Backprop: Representations, Representations, Representations

Garrison W. Cottrell
Gary's Unbelievable Research Unit (GURU)
Computer Science and Engineering Department
Temporal Dynamics of Learning Center
Institute for Neural Computation
UCSD

Characteristics of perceptron learning

- Supervised learning: Gave it a set of input-output examples for it to model the function (a *teaching signal*)
- Error correction learning: only correct it when it is wrong.
- Random presentation of patterns.
- Slow! Learning on some patterns ruins learning on others.

Perceptron Learning

- Learning rule:

$$w_i(\mathrm{t+1}) = w_i(\mathrm{t}) + \alpha(t - y)x_\mathrm{i}$$

t == time,
t==target

(α is the *learning rate*)

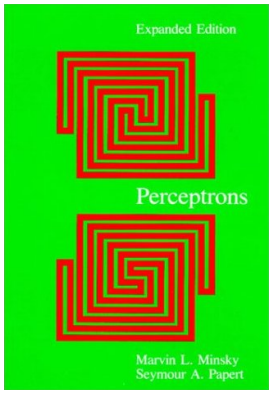
- This is known as the *delta rule* because learning is based on the *delta* (difference) between what you did and what you should have done: δ = (teacher - output):

$$w_i(\mathrm{t+1}) = w_i(\mathrm{t}) + \alpha\delta x_\mathrm{i}$$

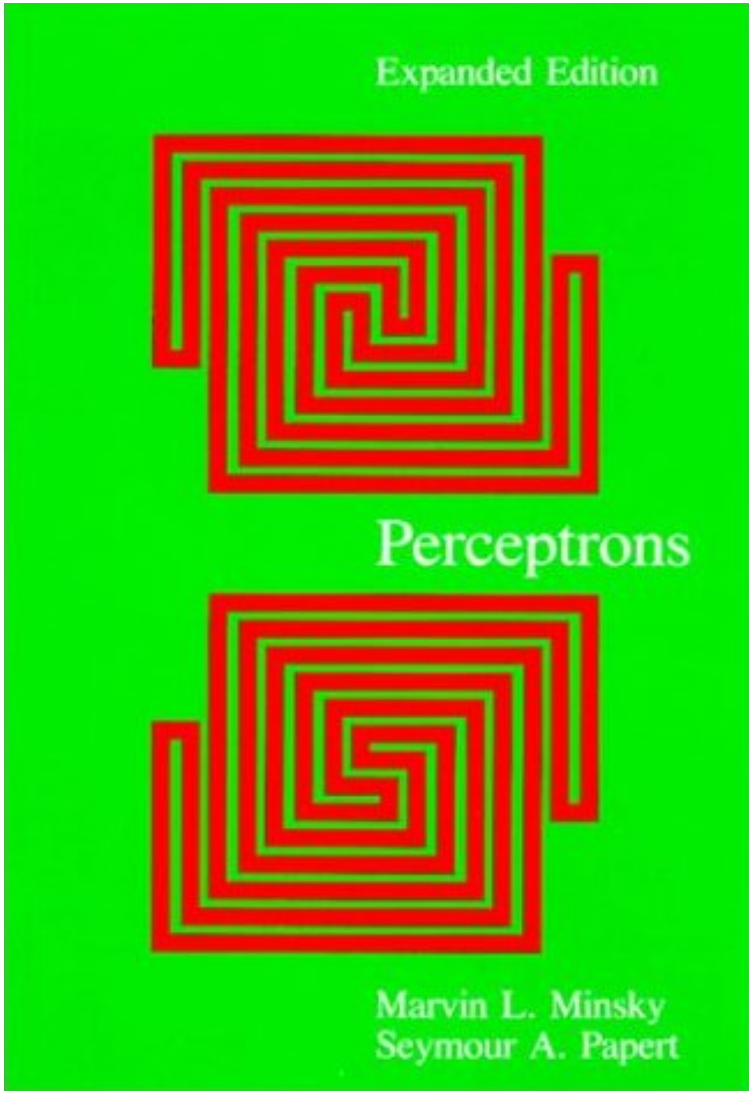
Problems with perceptrons

- The learning rule comes with a great guarantee: anything a perceptron can *compute*, it can *learn to compute.*
- Problem: Lots of things were not computable,

  e.g., XOR (Minsky & Papert, 1969)
- Minsky & Papert said:
  - if you had hidden units, you could compute *any* boolean function.
  - But no learning rule exists for such multilayer networks, *and we don't think one will ever be discovered.*



Problems with perceptrons



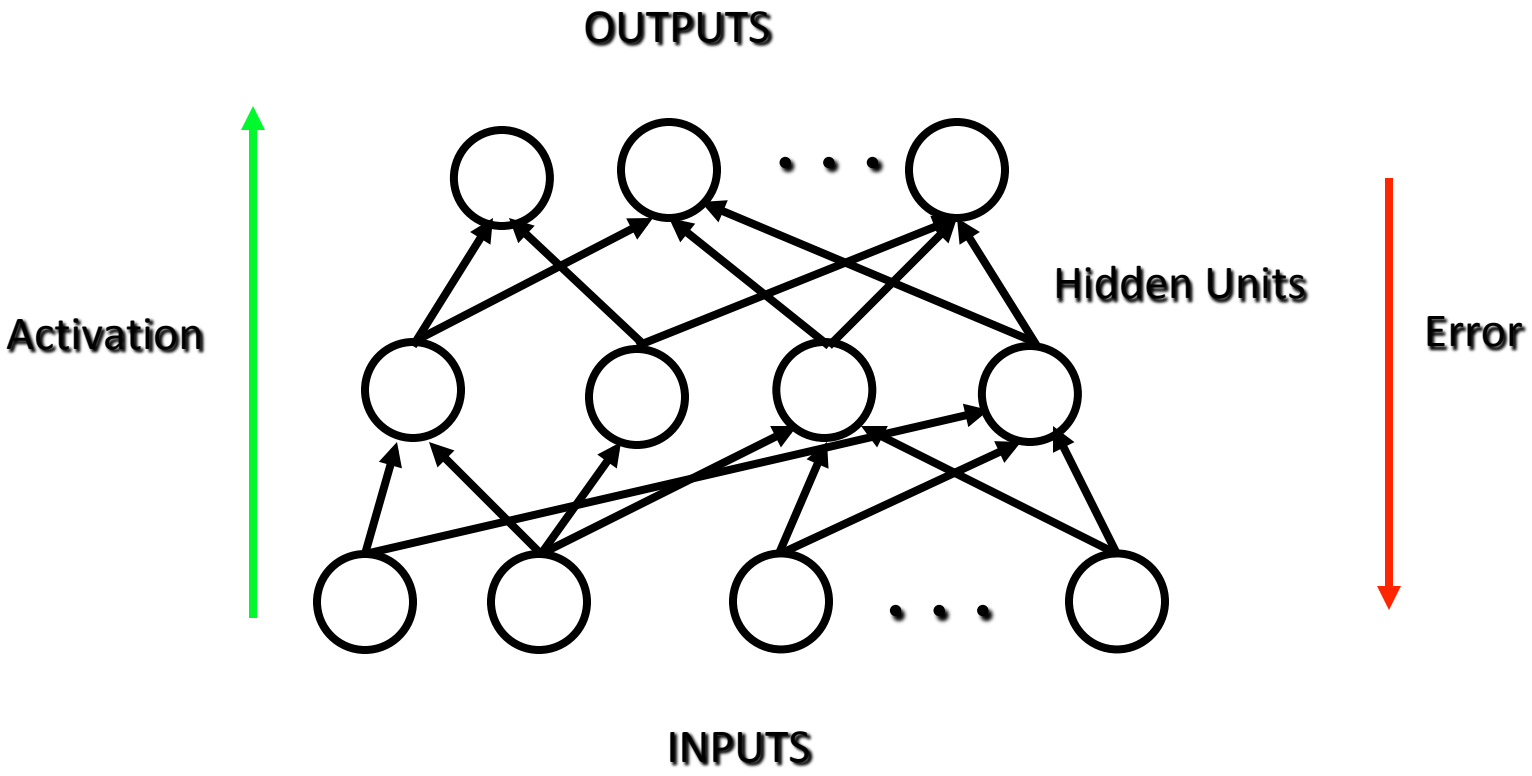
Aside about perceptrons

- They didn't have hidden units - but Rosenblatt assumed nonlinear preprocessing!
- Hidden units compute features of the input
- The nonlinear preprocessing is a way to choose features by hand.
- Support Vector Machines essentially do this in a principled way, followed by a (highly sophisticated) perceptron learning algorithm.

