Artificial Intelligence

Neural Networks

Lesson 15: Boltzmann Machines

Vincenzo Piuri

Università degli Studi di Milano

Contents

- Boltzmann machines
- Update procedure
- Training
- Restricted Boltzmann machine
- Deep Boltzmann machines

Boltzmann Machines (1)

- Closely related to Hopfield networks
  - Differ from Hopfield networks mainly in how the neurons are updated
  - It is possible to define an energy function that assigns a numeric value (an energy) to each state of the network
- A probability distribution over the states of the network is defined based on the Boltzmann distribution

$$P(\vec{s}) = \frac{1}{c}\, e^{-\frac{E(\vec{s})}{kT}}.$$

$\vec{s}$	describes the (discrete) state of the system,
c	is a normalization constant,
E	is the function that yields the energy of a state $\vec{s}$,
T	is the thermodynamic temperature of the system,
k	is Boltzmann's constant ($k \approx 1.38 \cdot 10^{-23}\, J/K$).

Boltzmann Machines (2)

- The state $\vec{s}$ consists of the vector $\overrightarrow{act}$ of the neuron activations
- The energy function of a Boltzmann machine

$$E(\vec{\mathrm{act}}) = -\frac{1}{2}\vec{\mathrm{act}}^{\top}\mathbf{W}\vec{\mathrm{act}} + \vec{\theta}^{\top}\vec{\mathrm{act}},$$

  - W: matrix of connection weights
    $\vec{\theta}$: vector of threshold values
  - Energy change resulting from the change of a single neuron u

$$\Delta E_u = E_{\mathrm{act}_u=1} - E_{\mathrm{act}_u=0} = \sum_{v \in U-\{u\}} w_{uv}\,\mathrm{act}_v - \theta_u$$

Boltzmann Machines (3)

- The probability of a neuron being active is a logistic function of the (scaled) energy difference between its active and inactive state

- The energy difference is closely related to the network input

$$\Delta E_u = \sum_{v \in U - \{u\}} w_{uv}\, \mathrm{act}_v - \theta_u = \mathrm{net}_u - \theta_u,$$

Update Procedure (1)

- Update step:
  - a neuron u is chosen (randomly)
  - the energy difference ΔE_u and then the probability of the neuron having activation 1 are computed
  - The neuron is set to activation 1 with this probability and to 0 with the 1-complement of this probability
- This update is repeated many times for randomly chosen neurons
- Simulated annealing is carried out by slowly lowering the temperature T
  - This update process is a **Markov-Chain Monte Carlo (MCMC)** procedure

Update Procedure (2)

- After enough steps, the probability that the network is in a specific activation state depends only on the energy of that state
  - It is independent of the initial activation state the process was started with
  - This final situation is also referred to as **thermal equilibrium**

Training (1)

- The training procedure with which the probability distribution represented by a Boltzmann machine via its energy function can be adapted to a given sample of data points
  - Can only be achieved sufficiently well if the data points are a sample from a Boltzmann distribution

Training (2)

- Boltzmann Machines:
  Deviation from the structure of Hopfield networks
  - **Visible neurons**
    - Receive the data points as input
  - **Hidden neurons**
    - Not fixed by the data points
      (Hopfield networks have only visible neurons)

Training (3)

- Objective
  - Adapt the connection weights and threshold values in such a way that the true distribution underlying a given data sample is approximated well by the probability distribution represented by the Boltzmann machine on its visible neurons
- Approach
  - Choose a measure for the difference between two probability distributions
  - Gradient descent in order to minimize this difference measure

Training (4)

- **Kullback–Leibler information divergence**

$$KL(p_1, p_2) = \sum_{\omega \in \Omega} p_1(\omega) \ln \frac{p_1(\omega)}{p_2(\omega)}.$$

  - p_1 refers to the data sample
  - p_2 refers to the visible neurons of the Boltzmann machine

Training (5)

- In each training step the Boltzmann machine is ran twice (two "phases")
  - "Positive Phase":
    - Visible neurons are fixed to a randomly chosen data point
    - Only the hidden neurons are updated until thermal equilibrium is reached
  - "Negative Phase":
    - All units are updated until thermal equilibrium is reached

Training (5)

- Update rule
  - Update is performed according to the following two equations

$$\Delta w_{uv} = \frac{1}{\eta}(p^{+}_{uv} - p^{-}_{uv}) \qquad \text{and} \qquad \Delta\theta_u = -\frac{1}{\eta}(p^{+}_{u} - p^{-}_{u}).$$

p_u probability that neuron u is active,
p_{uv} probability that neurons u and v are both active simultaneously,
$+\ -$ as upper indices indicate the phase referred to.

