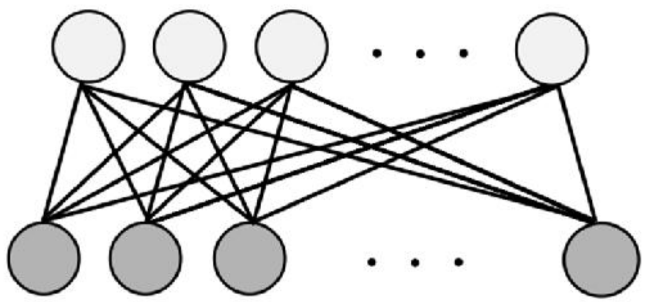
Restricted Boltzmann Machines

Alexander Tkachenko

Structure



- Two layers of binary-valued units: "visible" and "hidden".
- "Visible" and "hidden“ units form a bipartite graph.
- Parameters: weight matrix W_{ij}, bias weights a_i for the visible units and b_j for the hidden units.

Restricted Boltzmann Machines

- Used in unsupervised and supervised learning settings.
- Learn internal representation of the data.
- Originally invented in 1986, but only rose to prominence after G. Hinton and collaborators invented fast learning algorithms for them in mid-2000s.
- Have applications in dimensionality reduction, classification, collaborative filtering, feature learning, and topic modelling.
- Used in deep learning networks

Model

- The goodness of a particular configuration is quantified using energy function:

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

- Probability distribution over hidden and visible vectors is defined in terms of the energy function as

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)}, \qquad Z = \sum_{x,y} e^{-E(x,y)}$$

- Marginal probability of a visible vector *v*:

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

Training

- The goal is to maximize the product of probabilities assigned to some training set *V:*

$$argmax_W \prod_{v \in V} P(v) \quad \propto \quad argmax_W \sum_{v \in V} log\, P(v)$$

- Data log likelihood for a datapoint *v*:

$$logP(v) = log\frac{1}{Z}\sum_h e^{-E(v,h)} = log\sum_h e^{-E(v,h)} - log\sum_{x,y} e^{-E(x,y)}$$

- Optimizing maximum likelihood directly is infeasible!

Contrastive divergence algorithm

- Efficient way to train RBM.
- Performs Gibbs sampling inside a gradient descent procedure to compute weight update.

Contrastive divergence algorithm

- The basic, single-step contrastive divergence (CD-1) procedure for a single sample can be summarized as follows:
  - Take a training sample *v*, compute the probabilities of the hidden units $P(h|v)$ and sample a hidden activation vector *h* from this probability distribution.

$$P(h|v) = \prod_{j=1}^{n} P(h_j|v), \qquad P(h_j = 1|v) = sigmoid(b_j + \sum_{i=1}^{m} v_i w_{ij})$$

  - Compute the *positive gradient*: $p_{ij} = v_i h_j$
  - From *h*, sample a reconstruction *v'* of the visible units according to $P(v|h)$, then resample the hidden activations *h'* from this. (Gibbs sampling step).
  - Compute the *negative gradient*: $n_{ij} = v'_i h'_j$
  - Let the weight update to be the positive gradient minus the negative gradient, times some learning rate: $\Delta w_{ij} = \mu(p_{ij} - n_{ij})$
  - The update biases *a*, *b* analogously.