Enter Rumelhart, Hinton, & Williams (1985)

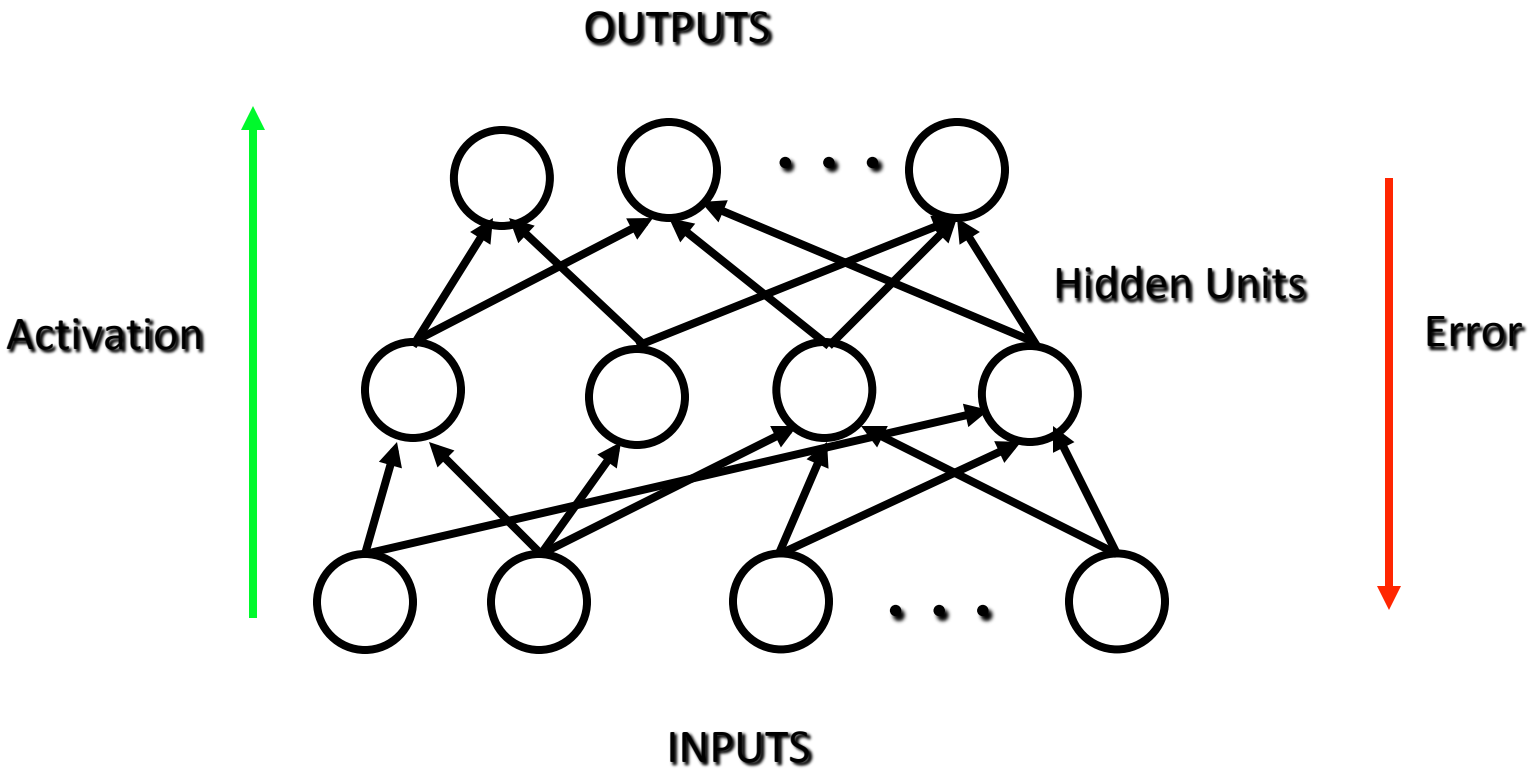
- (Re-)Discovered a learning rule for networks with hidden units.
- Works a lot like the perceptron algorithm:
  - Randomly choose an input-output pattern
  - present the input, let activation propagate through the network
  - give the *teaching signal*
  - propagate the error back through the network (hence the name *back propagation*)
  - change the connection strengths according to the error

Enter Rumelhart, Hinton, & Williams (1985)



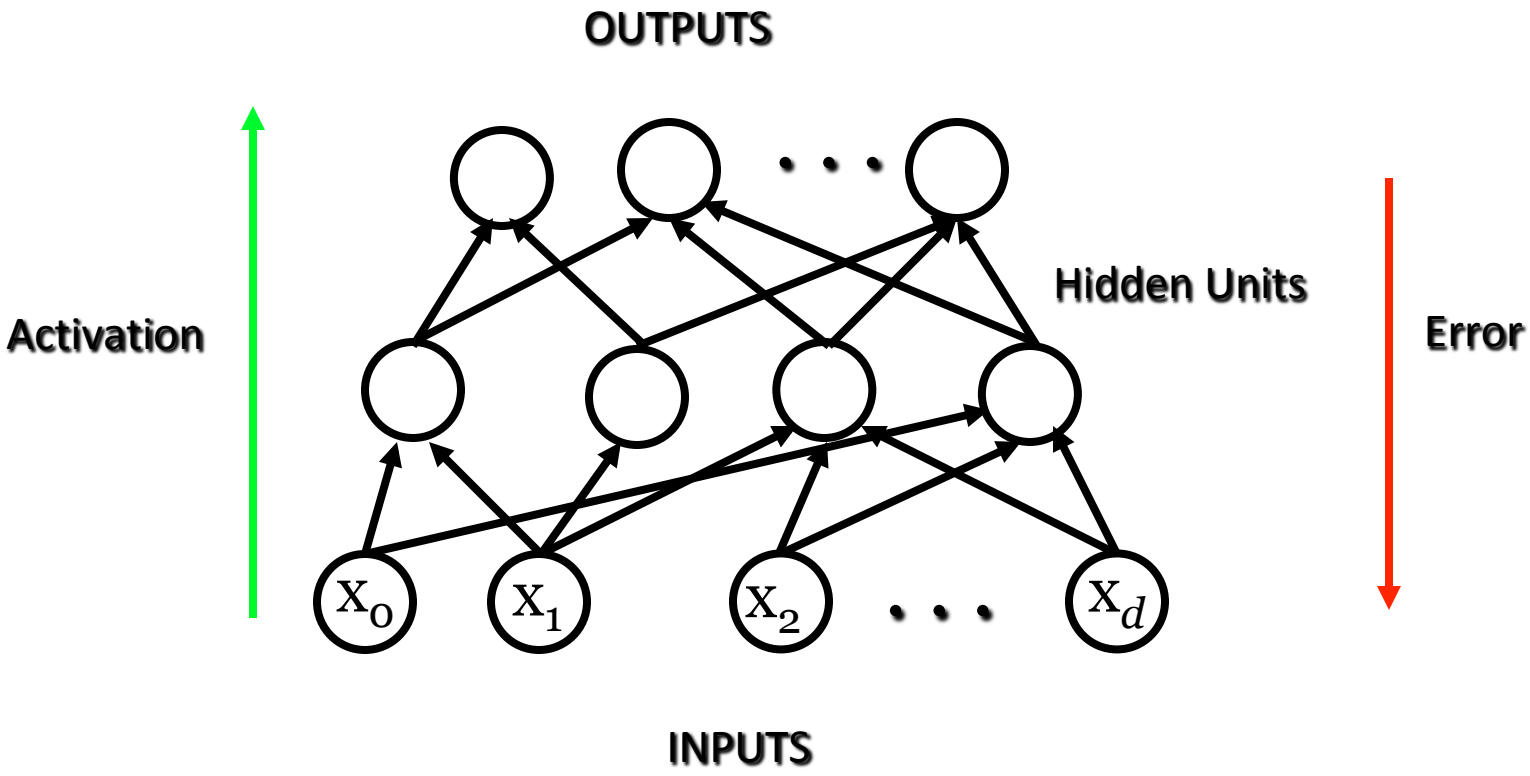
- The actual algorithm uses the chain rule of calculus to go *downhill* in an error measure with respect to the weights
- The hidden units must learn features that solve the problem

