CSC2535 2013
Lecture 8a

Learning Multiplicative Interactions

Geoffrey Hinton

Two different meanings of "multiplicative"

- If we take two density models and multiply together their probability distributions at each point in data-space, we get a "product of experts".
  - The product of two Gaussian experts is a Gaussian.
- If we take two variables and we multiply them together to provide input to a third variable we get a "multiplicative interaction".
  - The distribution of the product of two Gaussian-distributed variables is NOT Gaussian distributed. It is a heavy-tailed distribution. One Gaussian determines the standard deviation of the other Gaussian.
  - Heavy-tailed distributions are the signatures of multiplicative interactions between latent variables.

The heavy-tailed world

- The prediction errors for financial time-series are typically heavy-tailed. This is mainly because the variance is much higher in times of uncertainty.
- The prediction errors made by a linear dynamical systems are usually heavy-tailed on real data.
  - Occasional very weird things happen. This violates the conditions of the central limit theorem.
- The outputs of linear filters applied to images are heavy-tailed.
  - Gabor filters nearly always output almost exactly zero. But occasionally they have large outputs.

Learning multiplicative interactions

- It is fairly easy to learn multiplicative interactions if all of the variables are observed.
  - This is possible if we control the variables used to create a training set (e.g. pose, lighting, identity …)
- It is also easy to learn energy-based models in which all but one of the terms in each multiplicative interaction are observed.
  - Inference is still easy.
- If more than one of the terms in each multiplicative interaction are unobserved, the interactions between hidden variables make inference difficult.
  - Alternating Gibbs can be used if the latent variables form a bi-partite graph.

Higher order Boltzmann machines (Sejnowski, ~1986)

- The usual energy function is quadratic in the states:

$$-E = bias\ terms\ + \sum_{i<j} s_i s_j\ w_{ij}$$

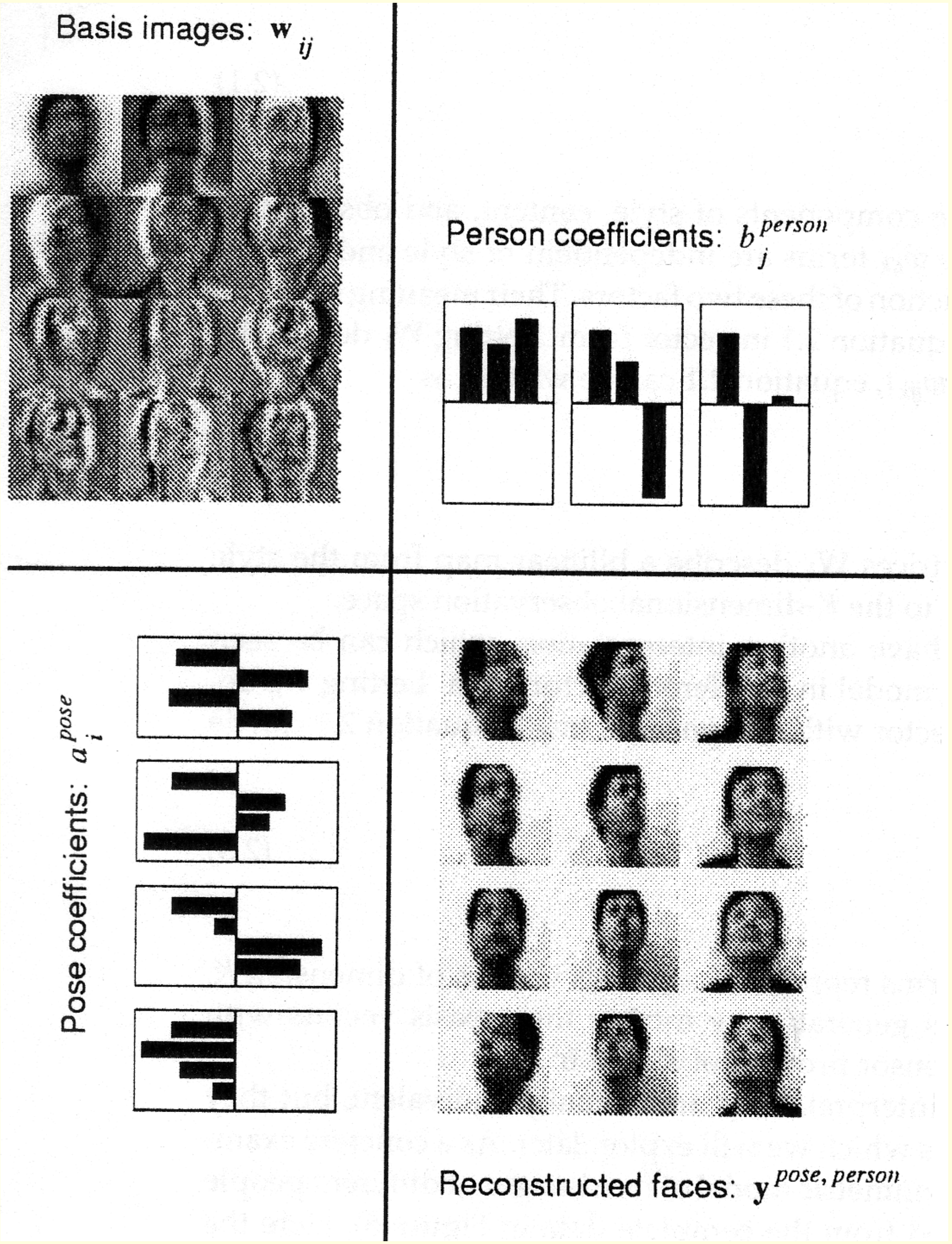
- But we could use higher order interactions:

$$-E = bias\ terms\ + \sum_{i,j,h} s_i s_j s_h\ w_{ijh}$$

- Hidden unit h acts as a switch. When h is on, it switches in the pairwise interaction between unit i and unit j.
  - Units i and j can also be viewed as switches that control the pairwise interactions between j and h or between i and h.

Learning how style and content interact

- Tenenbaum and Freeman (2000) describe a model in which a “style” vector and a “content” vector interact multiplicatively to determine a datavector (e.g. and image).
- The outer-product of the style and content vectors determines a set of coefficients for basis functions.
  - This is not at all like the way a user vector and a movie vector interact to determine a rating. The rating is the inner-product.

