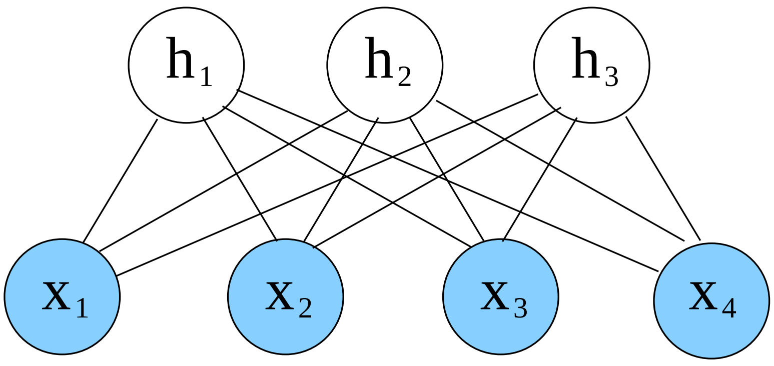
University of Cambridge
Engineering Part IIB

Module 4F10: Statistical Pattern Processing

Handout 6: Restricted Boltzmann Machines



Mark Gales
`mjfg@eng.cam.ac.uk`
Michaelmas 2015

Introduction

So far we have looked at generative models where the parametric form of the model is specified then the parameters trained. It would be useful to look at more general classes of generative model that are more universal. For discriminative classifiers, neural network are one form of universal approximator, what about generative models?

One form that has been examined for a range of tasks is the restricted Boltzmann machine (RBM). This is a particular form of energy based model (EBM) that is simple to train. In EBMs

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp\left(-\mathcal{G}(\boldsymbol{x}|\boldsymbol{\theta})\right)$$

where

- $\mathcal{G}(\boldsymbol{x}|\boldsymbol{\theta})$ is an energy function that yields a score for the observation $\boldsymbol{x}$
- Z is the normalisation term that ensures that there is a valid PDF

The aim is to adjust the parameters of the energy function, $\boldsymbol{\theta}$, so the amount of energy is minimised - this increase the likelihood.

Hidden Variables

To increase the power of the EBM it is possible to add latent variables in the same way as multiple components were added for the Gaussian mixture model. The joint distribution of (discrete) hidden $\boldsymbol{h}$ and (continuous) observed $\boldsymbol{x}$ variables is

$$p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}) = \frac{1}{Z}\exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)$$

the normalisation term, or partition function, is

$$Z = \int \sum_{h} \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right) d\boldsymbol{x}$$

The marginal distribution for the observed data is then

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z}\sum_{h}\exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)$$

This can be written in terms of the free energy

$$\mathcal{F}(\boldsymbol{x}|\boldsymbol{\theta}) = -\log\left(\sum_{h}\exp(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}))\right)$$

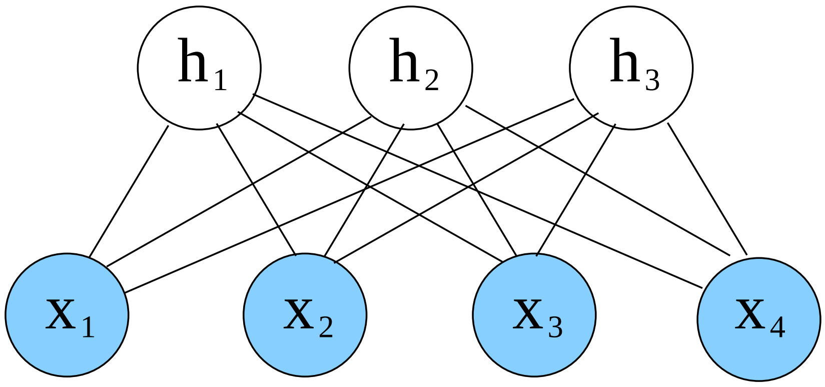
which allows the original (non latent variable form) to be derived

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z}\exp\left(-\mathcal{F}(\boldsymbol{x}|\boldsymbol{\theta})\right)$$

Though the form is the same the model parameters $\boldsymbol{\theta}$ have changed to reflect the fact that latent variables have been introduced. The complexity of optimising this function depends on the nature of the free energy function.

