Generative Adversarial Networks (GANs) & Restricted Boltzmann Machines (RBMs)

Lecture 24

Changho Suh

October 7, 2021

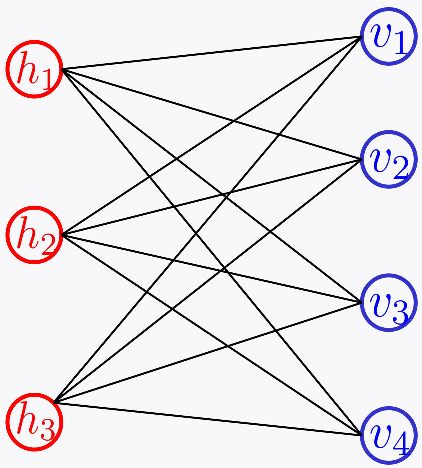
Outline

1. Investigate a training method for RBM.

2. Introduce a loss function, and discuss its theoretical background.

3. Study how to compute the loss function.

How to train RBM $\theta := (W, b, c)$



Given visible units with *m* examples:

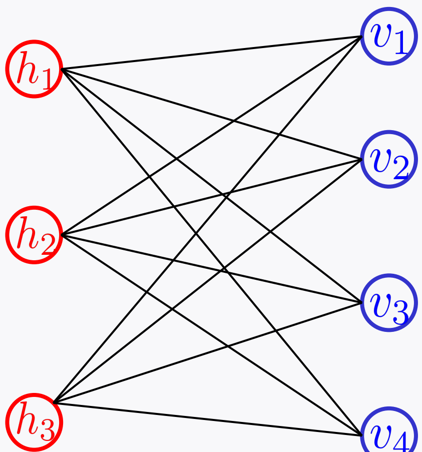
$$\{v^{(i)}\}_{i=1}^{m}$$

Iterative algorithm:

$v^{(t),(i)}$: estimate of visible units w.r.t. *i*-th example at the *t*-th iteration

$h^{(t),(i)}$: estimate of hidden units

Training procedure $\theta := (W, b, c)$



Given visible units with *m* examples:

$$\{v^{(i)}\}_{i=1}^{m}$$

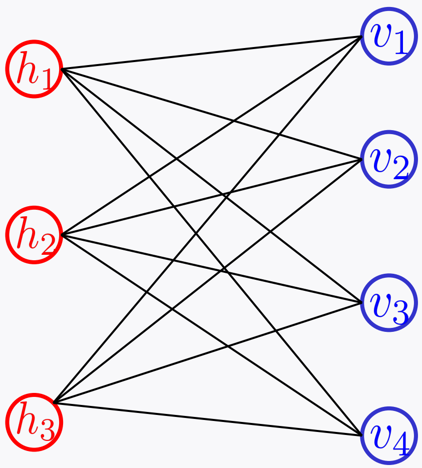
Step 1: Sample $h^{(t),(i)} \sim \mathbb{P}(h|v^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

$$\mathbb{P}(h|v^{(t),(i)}) = \frac{e^{c^{(t)T}h + h^T W^{(t)} v^{(t),(i)}}}{\sum_h e^{c^{(t)T}h + h^T W^{(t)} v^{(t),(i)}}}$$

Step 2: Sample $v^{(t),(i)} \sim \mathbb{P}(v|h^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

$$\mathbb{P}(v|h^{(t),(i)}) = \frac{e^{b^{(t)T}v + v^T W^{(t)T} h^{(t),(i)}}}{\sum_v e^{b^{(t)T}v + v^T W^{(t)T} h^{(t),(i)}}}$$

Training procedure $\theta := (W, b, c)$



Given visible units with *m* examples:

$$\{v^{(i)}\}_{i=1}^{m}$$

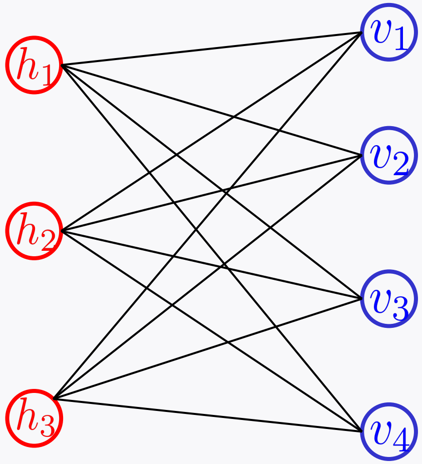
Step 1: Sample $h^{(t),(i)} \sim \mathbb{P}(h|v^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

Step 2: Sample $v^{(t),(i)} \sim \mathbb{P}(v|h^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

Step 3: Compute a cost function:

$$J^{(t)}(\theta) := \frac{1}{m} \sum_{i=1}^{m} \ell\left(v^{(i)}, v^{(t),(i)}\right)$$

Training procedure $\theta := (W, b, c)$



Given visible units with *m* examples:

$$\{v^{(i)}\}_{i=1}^{m}$$

Step 1: Sample $h^{(t),(i)} \sim \mathbb{P}(h|v^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

Step 2: Sample $v^{(t),(i)} \sim \mathbb{P}(v|h^{(t),(i)}) \quad \forall i \in \{1, \ldots, m\}$

Step 3: Compute a cost function: $J^{(t)}(\theta)$

Step 4: Update parameters via gradient descent:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha^{(t)} \nabla J^{(t)}(\theta)$$

Loss function $\ell\left(v^{(i)}, v^{(t),(i)}\right)$?

Turns out: The following loss is optimal in a certain sense:

$$\ell_{\text{opt}}\left(v, \hat{v}\right) = \underset{\uparrow \atop \text{free energy}}{F(v)} - F(\hat{v})$$

$$\text{where } F(v) = -\log\left(\sum_h e^{-E(v,h)}\right)$$

How to compute the loss function?

$$\ell_{\text{opt}}(v, \hat{v}) = F(v) - F(\hat{v})$$

where $F(v) = -\log\left(\sum_h e^{-E(v,h)}\right)$

Recall: $E(v,h) := -b^T v - c^T h - h^T W v$

$$F(v) = -\log\left(\sum_h e^{b^T v + c^T h + h^T W v}\right) = -\log\left(e^{b^T v} \sum_h e^{c^T h + h^T W v}\right)$$

$$= -b^T v - \log\left(\sum_h e^{c^T h + h^T W v}\right)$$

So far ...

Have learned about **DNNs, CNNs, RNNs, RFs, dimensionality reduction, clustering, autoencoder, matrix completion, GANs**, and **RBMs.**

RFs: Small-data technique!

There are more advanced small-data techniques, part of which rely upon autoencoder and matrix completion:

semi-supervised learning
transfer learning
simulator-based learning
:

Look ahead

Will explore three small-data techniques:

1. semi-supervised learning
2. transfer learning
3. simulator-based learning