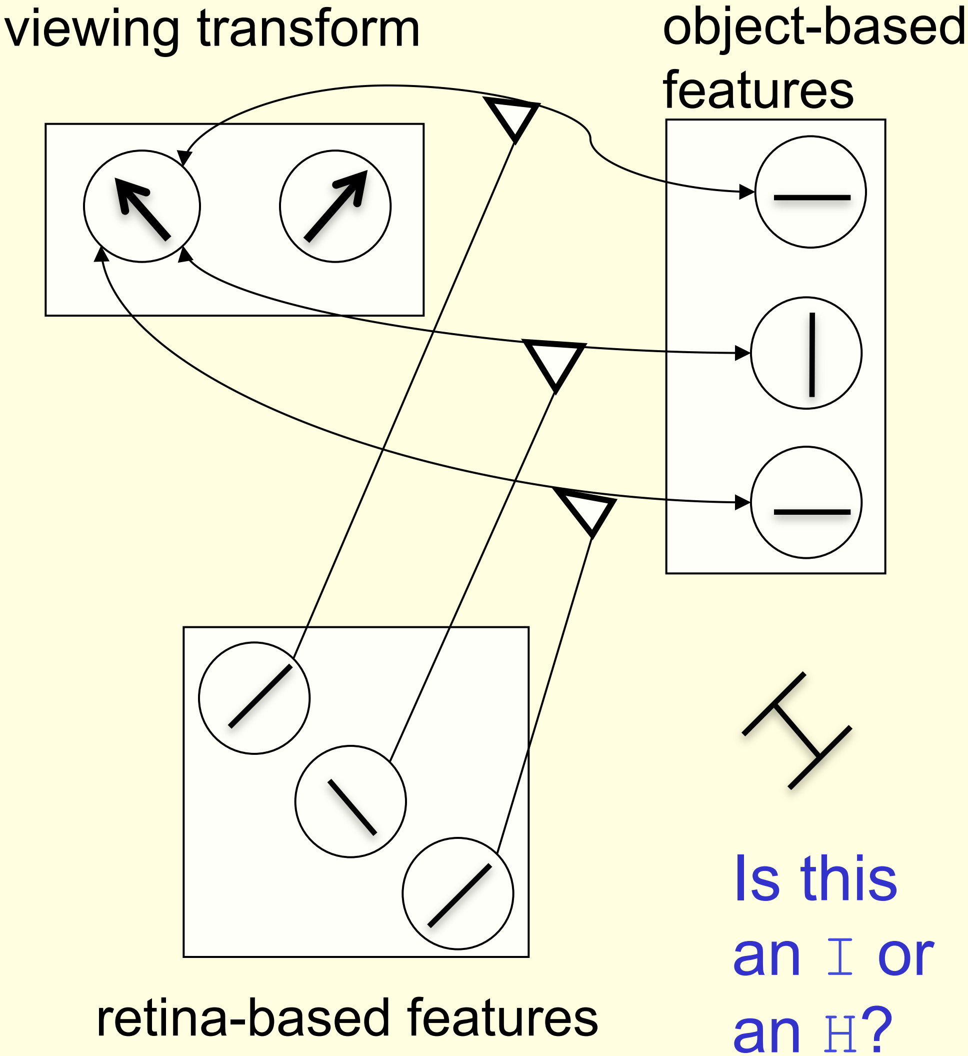

Basis images: $\mathbf{w}_{ij}$
Person coefficients: b_j^{person}
Pose coefficients: a_i^{pose}
Reconstructed faces: $\mathbf{y}^{pose, person}$

It is an unfortunate coincidence that the number of components in each pose vector is equal to the number of different pose vectors.

The model is only really interesting if we have less components per style or content vector than style or content vectors

A higher-order Boltzmann machine with one visible group and two hidden groups

- We can view it as a Boltzmann machine in which the inputs create interactions between the other variables.
  - This type of model is now called a conditional random field.
  - Inference can be hard in this model.
  - Inference is much easier with two visible groups and one hidden group

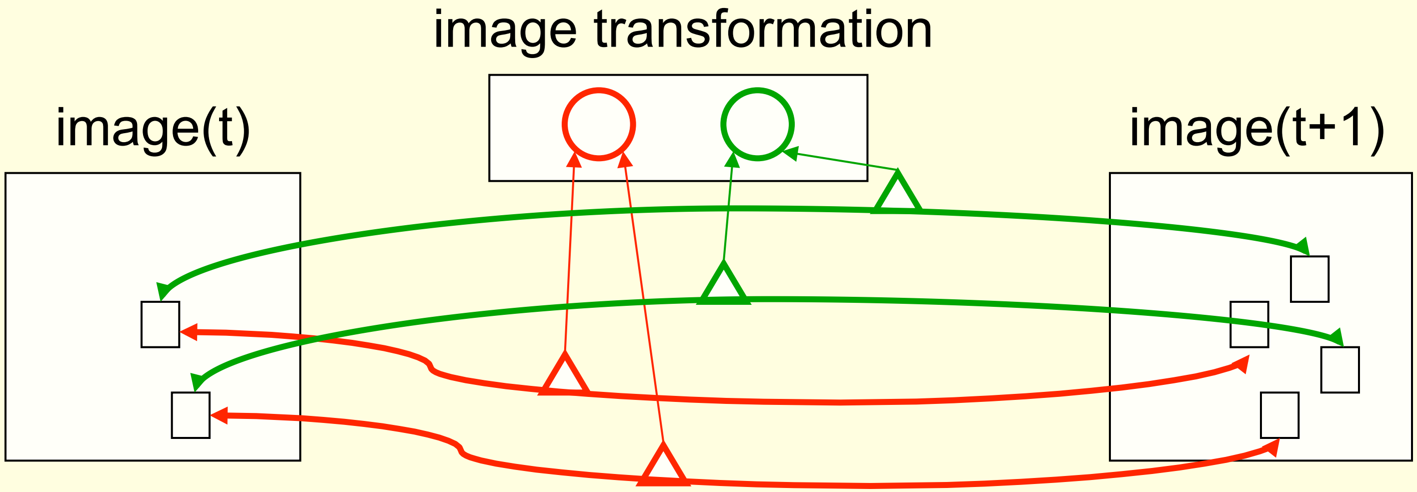


Is this an I or an H?

