Contrastive Divergence (CD)

From “Restricted Boltzmann Machine – Short Tutorial”

Rossen Radev
iMonad Software

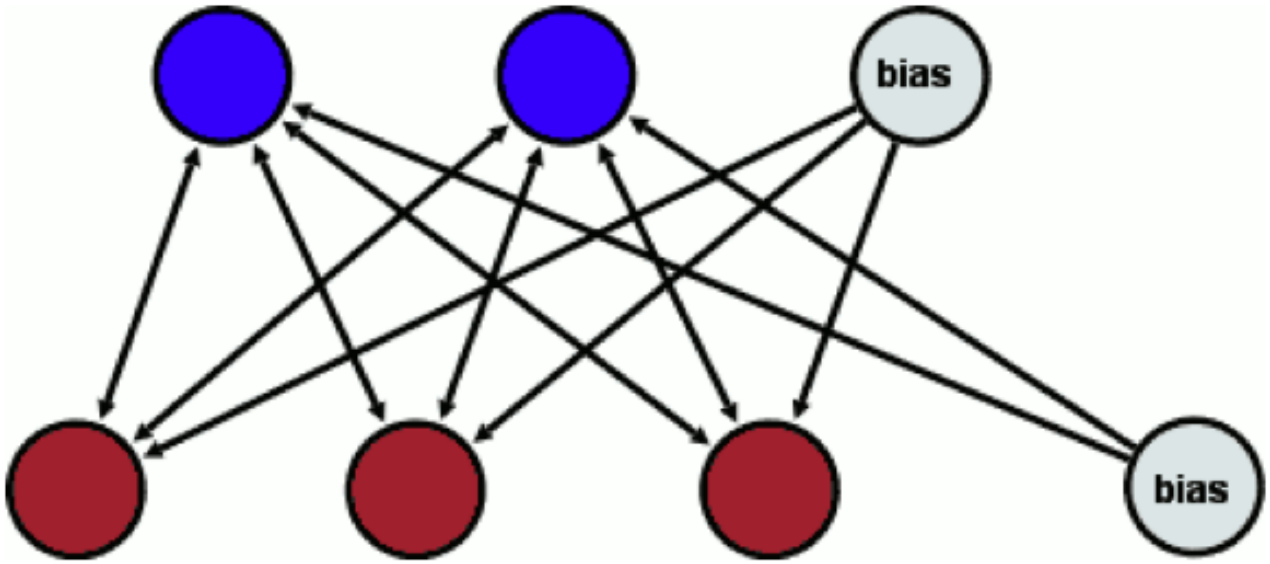
http://imonad.com/category/rbm/

which also gives Python code for an example

RBM Model

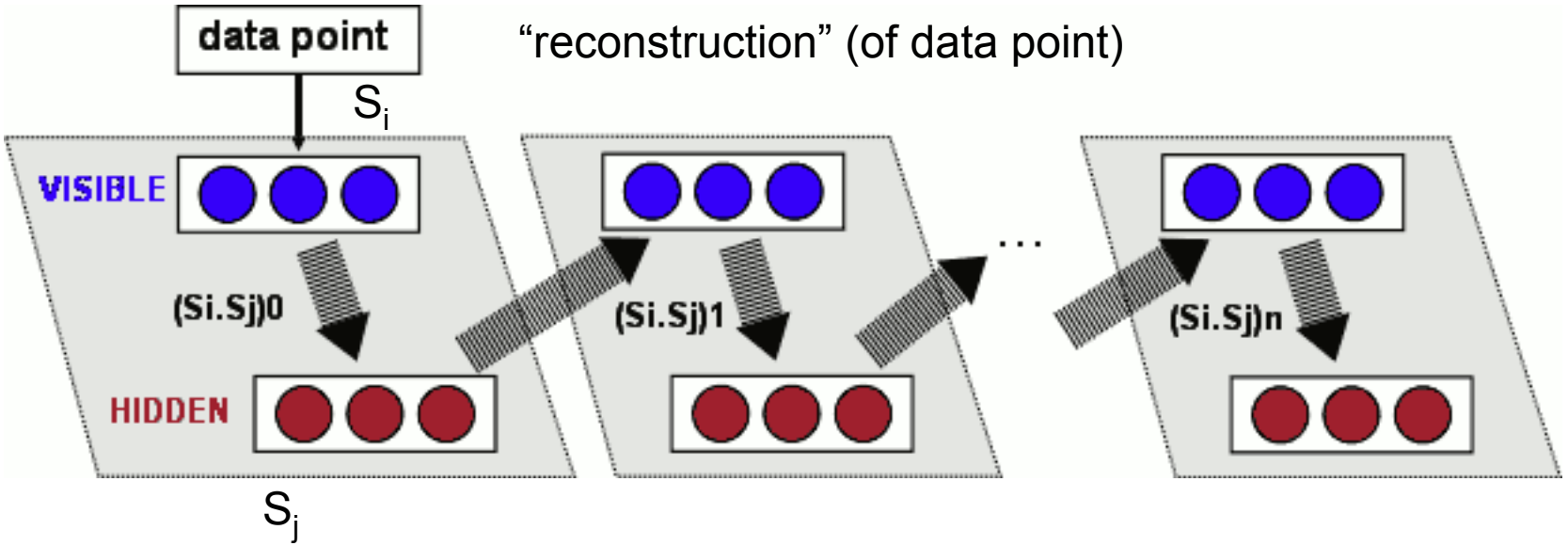
Visible

Hidden



"R" means no connections *within* layers

CD Training Step



S_i is the output of neuron i

$(S_i S_j)n$ is the mean value of the product $S_i S_j$ (neurons in opposite layers)

after the n^{th} iteration

(as the network is run through a few cycles probabilistically)

One Reconstruction Step

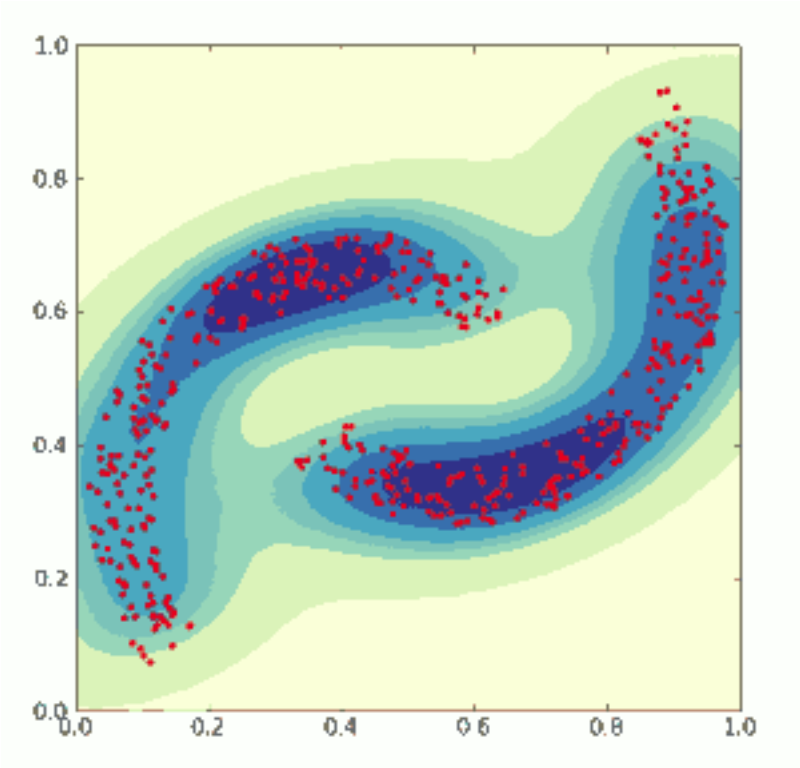
1. Get one data point from data set.
2. Use values of this data point to set state of visible neurons Si
3. Compute Sj for each hidden neuron based on formula above and states of visible neurons Si
4. Compute (Si.Sj)0 – here (…) means just values, not average
5. On visible neurons, compute Si using the Sj computed in step 3. This is known as “reconstruction”
6. Compute state of hidden neurons Sj again using Si from step 5.
7. Now use Si and Sj to compute (Si.Sj)1.
8. Repeating multiple times steps 5,6 and 7 compute (Si.Sj)n. Where n is small number and can increase with learning steps to achieve better accuracy.