Restricted Boltzmann Machines

One useful constraint to introduce is that the connections in the energy function are only between the observations and hidden layers, there are no connections between elements of the same layer.



Given this form a restriction, the resulting model is a restricted Boltzmann machine.

So far there has been no discussion of the nature of the energy function or the nature of the parameters. One form of parametrisation (related to the form of neural networks described later) is to have the following parameters

- observed layer bias: d-dimensional vector, $\boldsymbol{a}$
- hidden layer bias: J-dimensional vector, $\boldsymbol{b}$
- connection: $d \times J$-dimensional matrix, $\boldsymbol{W}$

Some standard forms of function are described in the next slide.

Energy Functions

Two forms of energy function are:

- Discrete Observations: The simplest (original) form is to have both observed and hidden variables being discrete. Here

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta}) = -\sum_{i=1}^{d} a_i x_i - \sum_{j=1}^{J} b_j h_j - \sum_{i,j} x_i h_j w_{ij}$$

  In order to train the parameters it will be necessary to define the two conditions distributions relating the observed, $\boldsymbol{x}$, and hidden, $\boldsymbol{h}$, variables:

$$\begin{aligned} P(h_j = 1|\boldsymbol{x}, \boldsymbol{\theta}) &= \frac{1}{1 + \exp(-b_j - \Sigma_i x_i w_{ij})} \\ P(x_i = 1|\boldsymbol{h}, \boldsymbol{\theta}) &= \frac{1}{1 + \exp(-a_i - \Sigma_j h_j w_{ij})} \end{aligned}$$

- Continuous Observations: when the observations are continuous (hidden still discrete) the following form can be used

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta}) = \sum_{i=1}^{d} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j=1}^{J} b_j h_j - \sum_{i,j} \frac{x_i}{\sigma_i} h_j w_{ij}$$

  Again the two conditions distributions relating the observed, $\boldsymbol{x}$ and hidden $\boldsymbol{h}$ variables need to be found:

$$\begin{aligned} P(h_j = 1|\boldsymbol{x}, \boldsymbol{\theta}) &= \frac{1}{1 + \exp\left(-b_j - \Sigma_i \frac{x_i}{\sigma_i} w_{ij}\right)} \\ p(x_i|\boldsymbol{h}, \boldsymbol{\theta}) &= \mathcal{N}\left(x_i; a_i + \sigma_i \sum_j h_j w_{ij}, \sigma_i^2\right) \end{aligned}$$

Parameter Estimation

As with the other models the parameters of the system

$$\boldsymbol{\theta} = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{W}\}$$

need to be estimated. Again maximum likelihood estimation can be used. The training data are a set of N samples

$$\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$$

The criterion to optimise is

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log(p(\boldsymbol{x}_i|\boldsymbol{\theta}))$$

For RBMs gradient decent based algorithms are used. Here the update rule has the form

$$\boldsymbol{\theta}[\tau+1] = \boldsymbol{\theta}[\tau] + \eta\, \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})|_{\theta[\tau]}$$

All that is required is to get the gradient of the log-likelihood. This can be written as

$$\boldsymbol{\nabla}\log(p(\boldsymbol{x}|\boldsymbol{\theta}) = \boldsymbol{\nabla}\log\left(\sum_{h}\exp\left(-\mathcal{G}(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta})\right)\right) - \boldsymbol{\nabla}\log(Z)$$

The problem is that deriving an analytic expression for the normalisation term Z may not possible (as with PoEs).

Normalisation Term Gradient

This gradient can be expressed as

$$\begin{aligned}
\boldsymbol{\nabla} \log(Z) &= \frac{1}{Z}\boldsymbol{\nabla}\left(\int \sum_{\boldsymbol{h}} \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right) d\boldsymbol{x}\right) \\
&= \int \sum_{\boldsymbol{h}} \frac{1}{Z}\boldsymbol{\nabla}\left(\exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)\right) d\boldsymbol{x} \\
&= -\sum_{\boldsymbol{h}} \int \left(\frac{1}{Z}\exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right) d\boldsymbol{x} \\
&= -\sum_{\boldsymbol{h}} \int \boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}) p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}) d\boldsymbol{x}
\end{aligned}$$

This has allowed the gradient of the normalisation term to be expressed as the expected value of the gradient of the energy function with respect to $p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})$. However it is still not possible to find an expression for Z.