Using higher-order Boltzmann machines to model image transformations

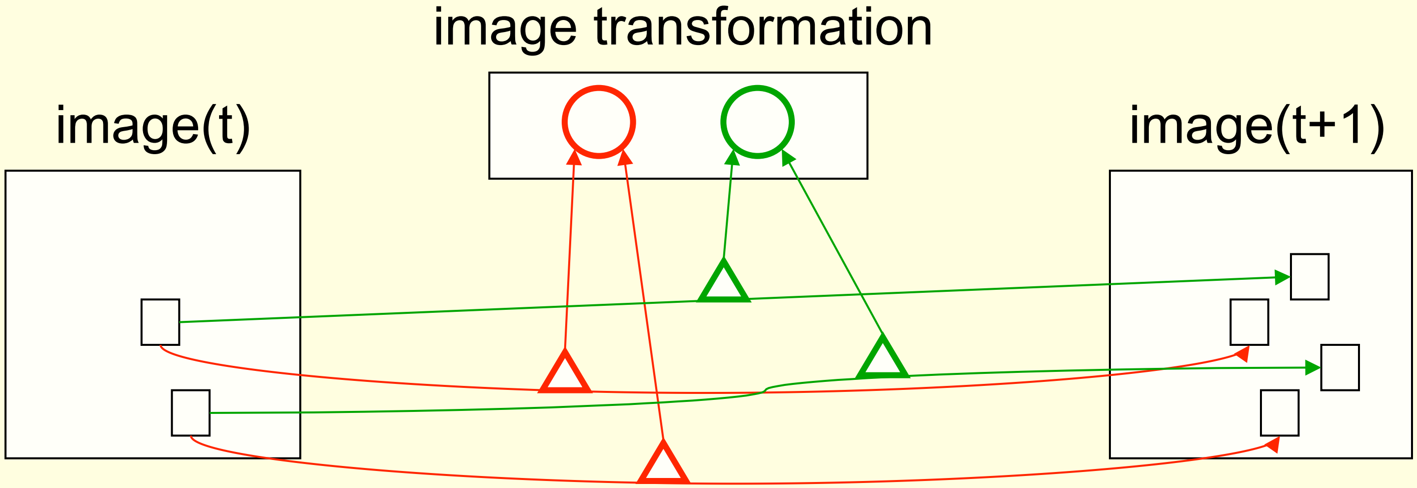
(Memisevic and Hinton, 2007)

- A global transformation specifies which pixel goes to which other pixel.
- Conversely, each pair of similar intensity pixels, one in each image, votes for a particular global transformation.



Making the reconstruction easier

- Condition on the first image so that only one visible group needs to be reconstructed.
  - Given the hidden states and the previous image, the pixels in the second image are conditionally independent.



The main problem with 3-way interactions

- There are far too many of them.
- We can reduce the number in several straight-forward ways:
  - Do dimensionality reduction on each group before the three way interactions.
  - Use spatial locality to limit the range of the three-way interactions.
- A much more interesting approach (which can be combined with the other two) is to factor the interactions so that they can be specified with fewer parameters.
  - This leads to a novel type of learning module.

Factoring three-way interactions

- If three-way interactions are being used to model a nice regular multi-linear structure, we may not need cubically many degrees of freedom.
  - For modelling effects like viewpoint and illumination many fewer degrees of freedom may be sufficient.
- There are many ways to factor 3-D interaction tensors.
- We use factors that correspond to 3-way outer-products.
  - Each factor only has 3N parameters.
  - By using about N/3 factors we get quadratically many parameters which is the same as a simple weight matrix.

Factoring the three-way interactions

unfactored

factored

$$-E = \sum_{i,j,h} s_i s_j s_h \; w_{ijh} \qquad\qquad -E = \sum_{f} \sum_{i,j,h} s_i s_j s_h \; w_{if} w_{jf} w_{hf}$$

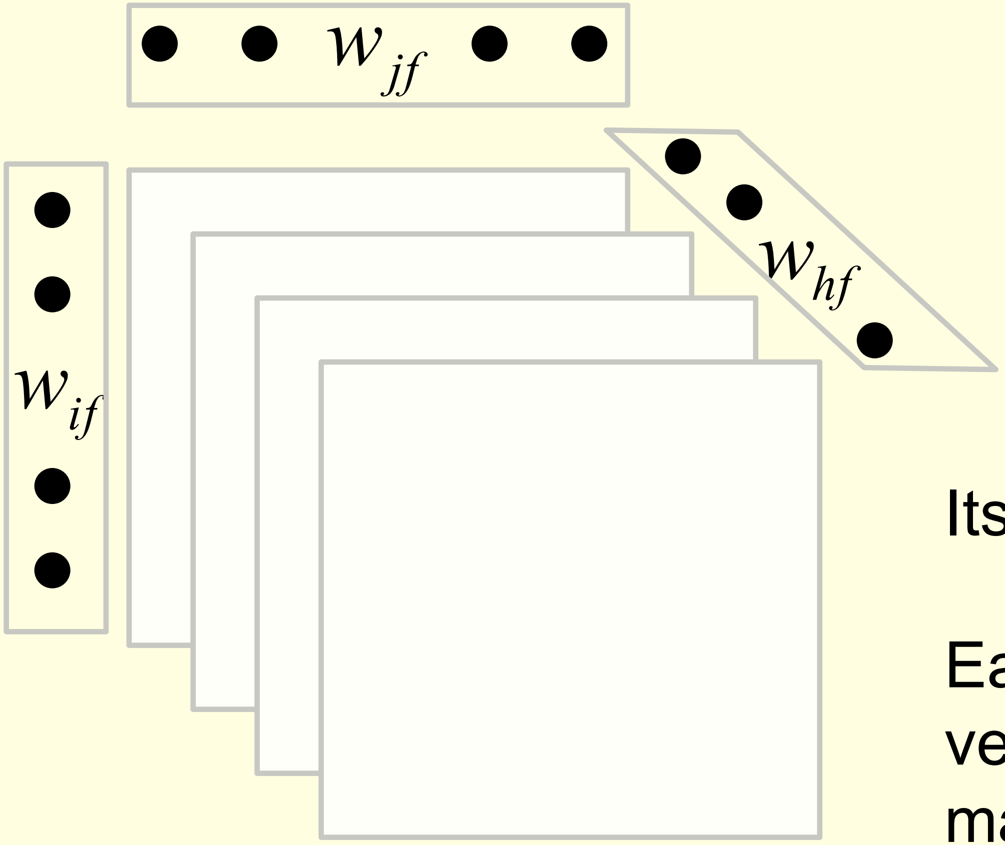
$$\left[E_f(s_h = 0) \; - E_f(s_h = 1)\right] \; = \; w_{hf} \sum_{i} s_i w_{if} \sum_{j} s_j w_{jf}$$

$$\left[E_f(s_j = 0) \; - E_f(s_j = 1)\right] \; = \; w_{jf} \sum_{i} s_i w_{if} \sum_{h} s_h w_{hf}$$

How changing the binary state of unit j changes the energy contributed by factor f.