But first, let's talk about *forward* propagation!

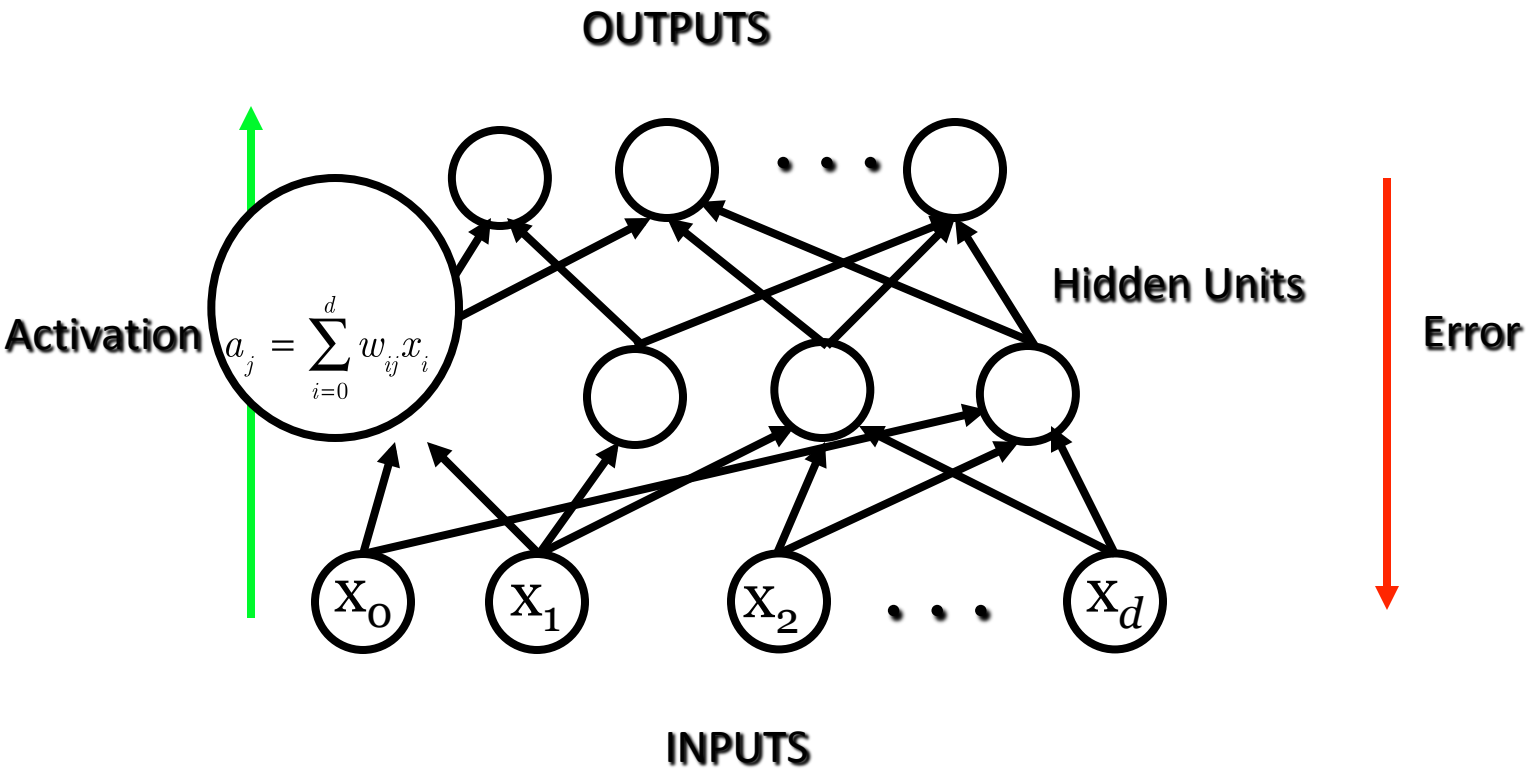


How does activation propagate forward in the network?

Start with the input pattern (e.g., MNIST pixel values)