Rather than solving integrals it is possible to approximate the integral by drawing a finite number of samples for the distribution (Monte-Carlo approach)

$$\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \approx \frac{1}{K}\sum_{k=1}^{K} f(\boldsymbol{x}^{(k)})$$

where $\boldsymbol{x}^{(k)}$ is a sample generated from $p(\boldsymbol{x})$.

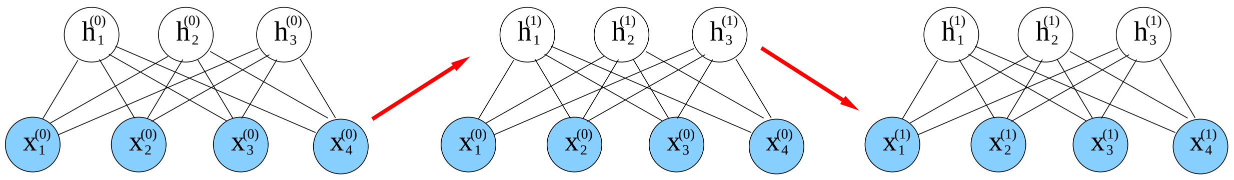
Fortunately drawing a sample from $p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})$ is not a function of Z - no need to find Z!

RBM Gibbs Sampling

Generating observations from general probability density functions is a whole research area. In this lecture we will just use (not discuss in detail) Gibbs sampling from $p(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta})$

For many situations it is not possible to generate a sample $\boldsymbol{x}^{(k)}$ from a distribution directly (as is the case here). Gibbs sampling says that a sample from a distribution can be generated by fixing some elements, generate samples conditional on the fixed elements, and repeat for all elements.

For RBMs as there are no connections within a layer it is simple to split the data between the hidden units and observed units.



$h^{(0)}_1$ $h^{(0)}_2$ $h^{(0)}_3$
$x^{(0)}_1$ $x^{(0)}_2$ $x^{(0)}_3$ $x^{(0)}_4$
$h^{(1)}_1$ $h^{(1)}_2$ $h^{(1)}_3$
$x^{(0)}_1$ $x^{(0)}_2$ $x^{(0)}_3$ $x^{(0)}_4$
$h^{(1)}_1$ $h^{(1)}_2$ $h^{(1)}_3$
$x^{(1)}_1$ $x^{(1)}_2$ $x^{(1)}_3$ $x^{(1)}_4$

1. set $k = 0$, initialise observed vector, $\boldsymbol{x}^{(0)}$
2. draw a sample of the hidden vector $\boldsymbol{h}^{(k+1)}$ from
$$P(\boldsymbol{h}|\boldsymbol{x}^{(k)}, \boldsymbol{\theta})$$
3. draw a sample of the observed vector $\boldsymbol{x}^{(k+1)}$ from
$$p(\boldsymbol{x}|\boldsymbol{h}^{(k+1)}, \boldsymbol{\theta})$$
4. $k = k + 1$; goto (2)

This generates a final sample: $\{\boldsymbol{x}^{(\infty)}, \boldsymbol{h}^{(\infty)}\}$

RBM Gibbs Sampling (cont)

For both the binary and continuous observations the conditional distributions have been defined. For the continuous case

$$
\begin{aligned}
P(h_j = 1|\boldsymbol{x}, \boldsymbol{\theta}) &= \frac{1}{1 + \exp\left(-b_j - \Sigma_i \frac{x_i}{\sigma_i} w_{ij}\right)} \\
p(x_i|\boldsymbol{h}, \boldsymbol{\theta}) &= \mathcal{N}\left(x_i; a_i + \sigma_i \sum_j h_j w_{ij}, \sigma_i^2\right)
\end{aligned}
$$

have been defined. Thus generating data from this RBM requires sampling from a discrete distribution (based on logistic regression) or Gaussian distribution and iterating. All these are simple operations.
There are a few issues that need to be addressed to make Gibbs' sampling work:

- number of iterations: a finite number of samples are run in practice. This needs to be large enough so that the samples that are actually used are independent of each other;
- sensitivity to initial value: if the initial observation vector is not "reasonable" it may take many iterations to reach values that are samples from the required distribution. This is the burn-in period;
- number of samples: there have to be sufficient samples that the estimate of the integral is "good".