What unit j needs to know in order to do Gibbs sampling

A picture of the rank 1 tensor contributed by factor f



Its a 3-way outer product.

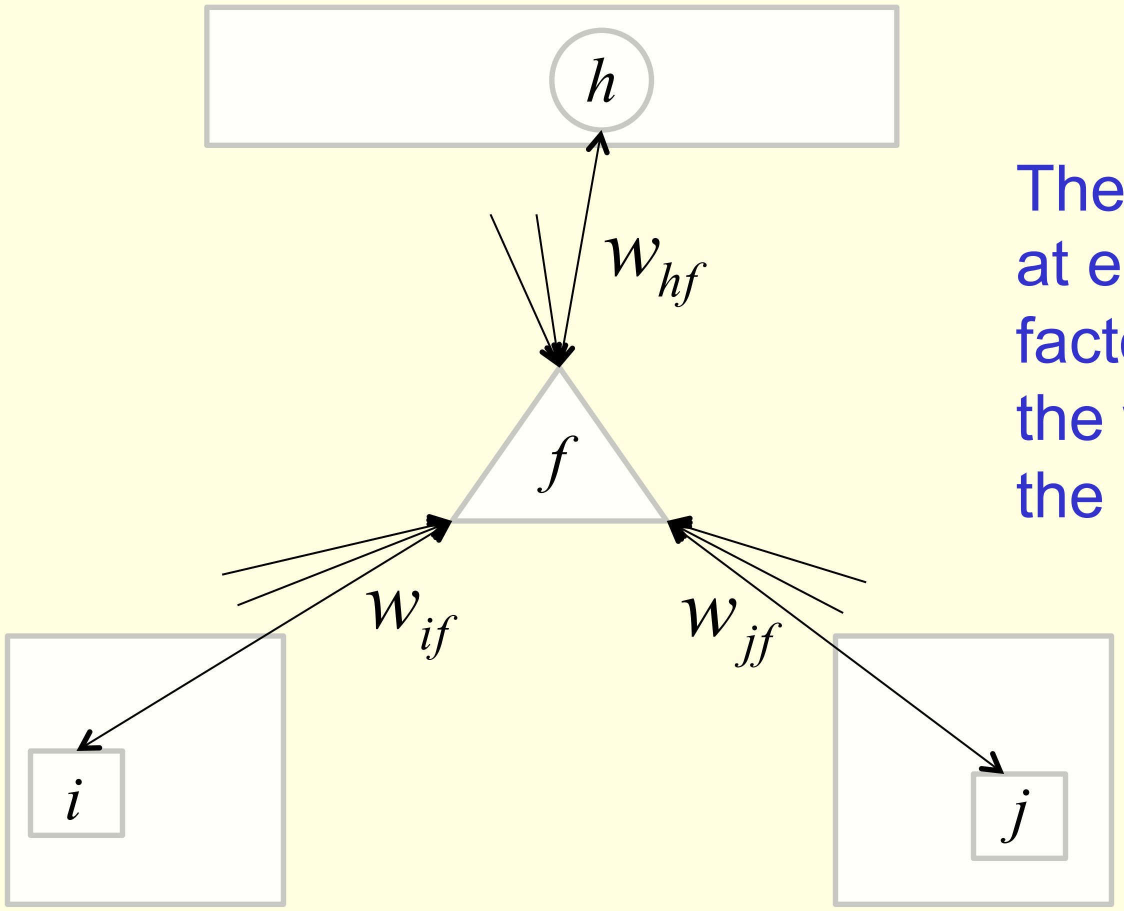
Each layer is a scaled version of the same rank 1 matrix.

The dynamics

- The visible and hidden units get weighted input from the factors and use this input in the usual stochastic way.
  - They have stochastic binary states (or a mean-field approximation to stochastic binary states).

- The factors are deterministic and implement a type of belief propagation. They do not have “states”.
  - Each factor computes three separate sums by adding up the input it gets from each separate group of units.
  - Then it sends the product of the summed inputs from two groups to the third group.