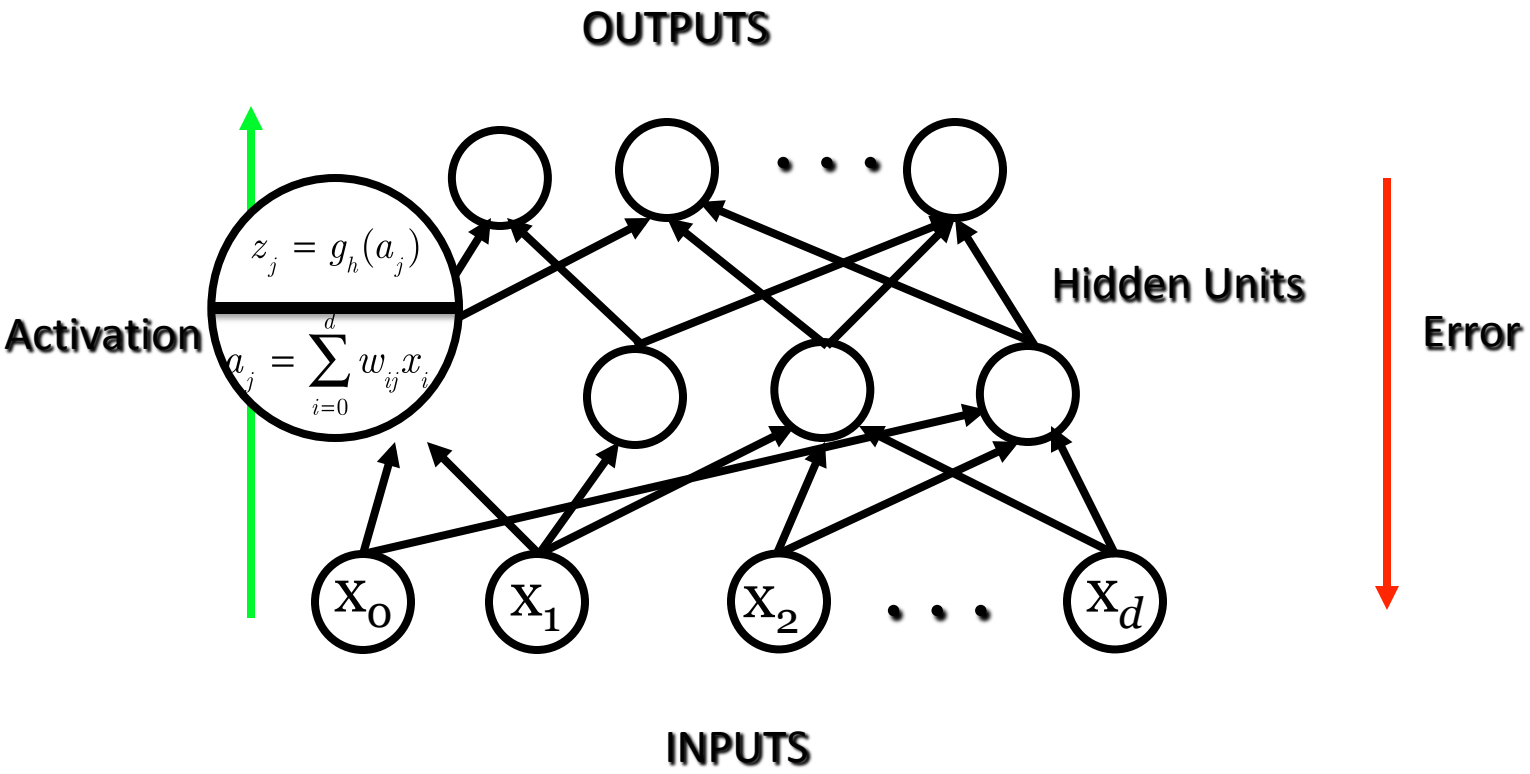


Start with the input pattern (e.g., MNIST pixel values)



First, compute the weighted sum of the inputs to the hidden units
Call that a_j as before (the *net input* to the hidden unit)

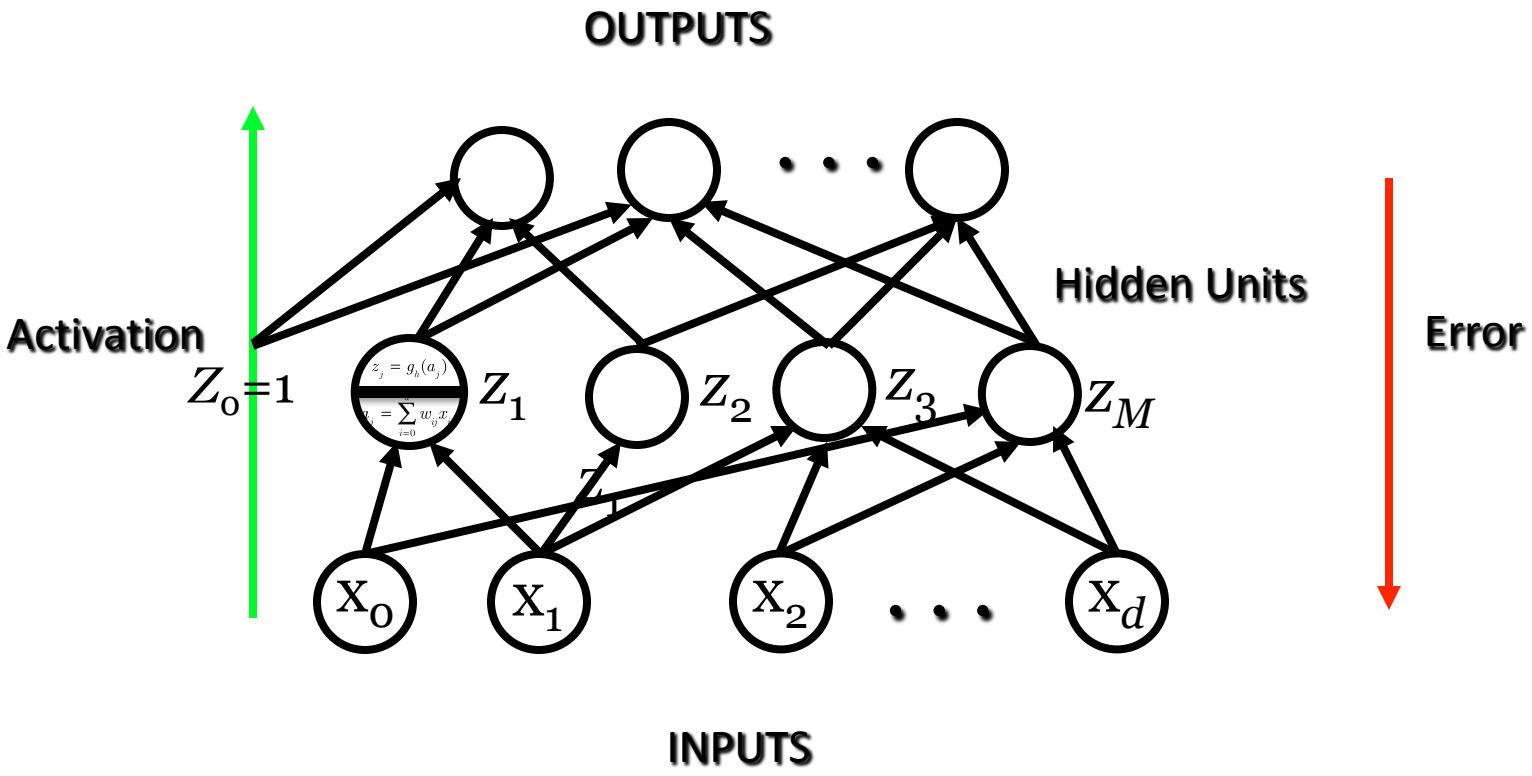
Start with the input pattern (e.g., MNIST pixel values)



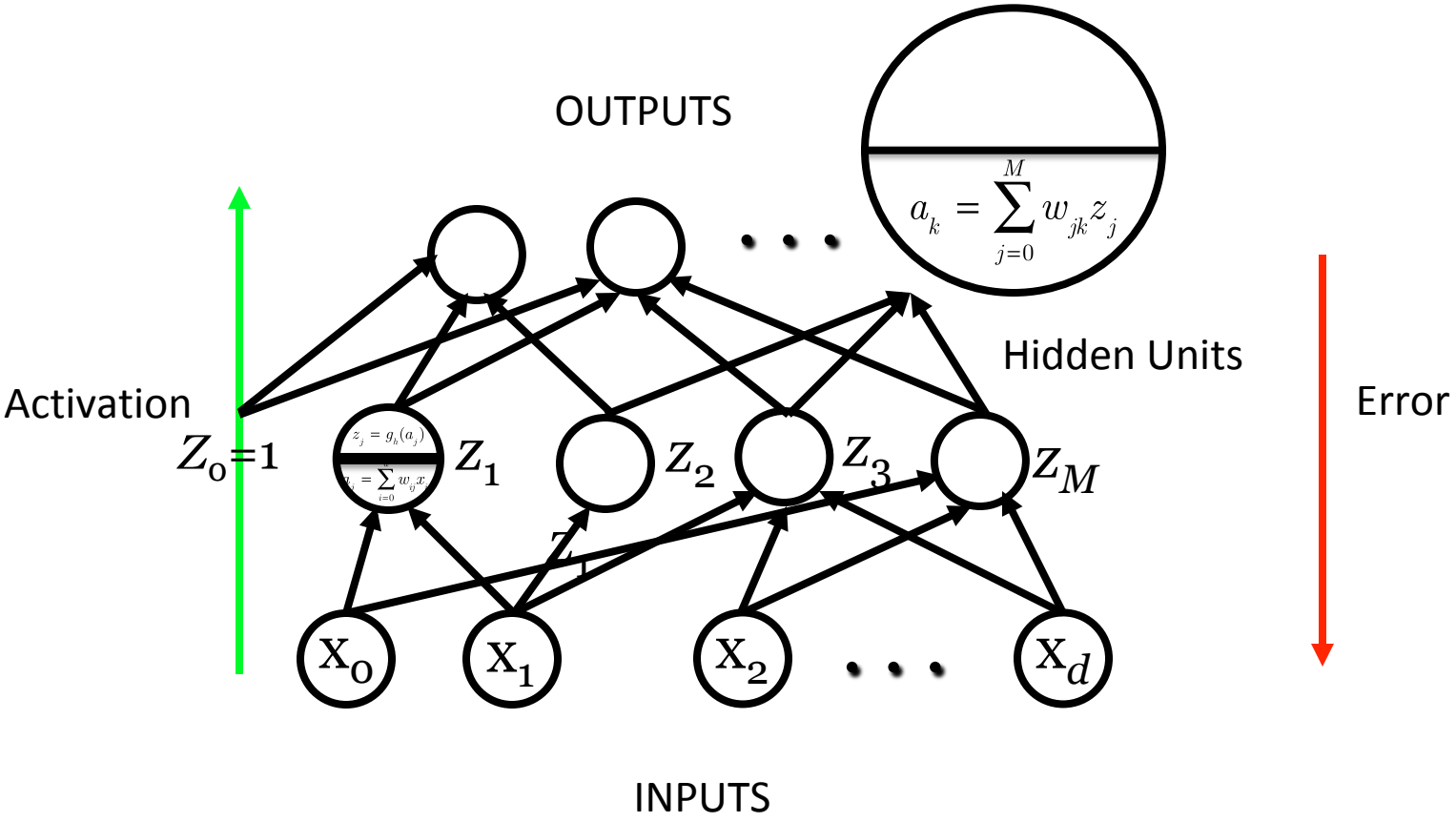
Now, compute the nonlinear activation function of a_j (could be logistic, tanh, ReLU)

