathcal{S}' \leftarrow \mathcal{S}) = \min(1, \left| \frac{\Psi(\mathcal{S}')}{\Psi(\mathcal{S})} \right|^2)$$

Application to Quantum Physics

Carleo & Troyer, Science 2017





Exploit translational invariance (like in convolutional nets); weights are "filters" (convolutional kernels)