Belief propagation

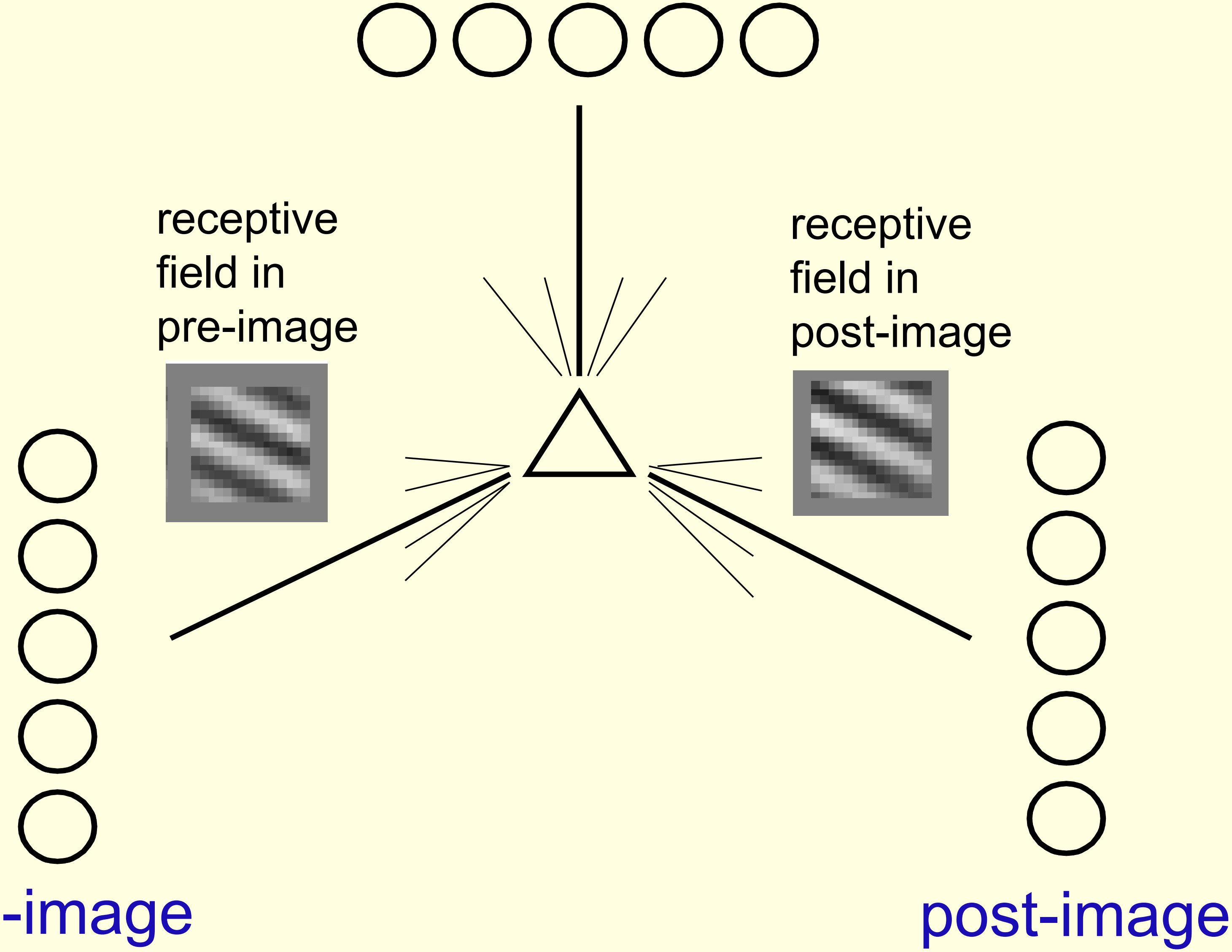


The outgoing message at each vertex of the factor is the product of the weighted sums at the other two vertices.

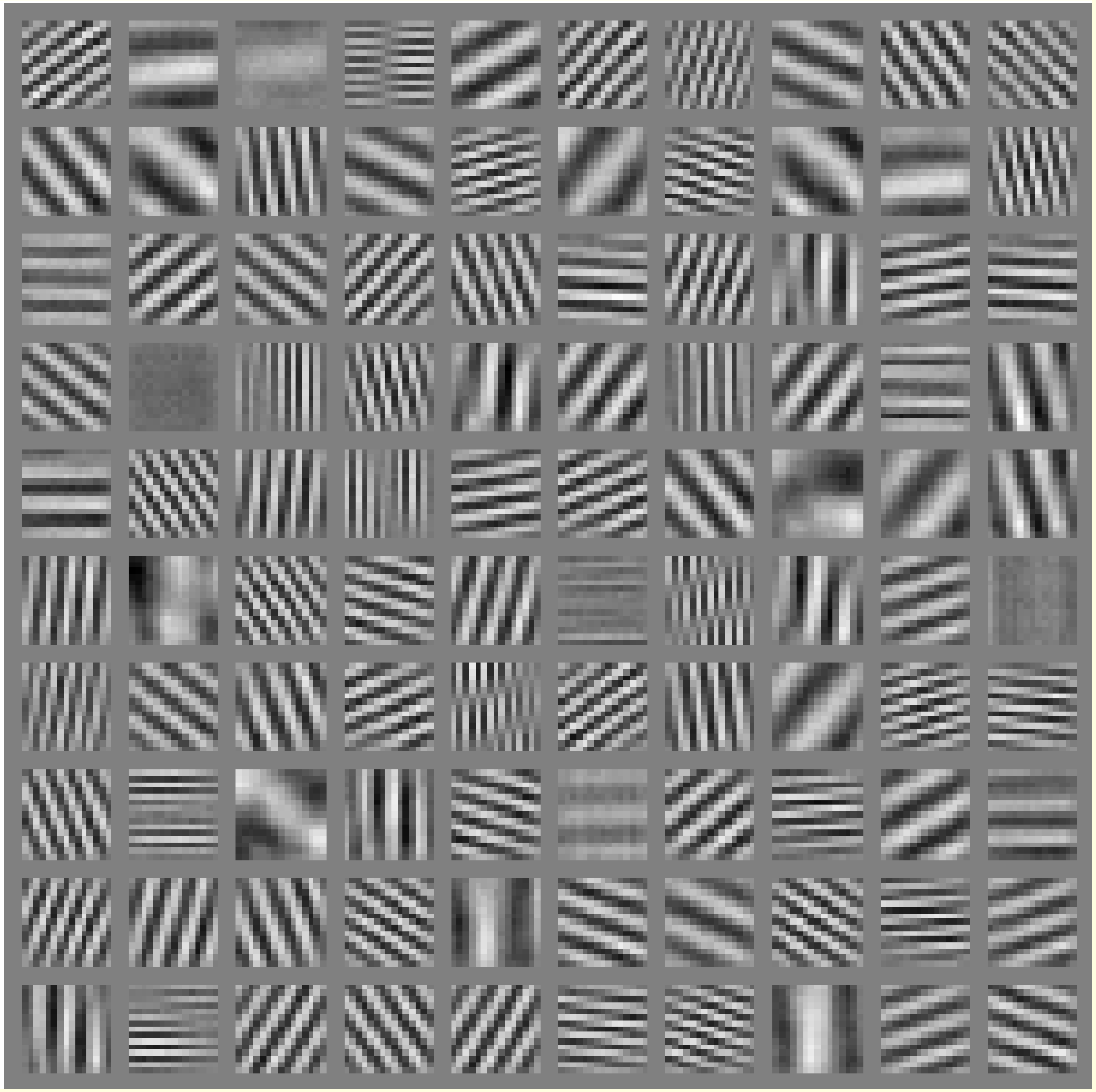
A nasty numerical problem

- In a standard Boltzmann machine the gradient of a weight on a training case always lies between 1 and -1.
- With factored three-way interactions, the gradient contains the product of two sums each of which can be large, so the gradient can explode.
- We can keep a running average of each sum over many training cases and divide the gradient by this average (or its square). This helps.
  - For any particular weight, we must divide the gradient by the same quantity on all training cases to guarantee a positive correlation with the true gradient.
- Updating the weights on every training case may also help because we get feedback faster when weights are blowing up.