CD Algorithm

For each epoch do:

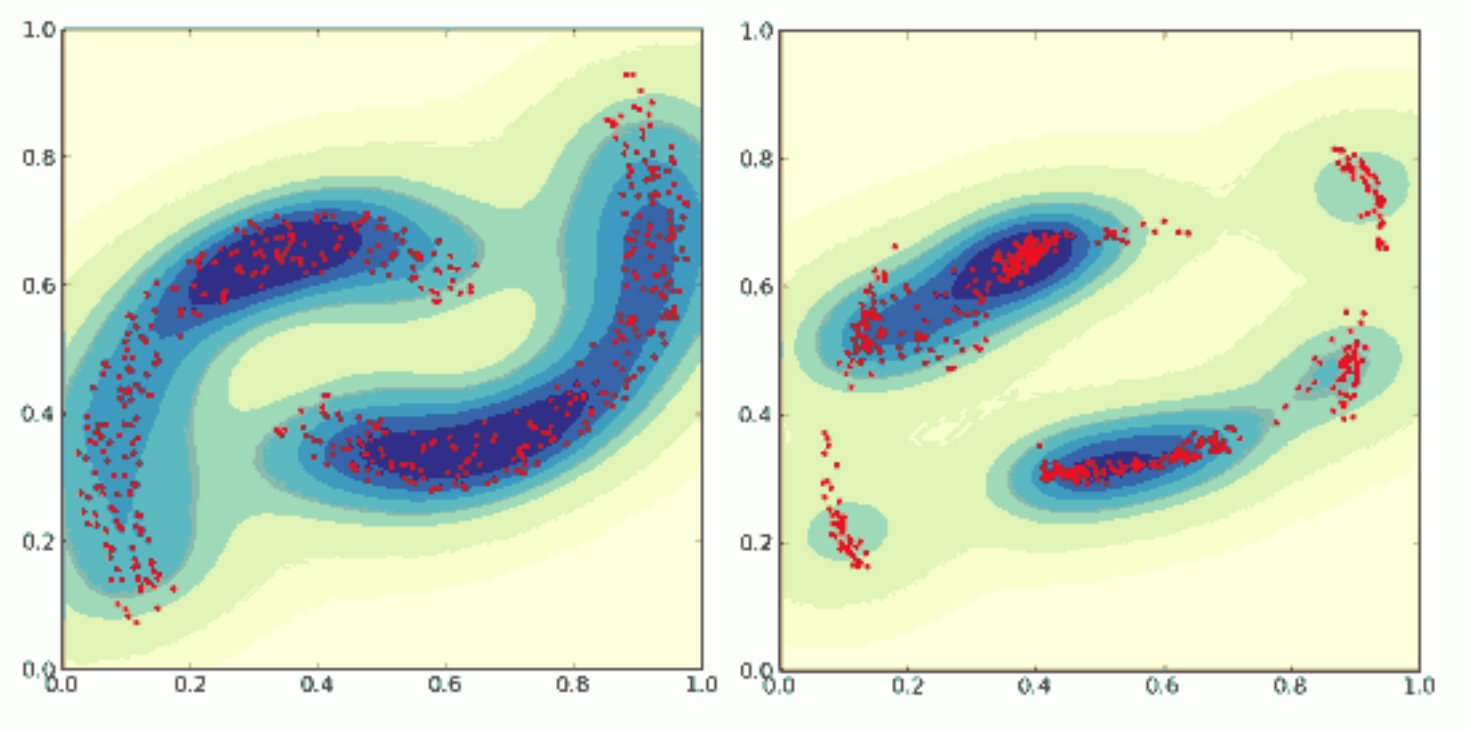
- For each data point in the data set do:
  - CDpos =0, CDneg=0 (matrices)
  - Perform steps 1…8 from the previous slide.
  - Accumulate CDpos = CDpos + (Si.Sj)0
  - Accumulate CDneg = CDneg + (Si.Sj)n
  - Compute average of CDpos and CDneg by dividing them by number of data points.
  - Compute CD = (Si.Sj)0 – (Si.Sj)n = CDpos – CDneg
  - Update weights and biases W' = W + η*CD
    (biases are just weights to neurons that stay always 1.0)
  - Compute some "error function" like sum of squared difference between Si in Step 1 and Si in Step 5 i.e. between data and reconstruction.
  - repeat for the next epoch until error is small or some fixed number of epochs.