Training (6)

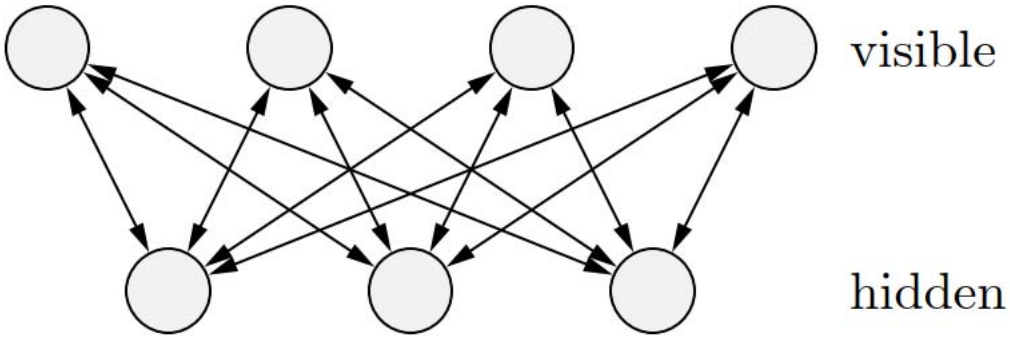
- Update rule (**Hebbian learning rule**)
  - If a neuron is more often active when a data sample is presented than when the network is allowed to run freely, the probability of the neuron being active is too low: therefore, the threshold should be reduced
  - If neurons are more often active together when a data sample is presented than when the network is allowed to run freely, the connection weight between them should be increased, so that they become more likely to be active together.

Training (7)

- This training procedure is impractical unless the networks are very small
  - The larger the network, the more update steps need to be carried out to obtain sufficiently reliable statistics for the neuron activation (pairs) needed in the update formulas

Restricted Boltzmann Machine (1)

- Efficient training is possible for the **restricted Boltzmann machine**
  - Bipartite graph instead of a fully connected graph
    - Vertices are split into two groups, the visible and the hidden neurons
    - Connections only exist between neurons from different groups



Restricted Boltzmann Machine (2)

- In a restricted Boltzmann machine, all input neurons are also output neurons and vice versa (all output neurons are also input neurons)
  - There are hidden neurons, which are different from input and output neurons
  - Each input/output neuron receives input from all hidden neurons
  - Each hidden neuron receives input from all input/output neurons

Restricted Boltzmann Machine (3)

- Lack of connections within the visible units and within the hidden units, training can proceed by repeating the following three steps
  - Step 1
    - Visible units are fixed to a randomly chosen data sample $\vec{x}$
    - Hidden units are updated once and in parallel ($\vec{y}$)
  - Step 2
    - Hidden neurons are fixed to the computed vector $\vec{y}$
    - Visible units are updated once and in parallel ($\vec{x}^*$)
    - Visible neurons are fixed to "reconstruction" $\vec{x}^*$
    - Hidden neurons are update once more ($\vec{y}^*$)
    - $\vec{x}^*\vec{y}^{*T}$ is called the **negative gradient** for the weight matrix

Restricted Boltzmann Machine (4)

- Step 3
  - Connection weights are updated with difference of positive and negative gradient

$$\Delta w_{uv} = \eta(\vec{x}_u \vec{y}_v^{\top} - \vec{x}_u^{*} \vec{y}_v^{*\top}) \qquad \text{where } \eta \text{ is a learning rate.}$$

Restricted Boltzmann Machine (5)

- Improvements of this basic procedure
  - Momentum term for the training
  - Actual probabilities instead of binary reconstructions
  - Online-like training based on small batches

Deep Boltzmann Machines

- Restricted Boltzmann machines have also been used to build deep networks similar to stacked auto-encoders for multi-layer perceptrons
  - Train a restricted Boltzmann machine
  - Create a data set of hidden neuron activations by sampling from the trained Boltzmann machine
  - Build another restricted Boltzmann machine from the obtained data set
  - Fine-tuned with back-propagation