Gradient Estimate (details)

Need to find the gradient of the log-likelihood with respect to the model parameters $\boldsymbol{\theta}$. From slide 5

$$\boldsymbol{\nabla}\log(p(\boldsymbol{x}|\boldsymbol{\theta})) = \boldsymbol{\nabla}\log\left(\sum_h \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)\right) - \boldsymbol{\nabla}\log(Z)$$

Differentiating the first term yields

$$\begin{aligned}\boldsymbol{\nabla}\log\left(\sum_h \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)\right) &= \frac{-\sum_h \boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)}{\sum_h \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\right)} \\ &= -\sum_h P(\boldsymbol{h}|\boldsymbol{x})\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\end{aligned}$$

The derivative of the training data for this first term is sometimes expressed in terms of approximating samples drawn from the true distribution of the observed data $p_{\text{data}}(\boldsymbol{x})$, $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$ the training data!)

$$\begin{aligned}&\int\sum_h P(\boldsymbol{h}|\boldsymbol{x})\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})p_{\text{data}}(\boldsymbol{x})d\boldsymbol{x} \\ &\qquad\approx \frac{1}{N}\sum_{i=1}^N\sum_h P(\boldsymbol{h}|\boldsymbol{x}_i)\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x}_i,\boldsymbol{h}|\boldsymbol{\theta})\end{aligned}$$

From slide 6 the normalisation can be approximated as

$$\begin{aligned}\boldsymbol{\nabla}\log(Z) &= -\sum_h\int\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})p(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})d\boldsymbol{x} \\ &\approx -\frac{1}{K}\sum_{k=1}^K\boldsymbol{\nabla}\mathcal{G}(\hat{\boldsymbol{x}}^{(k)},\hat{\boldsymbol{h}}^{(k)}|\boldsymbol{\theta})\end{aligned}$$

Combining these together yields

$$\begin{aligned}\frac{1}{N}\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}) \approx& \frac{1}{K}\sum_{k=1}^K\boldsymbol{\nabla}\mathcal{G}(\hat{\boldsymbol{x}}^{(k)},\hat{\boldsymbol{h}}^{(k)}|\boldsymbol{\theta}) \\ &-\frac{1}{N}\sum_{i=1}^N\sum_h P(\boldsymbol{h}|\boldsymbol{x}_i)\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x}_i,\boldsymbol{h}|\boldsymbol{\theta})\end{aligned}$$

Gradient Estimate

As both the derivative of the numerator and denominator can be expressed in terms of drawing samples from distributions, the complete log-likelihood derivative is often written as

$$\frac{1}{N}\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}) = \langle\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\rangle_{\text{model}} - \langle\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})\rangle_{\text{data}}$$

where

- $\langle . \rangle_{\text{data}}$ indicates the expected value based on the data
- $\langle . \rangle_{\text{model}}$ indicates the expected value based on the model

Using the Gibbs sampling estimates it is now possible to define gradient of the log-likelihood function

$$\begin{aligned}\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}) &= N\int\sum_{h} p(\boldsymbol{x},\boldsymbol{h})\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta})d\boldsymbol{x} - \sum_{i=1}^{N}\sum_{h} P(\boldsymbol{h}|\boldsymbol{x}_i)\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x}_i,\boldsymbol{h}|\boldsymbol{\theta})\\ &\approx \frac{N}{K}\sum_{k=1}^{K}\boldsymbol{\nabla}\mathcal{G}(\hat{\boldsymbol{x}}^{(k)},\hat{\boldsymbol{h}}^{(k)}|\boldsymbol{\theta}) - \sum_{i=1}^{N}\sum_{h} P(\boldsymbol{h}|\boldsymbol{x}_i)\boldsymbol{\nabla}\mathcal{G}(\boldsymbol{x}_i,\boldsymbol{h}|\boldsymbol{\theta})\end{aligned}$$

Here the samples $\{\hat{\boldsymbol{x}}^{(k)},\hat{\boldsymbol{h}}^{(k)}\}$ are the ones extracted from Gibbs sampling that are expected to be independent.