Showing what a factor learns by alternating between its pre- and post- fields

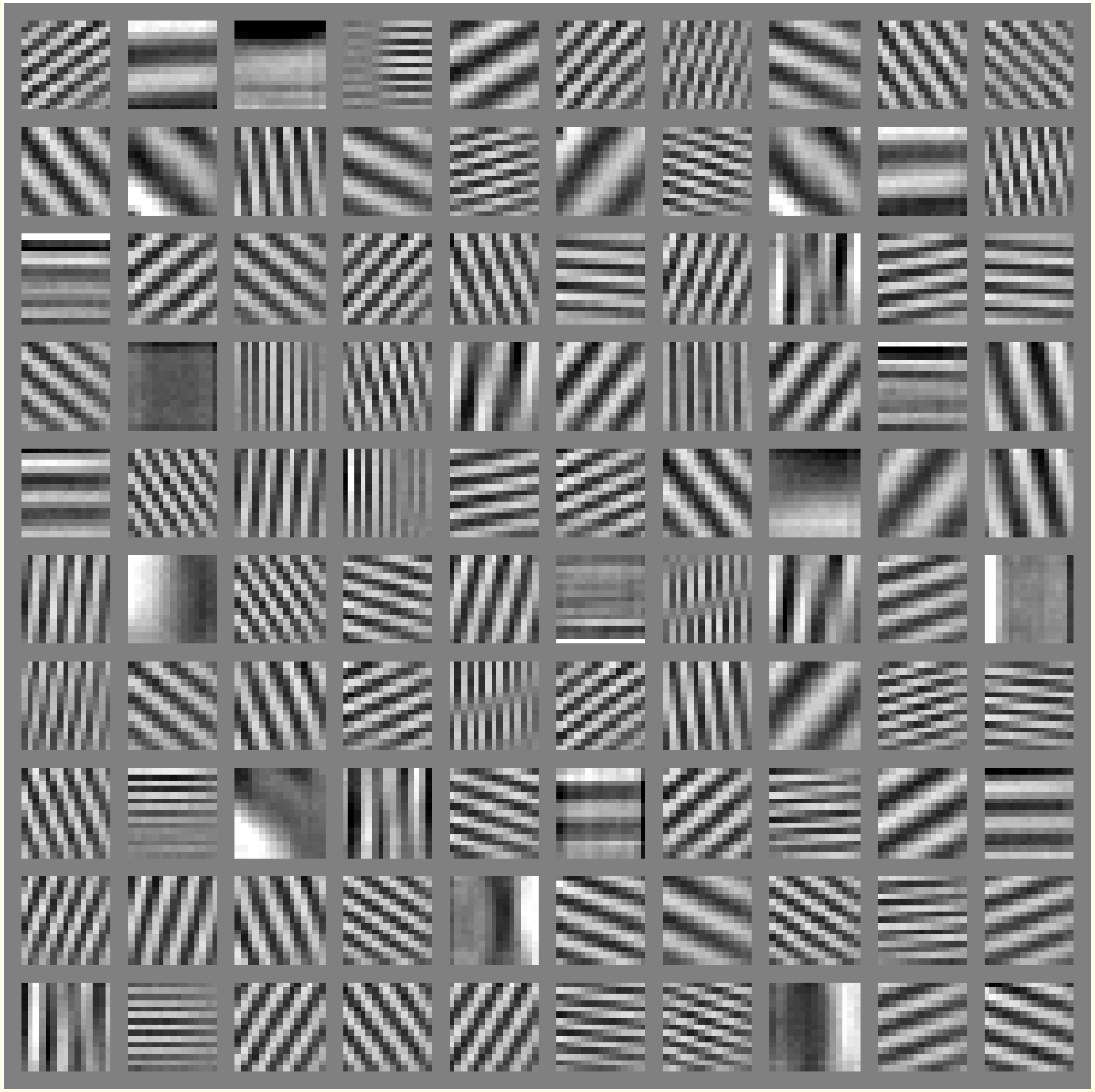


The factor receptive fields

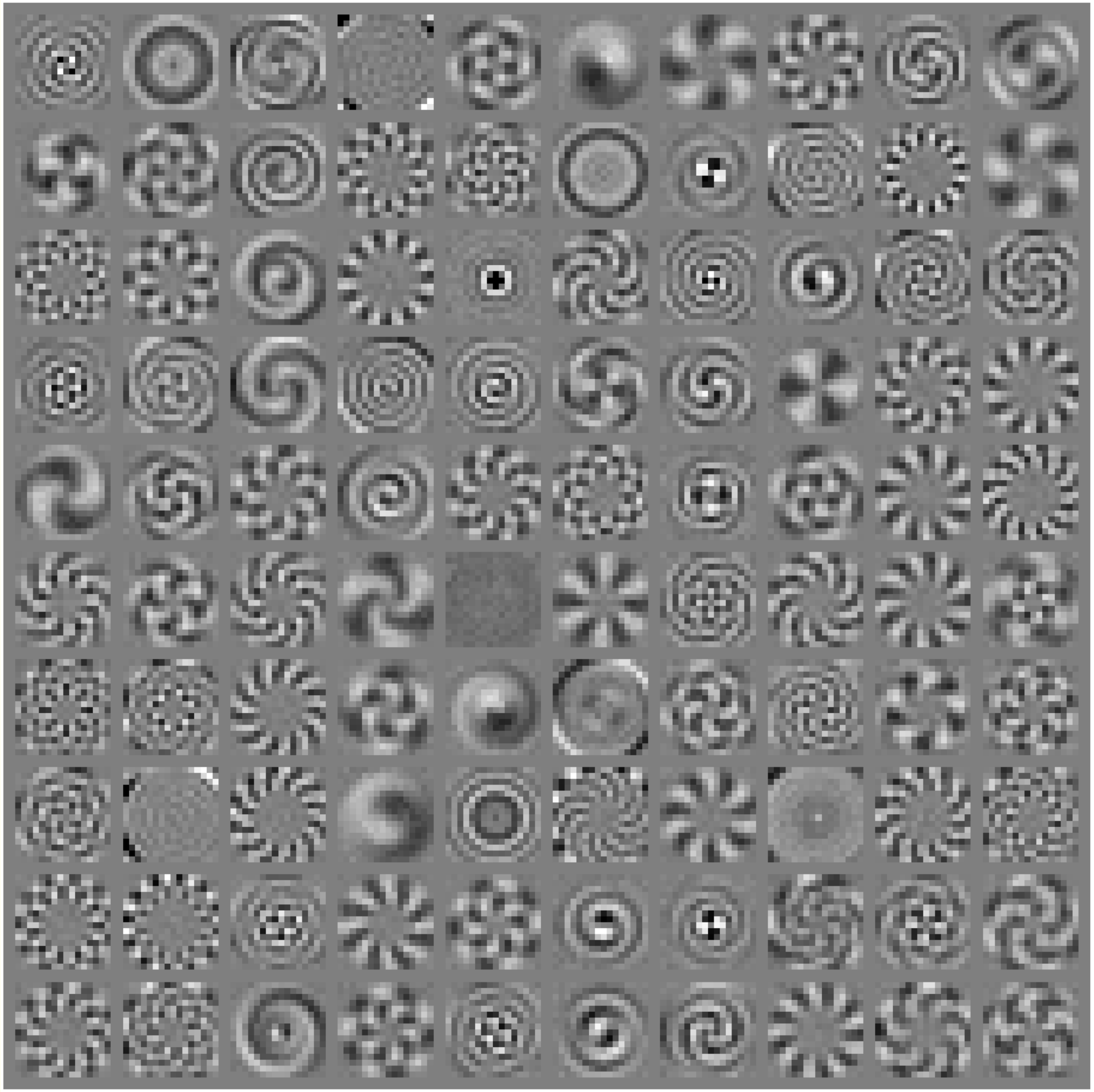


The network is trained on translated random dot patterns.

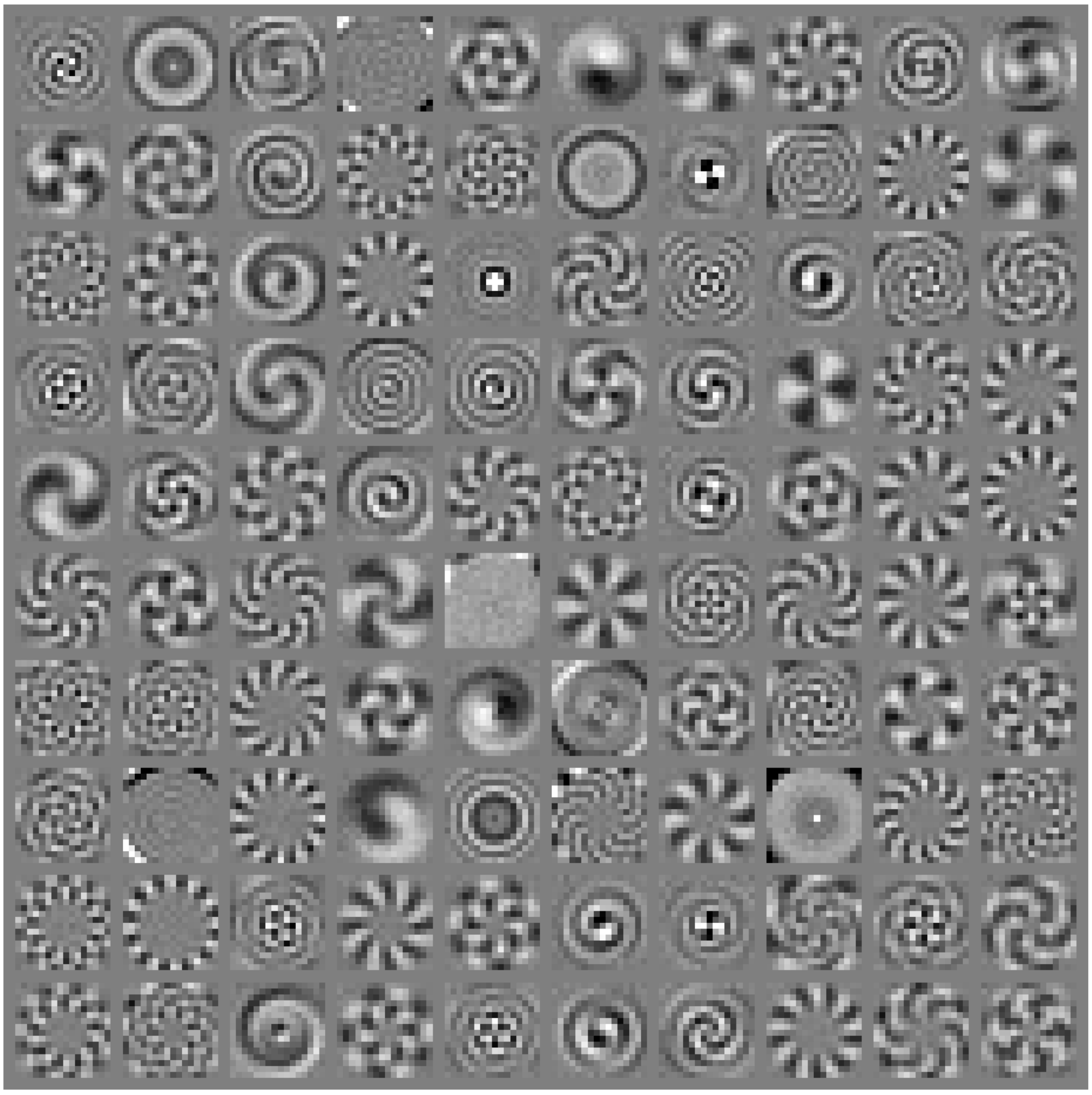
The factor receptive fields



The network is trained on translated random dot patterns.



The network is trained on rotated random dot patterns.



The network is trained on rotated random dot patterns.

How does it perceive two overlaid sparse dot patterns moving in different directions?

- First we train a second hidden layer. Each of these units prefers motion in a different direction.
- Then we compute the perceived motion by adding up the preferences of the active units in the second hidden layer.
- If the two motions are within about 30 degrees it sees a single average motion.
- If they are further apart it sees two separate motions.
  - The separate motions are slightly further apart than the real ones.
  - This is just like human perception and it was not trained on transparent motion.
  - The training is entirely unsupervised.