z_j is the *output* of the hidden unit

Ok, now we have the activations of the hidden units…

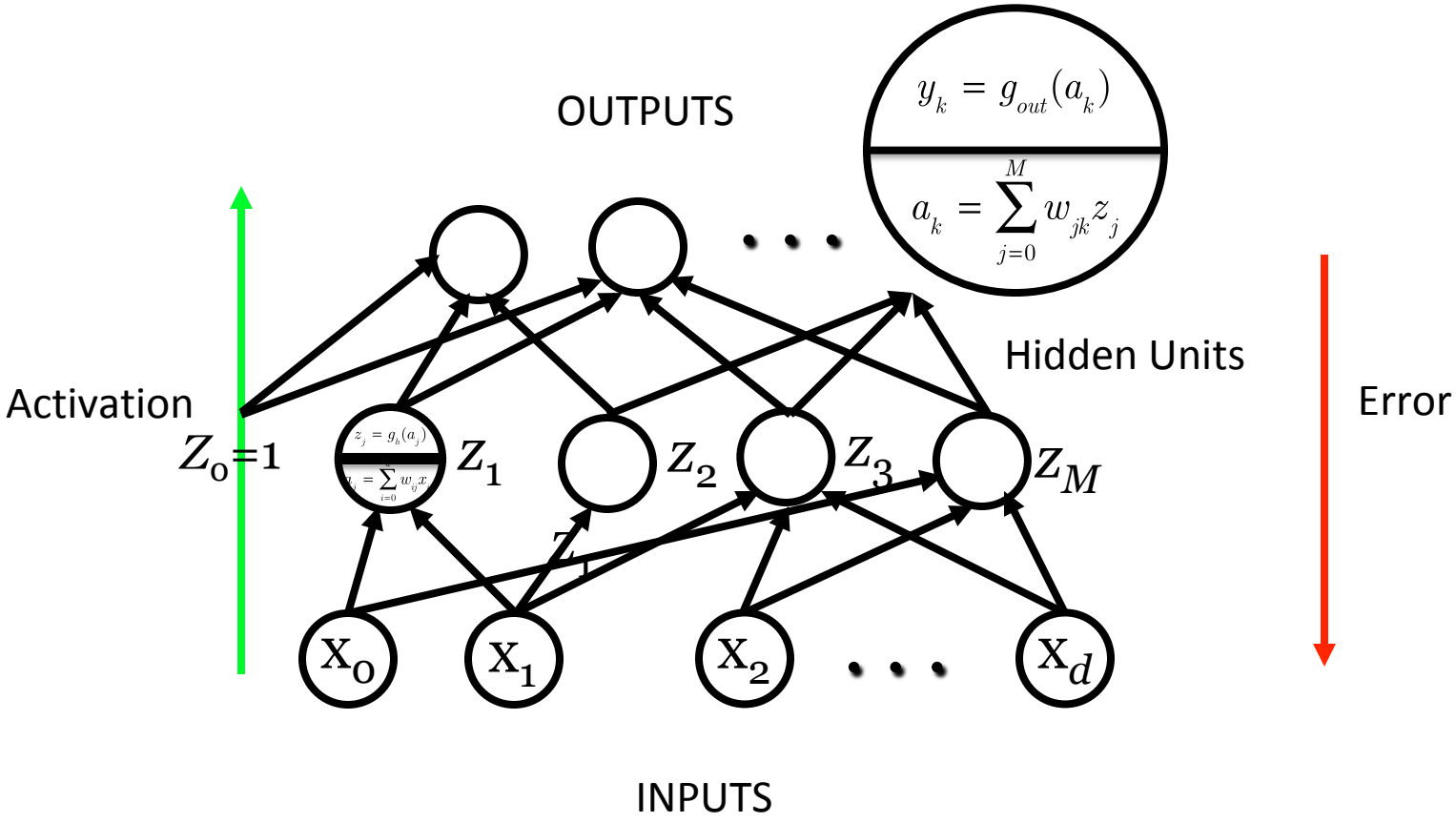


Now, compute the net input to the output units...



Compute the weighted sum of the inputs to the output units
Call that a_k as before (the *net input* to the output unit)

Now, compute the net input to the output units...



g_{out} is the activation function of the outputs...
(softmax, linear, logistic...)

Backpropagation Learning

- We will still do gradient descent:

$$w_{ij} = w_{ij} - \frac{\partial J}{\partial w_{ij}}$$

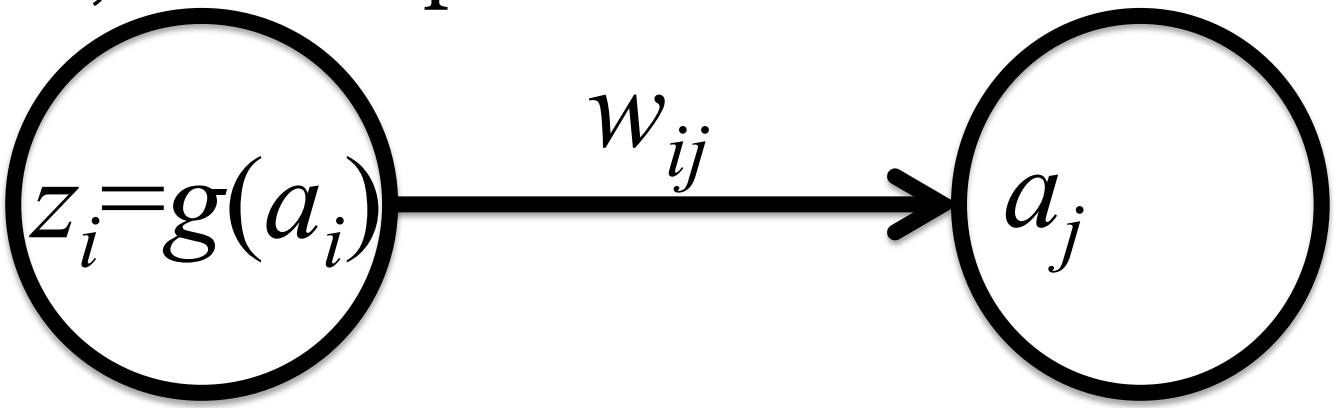
- But now, we need to take into account the hidden units!