Thus the gradient can be defined without explicitly having to generate the normalisation term (though the samples could be used to find this of course).

Contrastive Divergence

Though the previous approach can be used to find the gradient it is still too slow for many applications, due to both the burn-in period and number of Gibbs sampling iterations to obtain independent samples.
To address this some additional approximations can be used.

- set the initial value of the observations to one of the training samples $\boldsymbol{x}^{(0)} = \boldsymbol{x}_i$
- limit the number of iterations of Gibbs sampling before generating the sample $\hat{\boldsymbol{x}}^{(k)}, \hat{\boldsymbol{h}}^{(k)}$. Though this means that the samples are still going to be correlated with the data point it allows the computational load of the sampling process to be controlled. Systems can be trained using a single iteration of Gibbs sampling.

Both of these approximations means that the gradient used is not the "correct" gradient. However in practice these approximations allow large data sets to be used to train the RBM.

Contrastive divergence optimisation is a general approach for optimisation systems where it is not possible, or impractical, to compute the normalisation term Z. The example of general PoEs is one scenario where this can also be applied.

RBM Classifiers

RBMs are generative models, they yield $p(\boldsymbol{x}|\boldsymbol{\theta})$. In theory they could be used as part of a generative classifier - for example for a K class classifier

$$P(\omega_i|\boldsymbol{x},\boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta}_i)P(\omega_i)}{\Sigma_{k=1}^{K} p(\boldsymbol{x}|\boldsymbol{\theta}_k)P(\omega_k)}$$

where $p(\boldsymbol{x}|\boldsymbol{\theta}_k)$ is an RBM trained on the data from class ω_k.

BUT the training of the RBM was designed so that it was not necessary to find the normalisation term. As this will vary from class to class, ie Z_k for class ω_k, it will not simply cancel between numerator and denominator.

Various approximations have been examined to achieve this. For example the unnormalised RBM probabilities can be used

$$\tilde{p}(\boldsymbol{x}|\boldsymbol{\theta}_k) = \sum_{h} \exp\left(-\mathcal{G}(\boldsymbol{x},\boldsymbol{h}|\boldsymbol{\theta}_k)\right)$$

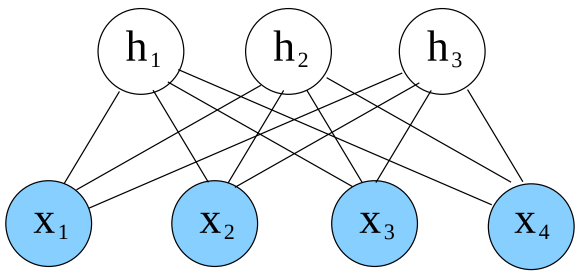
This can be fed into a softmax function of the form

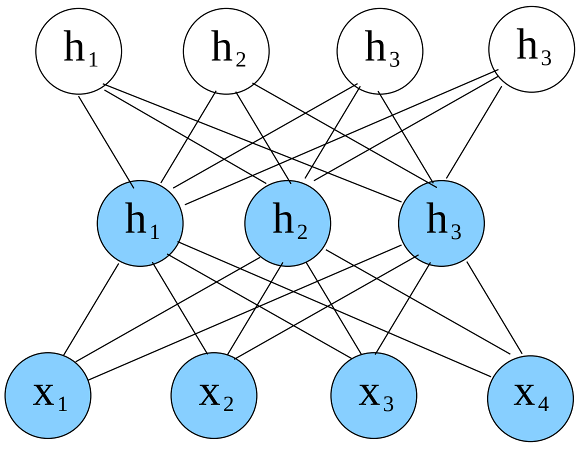
$$P(\omega_i|\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{\alpha}) = \frac{\exp(\log(\tilde{p}(\boldsymbol{x}|\boldsymbol{\theta}_i)) + \alpha_i)}{\Sigma_{k=1}^{K} \exp(\log(\tilde{p}(\boldsymbol{x}|\boldsymbol{\theta}_k)) + \alpha_k)}$$

Effectively normalisation terms (and priors) are trained in a separate softmax optimisation stage.