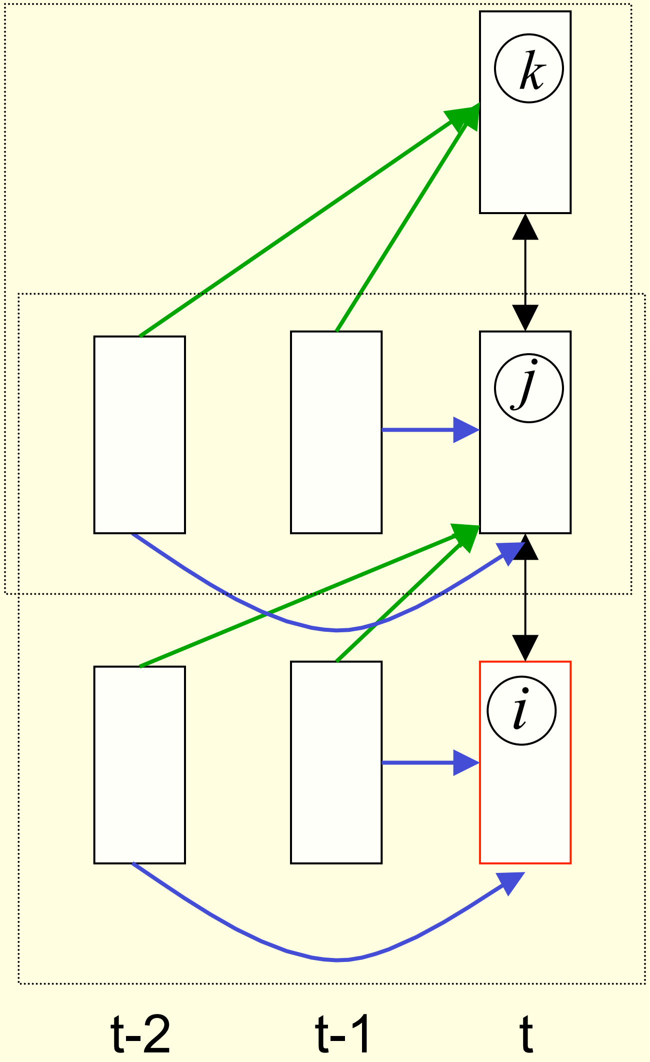
An application to modeling motion capture data

- Human motion can be captured by placing reflective markers on the joints
  - Use lots of infrared cameras to track the 3-D positions of the markers

- Given a skeletal model, the 3-D positions of the markers can be converted into
  - The joint angles
  - The 3-D translation of the pelvis
  - The roll, pitch and delta yaw of the pelvis

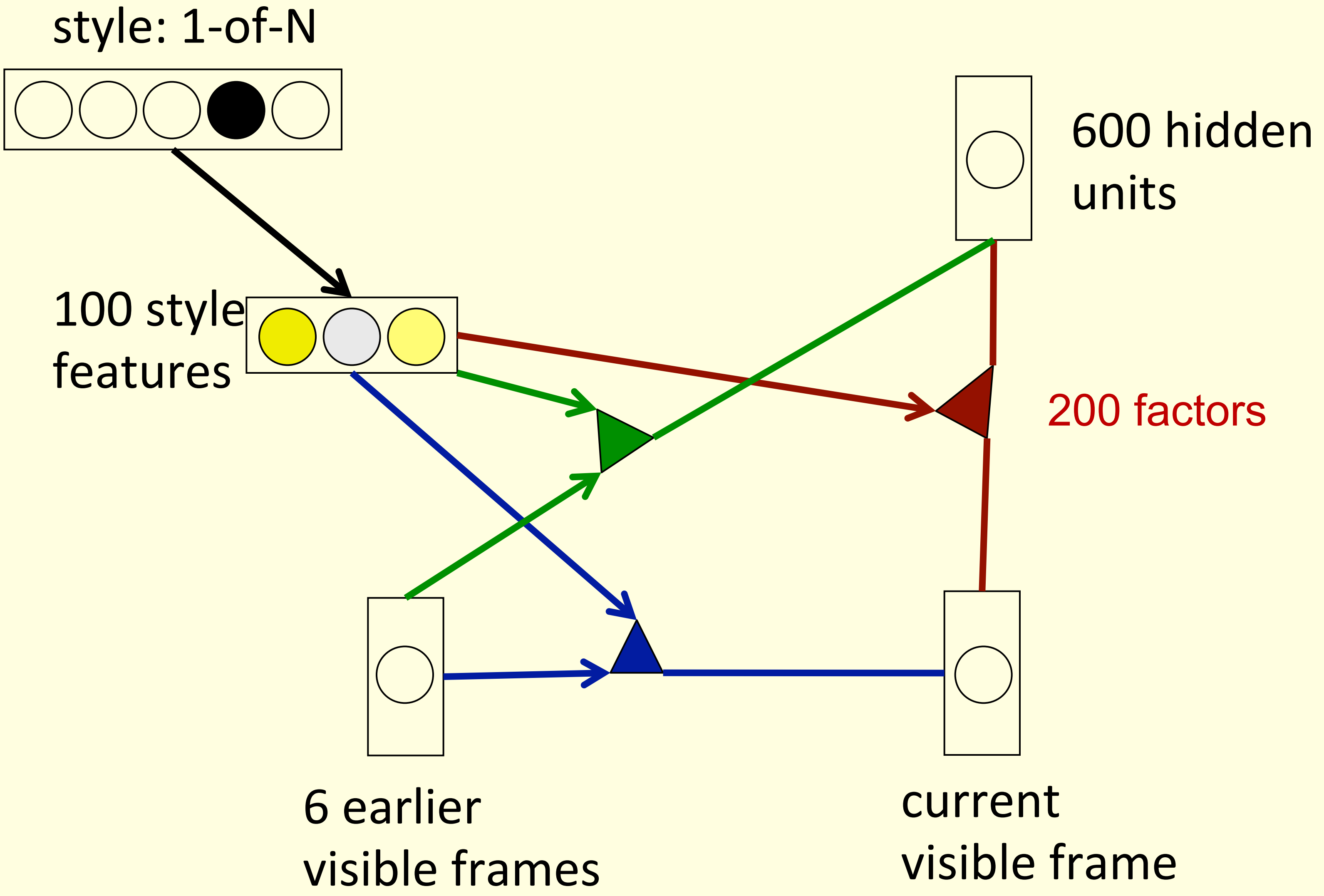
Higher level models

- Once we have trained the model, we can add more layers.
- Treat the hidden activities of the first CRBM as data for training the next CRBM.
  - Add "autoregressive" connections to a layer when it becomes the visible layer.
- Adding a second layer makes it generate more realistic sequences.



Using a style variable to modulate the interactions

(there is additional weight sharing: Taylor&Hinton, ICML 2009)


style: 1-of-N
600 hidden units
100 style features
200 factors
6 earlier visible frames
current visible frame