- **Notation**: z_i will mean *either* the output of a hidden unit, or an input.

The picture to have in mind:

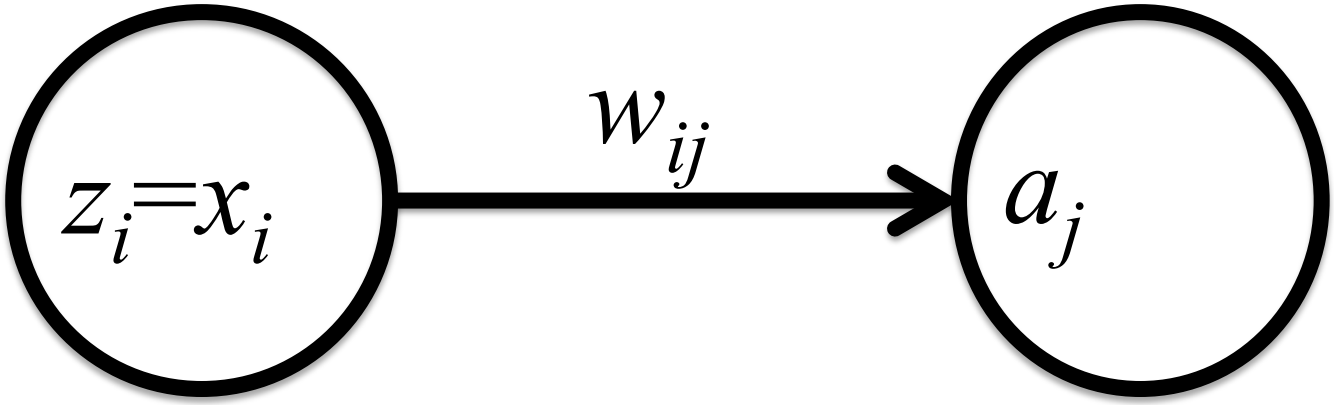
- **Notation**: z_i will mean *either* the output of a hidden unit, or an input.

i is a hidden unit



OR:

i is an input



Derivation of Backprop…

- J is the objective function.
- We want: $\frac{\partial J}{\partial w_{ij}}$ where w_{ij} is any weight in the network
- As usual, we start with:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial a_j} \frac{\partial a_j}{\partial w_{ij}}$$

Derivation of Backprop…

- Recall:

$$\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial \sum_{k=0}^{H} w_{kj} z_k}{\partial w_{ij}} = \frac{\sum_{k=0}^{H} \partial w_{kj} z_k}{\partial w_{ij}} = z_i$$

Derivation of Backprop…

- So we have:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} = \frac{\partial J}{\partial a_j} z_i$$

- As before, we have a term that is the input on the line from i to j.

Derivation of Backprop…

- Now, we *define*:

$$\text{define } \delta_j = -\frac{\partial J}{\partial a_j}$$

- So we have:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} = -\delta_j z_i$$

Derivation of Backprop…

- Gradient descent says (ignoring the learning rate for now):

$$w_{ij} = w_{ij} - \frac{\partial J}{\partial w_{ij}}$$

$$= w_{ij} + \delta_j z_i$$

- And we know for output units, this comes out to: $= w_{ij} + (t_j - y_j) z_i$

Derivation of Backprop…

- For hidden units, we have to take into account every unit that unit *j* sends output to:

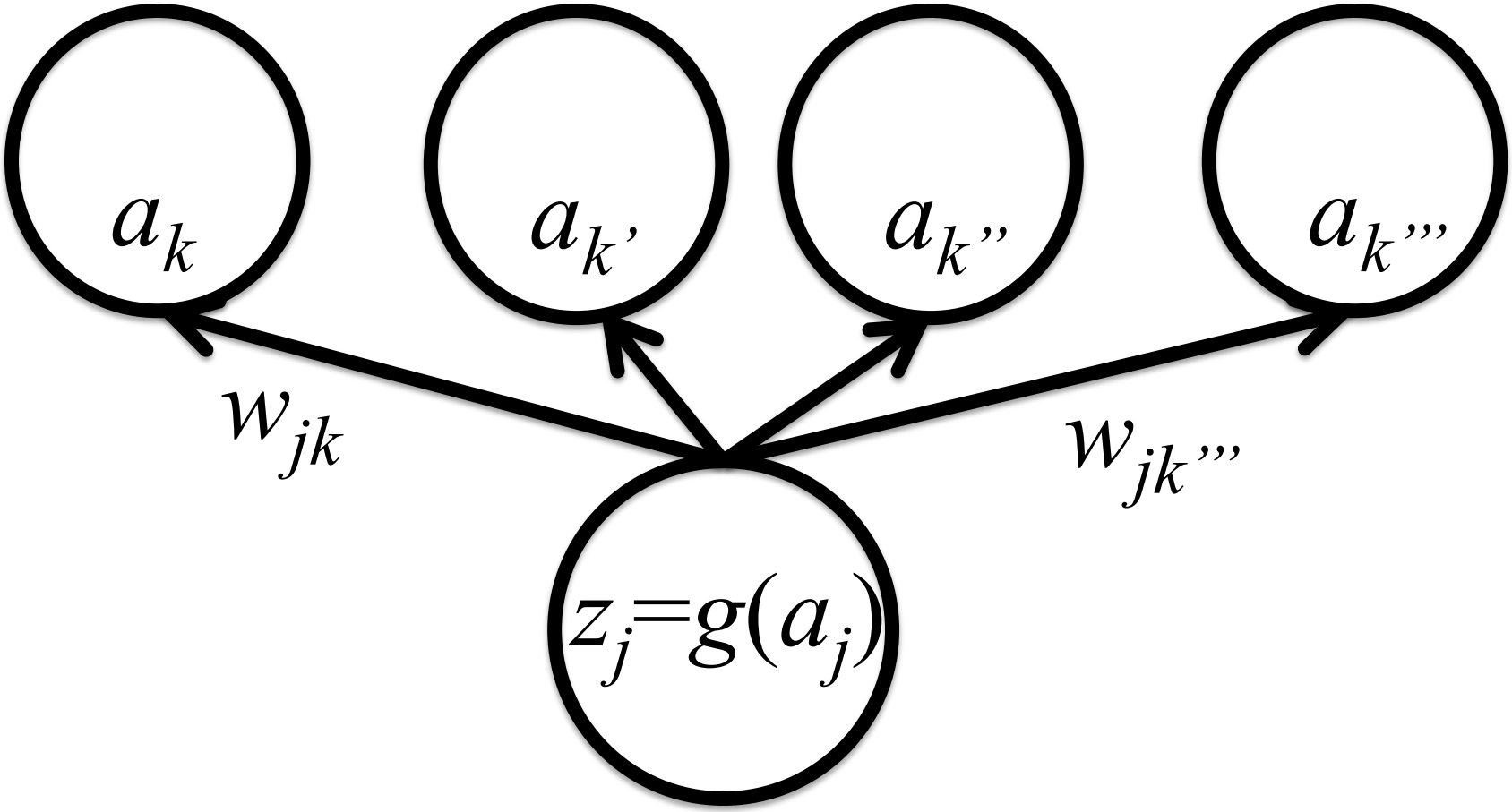
$$\frac{\partial J}{\partial a_j} = \sum_k \frac{\partial J}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