Multiple Layer RBMs

It is possible to stack multiple layers of RBMs together, provided only connections between layers are used. After training one layer, the inferred hidden units $\boldsymbol{h}$ can be fixed and used as training examples for another level.



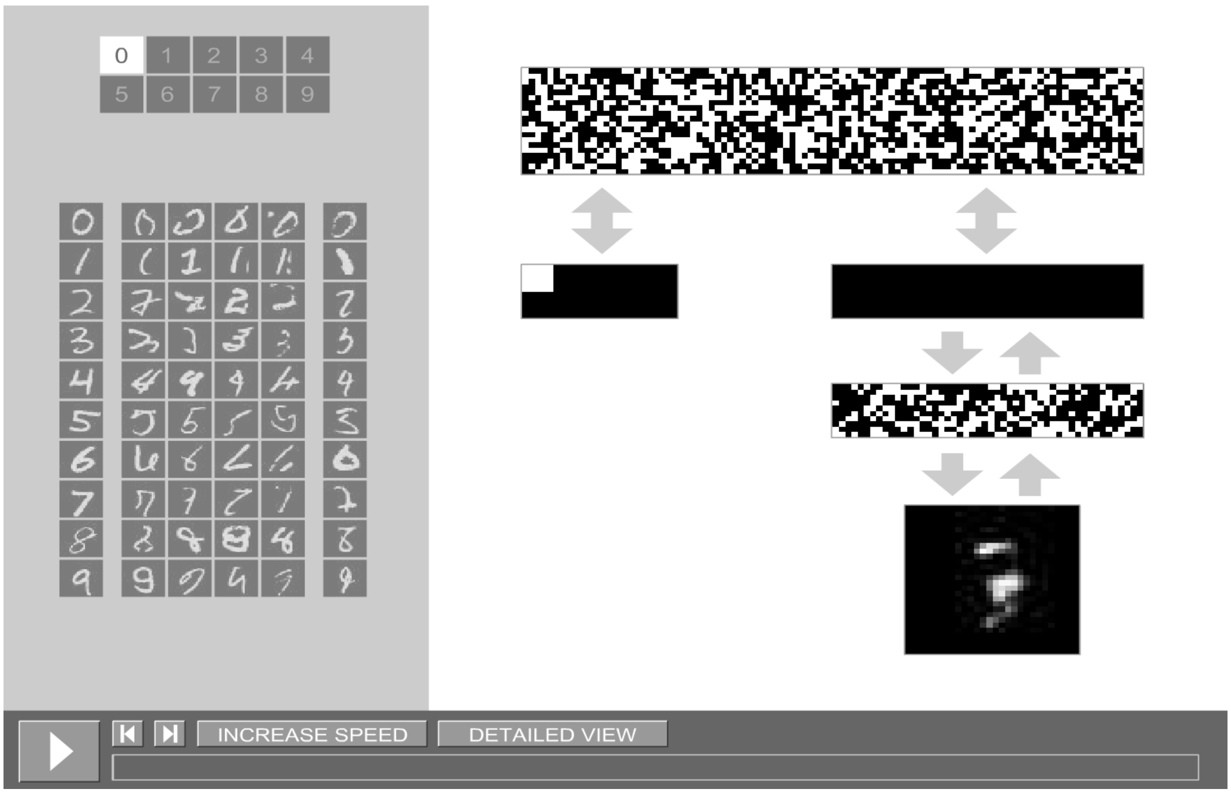


This process allows a stacked version of the RBM to be trained. This will be discussed later in the course for the initialisation of multi-layer perceptrons.

UPS Digits Example

For an example of RBMs generating digits see:

`http://www.cs.toronto.edu/%7Ehinton/digits.html`


0
1
2
3
4
5
6
7
8
9
INCREASE SPEED
DETAILED VIEW