Example Training Data

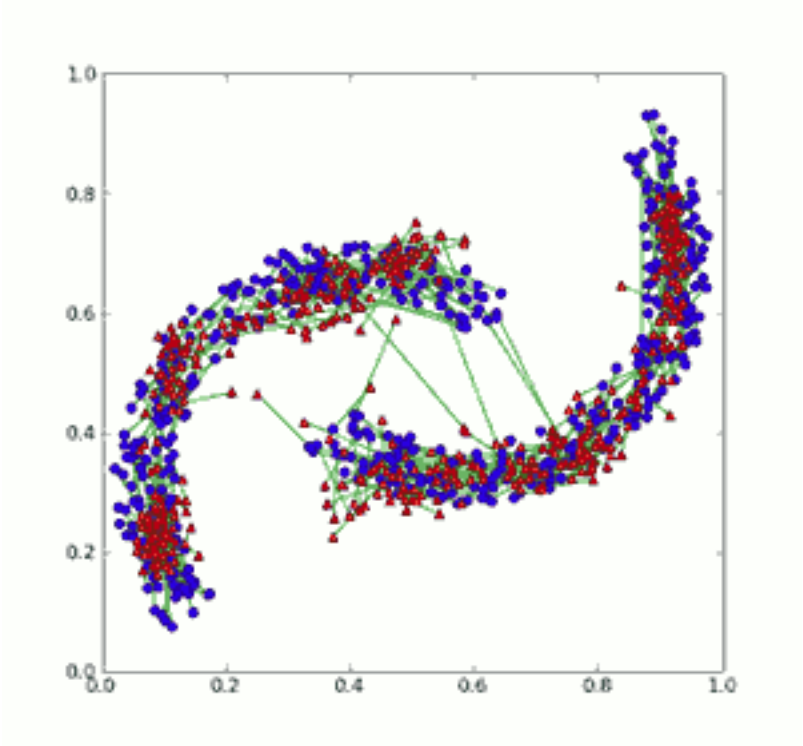


500 data points, synthetic

Training and Reconstruction



Training (blue), Reconstruction (red)



Details Provided

- Neuron function F = lo + (hi – lo)/(1.0 + exp(-A*Sj)),
  with Sj = F(Sum(Si x Wij + N(0,sigma)))
  N meaning normal dist with mean 0, stdev sigma (Why add this?)
  A is adjusted during learning process??
- RBM architecture is 2 visible neurons for 2D data points and 8 hidden neurons. On some experiments with simple patterns 4 neurons are enough to reconstruct pattern successfully.
- Main difficulty observed is to fine tune the set of learning parameters.
- Used the following parameters:
  - sigma=0.2
  - A=0.1 (on visible neurons)
  - Learning Rate W = 0.5
  - Learning Rate A = 0.5
  - Learning Cost = 0.00001
  - Learning Moment = 0.9