Neural Networks

12. Restricted Boltzmann Machines

Center for Cognitive Science
Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

Thursday 9$^{\text{th}}$ May, 2024

Preface

Bias integration

- until now:
  - $\mathbf{x} \rightarrow \mathbf{x}'$, bias is $(n+1)$-th weight vector

$$\mathbf{y} = f(\mathbf{W}\mathbf{x}') \qquad \mathbf{W} \in \mathbb{R}^{m \times (n+1)}$$

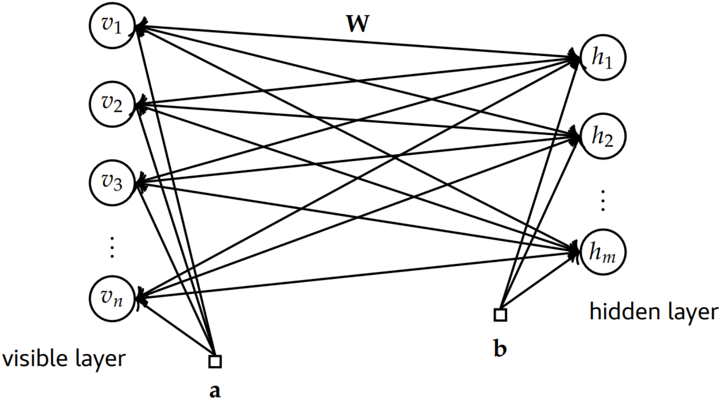
- today:
  - $\mathbf{x}$ stays as it is, bias $\mathbf{b}$ is separated from weight matrix $\mathbf{W}$
  - $\mathbf{b}$ is a vector - one weight per output neuron

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \qquad \mathbf{W} \in \mathbb{R}^{m \times n} \quad \mathbf{b} \in \mathbb{R}^{m}$$

Restricted Boltzmann Machines

- recurrent generative model
- binary states: $v_j,\ h_i \in \{0, 1\}$
  - v_j, h_i are "visible" and "hidden" neurons
- **probabilistic NN**:
  - activations are *not* computed directly, e.g. $\mathbf{y} = f(\mathbf{net})$
  - probabilistic activation:
    $$\mathbf{p} = f(\mathbf{net}) \quad \rightarrow \quad P[y_i = 1] = p_i$$
- RBM represents (approximates) complex probability distribution

Transition Functions



- **probabilistic computation**
  - "forward": $P[h_i = 1] = p_i^{hid}$ $\quad \mathbf{p}^{hid} = f(\mathbf{W}\mathbf{v} + \mathbf{b})$
  - "backward": $P[v_j = 1] = p_j^{vis}$ $\quad \mathbf{p}^{vis} = f(\mathbf{W}^\mathsf{T}\mathbf{h} + \mathbf{a})$

RBM Operation Modes

- **Positive phase**
  - visible layer is fixed to input
    $\mathbf{v}^{+} = \mathbf{x}$
  - activation on the hidden layer is generated
    $\mathbf{v}^{+} \rightarrow \mathbf{h}^{+}$
  - no need to repeat as $\mathbf{v}^{+}$ is fixed
- **Negative phase**
  - starting with hidden activation, the activation on visible layer is generated (*)
    $\mathbf{h} \rightarrow \mathbf{v}^{-}$
  - activation on the hidden layer is generated
    $\mathbf{v}^{-} \rightarrow \mathbf{h}^{-}$
  - repeat (if desired)

Generation of $\mathbf{v}^-$

Simple: backward pass $\mathbf{h} \rightarrow \mathbf{v}^-$

Gibbs sampling:

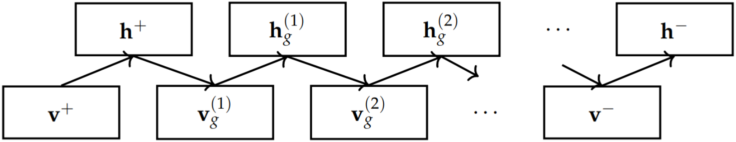
- proper sampling from complex distribution (distribution is represented by our network)
- simplified version for RBMs:
  - repeat n times:
    $\mathbf{v} :=$ backward($\mathbf{h}$)
    $\mathbf{h} :=$ forward($\mathbf{v}$)
    $\mathbf{v}^- :=$ backward($\mathbf{h}$)

Training – Contrastive Divergence

- traditional SGD scheme (epochs, training inputs, . . .)

- gradients computed using Contrastive Divergence:
  - positive phase to obtain $\mathbf{v}^+$, $\mathbf{h}^+$
  - negative phase to obtain $\mathbf{v}^-$, $\mathbf{h}^-$ using Gibbs sampling from $\mathbf{h}^+$
  - compute deltas

$$\Delta \mathbf{W} = \mathbf{h}^+ {\mathbf{v}^+}^T - \mathbf{h}^- {\mathbf{v}^-}^T$$
$$\Delta \mathbf{a} = \mathbf{v}^+ - \mathbf{v}^-$$
$$\Delta \mathbf{b} = \mathbf{h}^+ - \mathbf{h}^-$$



Task

- Train RBM to store MNIST digits (similar to auto-encoder or Hopfield) and generate new digits from random input
  - implement binary sampling from a distribution
  - initialize weights
  - compute forward & backward pass
  - implement gibbs sampling
  - update the weights iteratively to train the network