- To know how changing my input changes the error, I have to know
  - How the error changes as the input to the units I am connected to change,
  - times how their input changes as my input changes

Derivation of Backprop…

- This idea in pictures:

$$\frac{\partial J}{\partial a_j} = \sum_k \frac{\partial J}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$



Derivation of Backprop…

- Moving along: $\frac{\partial J}{\partial a_j} = \sum_k \frac{\partial J}{\partial a_k}\frac{\partial a_k}{\partial a_j}$

Definition of delta $= -\sum_k \delta_k \frac{\partial a_k}{\partial a_j}$

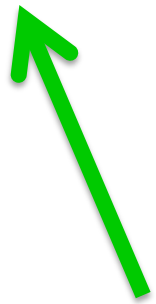
Chain rule $= -\sum_k \delta_k \frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j}$

j is independent of k $= -\frac{\partial z_j}{\partial a_j}\sum_k \delta_k \frac{\partial a_k}{\partial z_j}$

Definition of a_k $= -\frac{\partial z_j}{\partial a_j}\sum_k \delta_k \sum_i \frac{\partial w_{ik} z_i}{\partial z_j} = -\frac{\partial z_j}{\partial a_j}\sum_k \delta_k w_{jk}$

and:

$$\frac{\partial z_j}{\partial a_j} = \frac{\partial g(a_j)}{\partial a_j} = g'(a_j)$$



Every term in the sum is 0 except when $i=j$

Derivation of Backprop…

- SO: $\frac{\partial J}{\partial a_j} = -g'(a_j)\sum_k \delta_k w_{jk}$ if j is a hidden unit
- But since delta is $-\frac{\partial J}{\partial a_j}$, $\delta_j = g'(a_j)\sum_k \delta_k w_{jk}$

Backpropagation Learning

- Learning rule: $$w_{ij} = w_{ij} - \alpha \frac{\partial J}{\partial w_{ij}} = w_{ij} + \alpha \delta_j z_i$$

  (α is the *learning rate*)

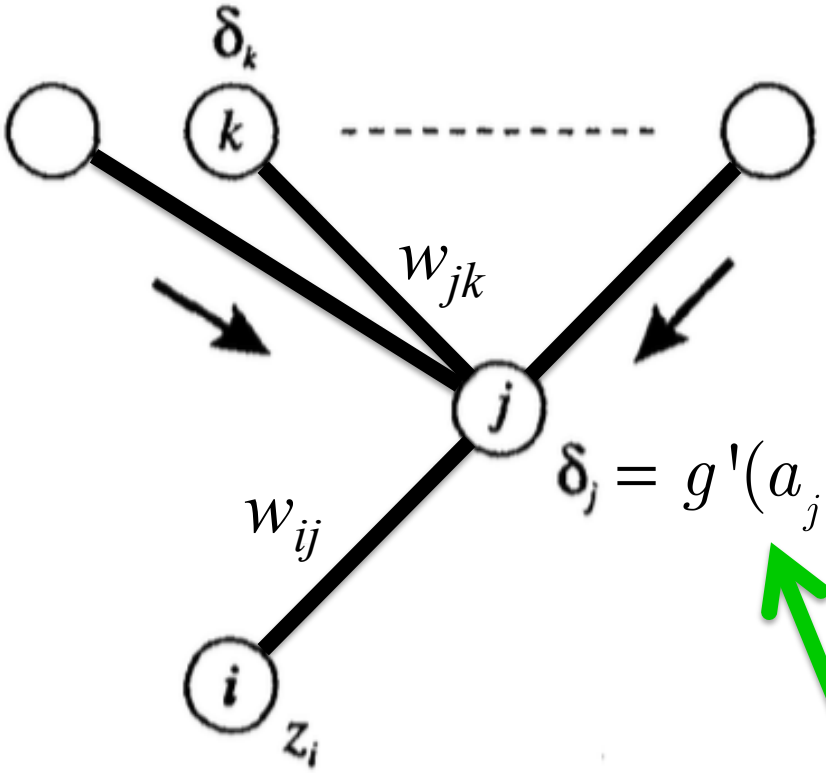
- Here: $\delta_j = (t_j - y_j)$ if j is an output unit*

  $$\delta_j = g'(a_j) \sum_k \delta_k w_{jk}$$ if j is a hidden unit

- So, we have a recursive definition of delta

*assuming the right combination of objective function and output activation function

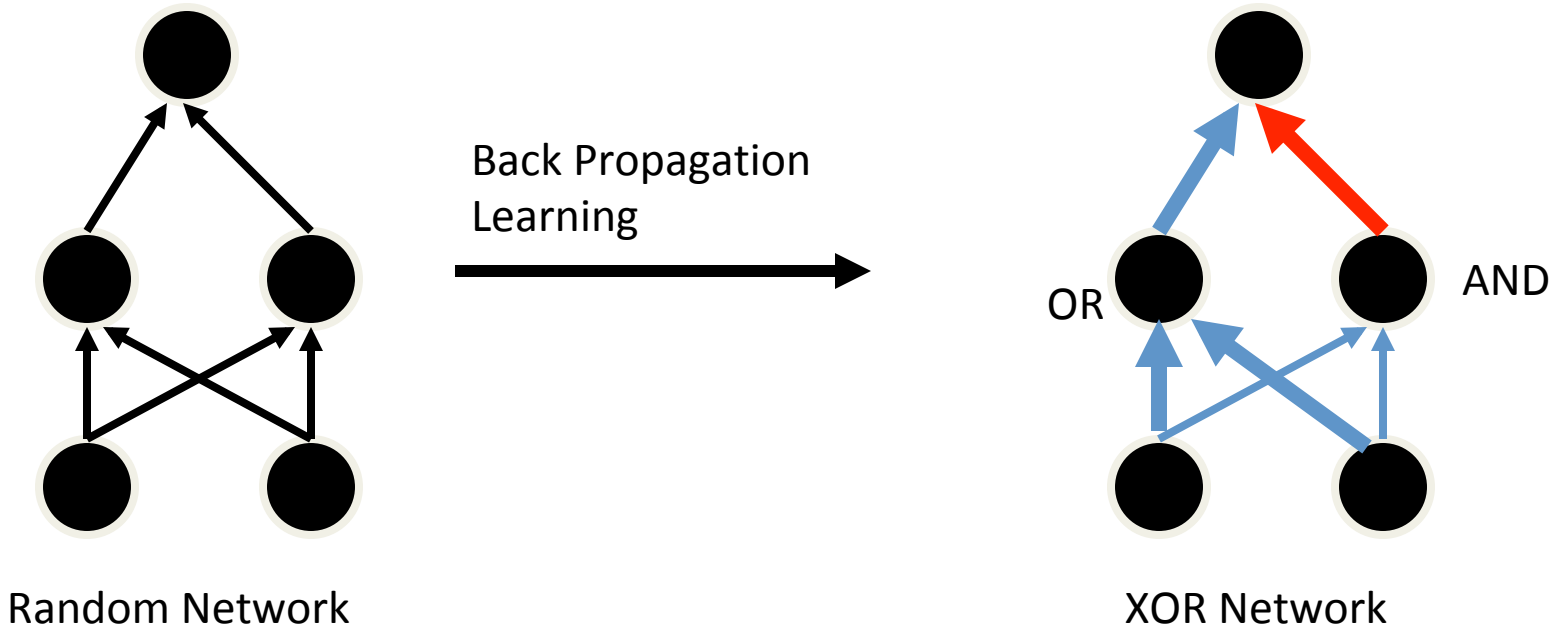
For hidden units, this is the picture to have in mind:



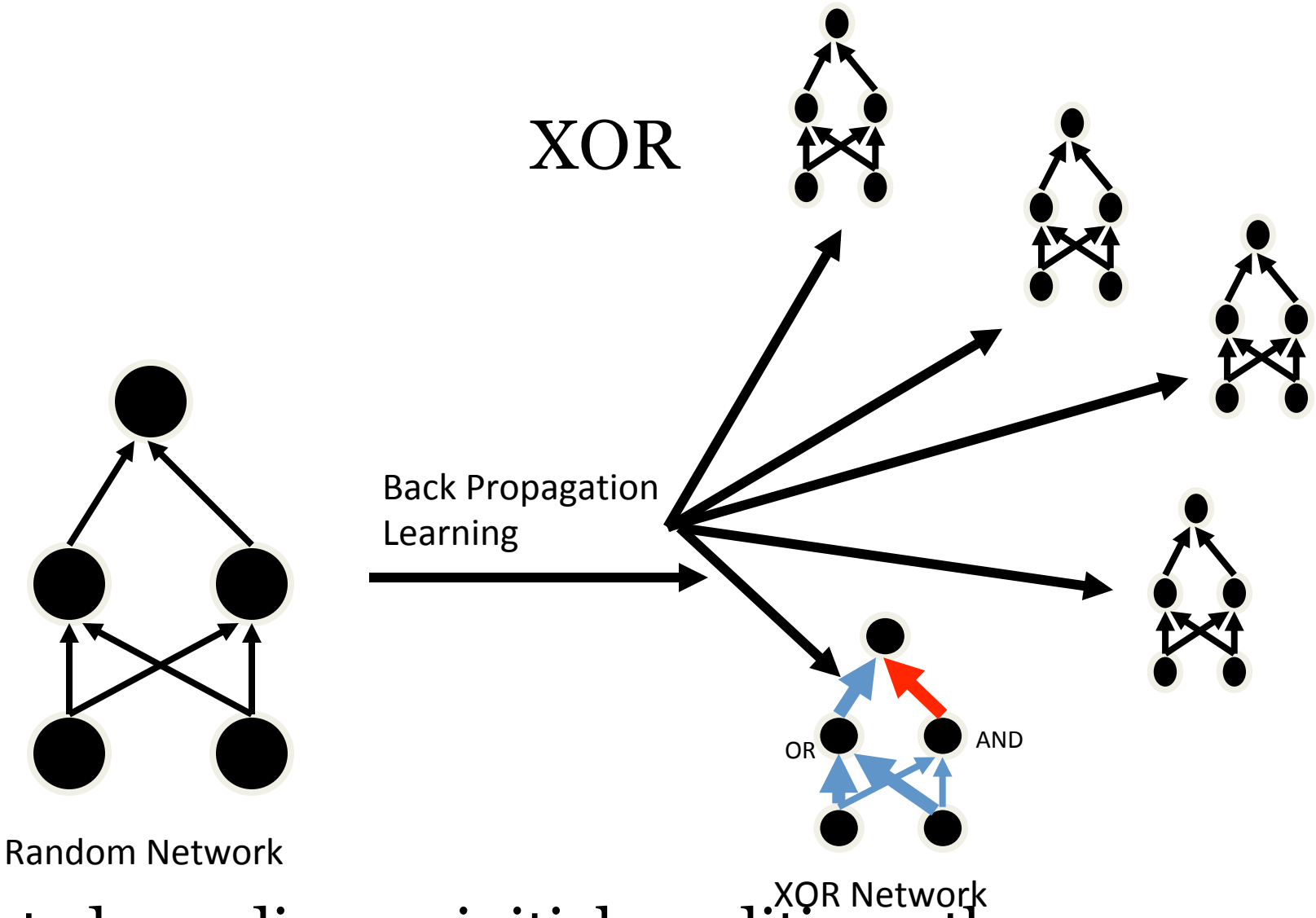
$$\delta_j = g'(a_j)\sum_k \delta_k w_{jk}$$

Note! You can't get rid of **the slope term** for hidden units!

XOR



Here, the hidden units learned AND and OR - two features that when combined appropriately, can solve the problem



But, depending on initial conditions, there are an infinite number of ways to do XOR - backprop can surprise you with innovative solutions.

Clicker Questions

1. The tl;dr of backpropagation is:

A. It's the opposite of forward propagation

B. It learns internal representations (features)

C. It is a form of gradient ascent

D. It changes the activations to go downhill in the parameters

Clicker Questions

1. The tl;dr of backpropagation is:

A. It's the opposite of forward propagation

B. It learns internal representations (features)

C. It is a form of gradient ascent

D. It changes the activations to go downhill in the parameters

Clicker Questions

2. Computing the deltas

A. Starts at the output, they are propagated backwards, and the weights are changed
B. Starts at the output, the weights are changed, and the deltas are propagated backwards
C. Uses the slope of the output units
D. Uses the slope of the hidden units
E. A&D

Clicker Questions

2. Computing the deltas

A. Starts at the output, they are propagated backwards, and the weights are changed
B. Starts at the output, the weights are changed, and the deltas are propagated backwards
C. Uses the slope of the output units
D. Uses the slope of the hidden units
E.A&D

Why is/was this wonderful?

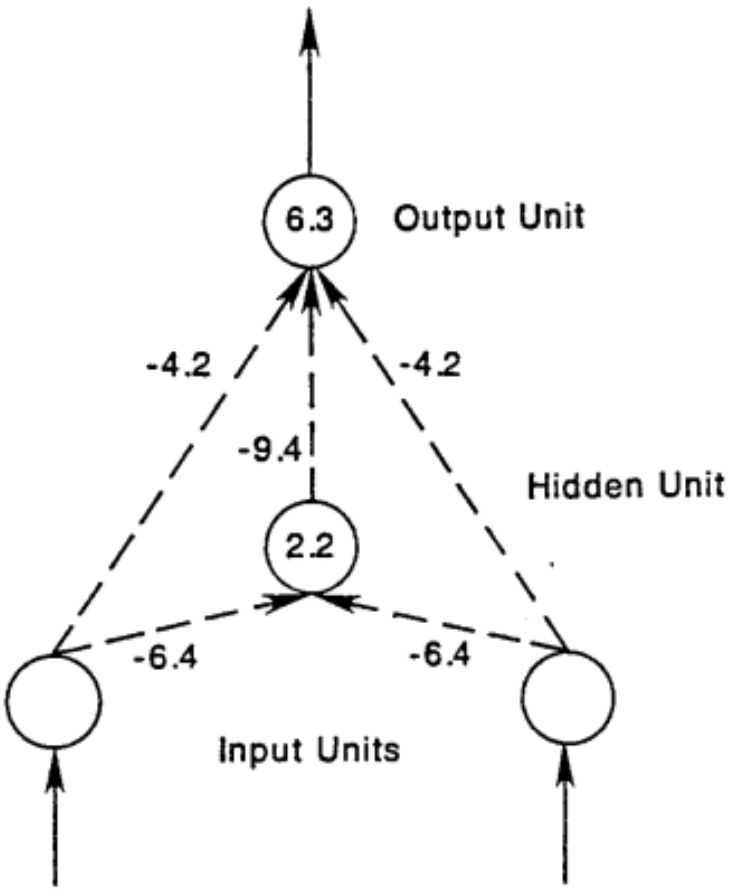
- Learns internal representations
- Learns internal representations
- Learns internal representations
- Efficient
- Generalizes to ***recurrent networks***

Representations

- In the next *k* slides, where *k* is some medium-sized integer, we will look at
  - various representations backprop has learned, and
  - problems backprop has solved
- The mantra here is:
  - Backprop learns representations in the service of the task

XOR

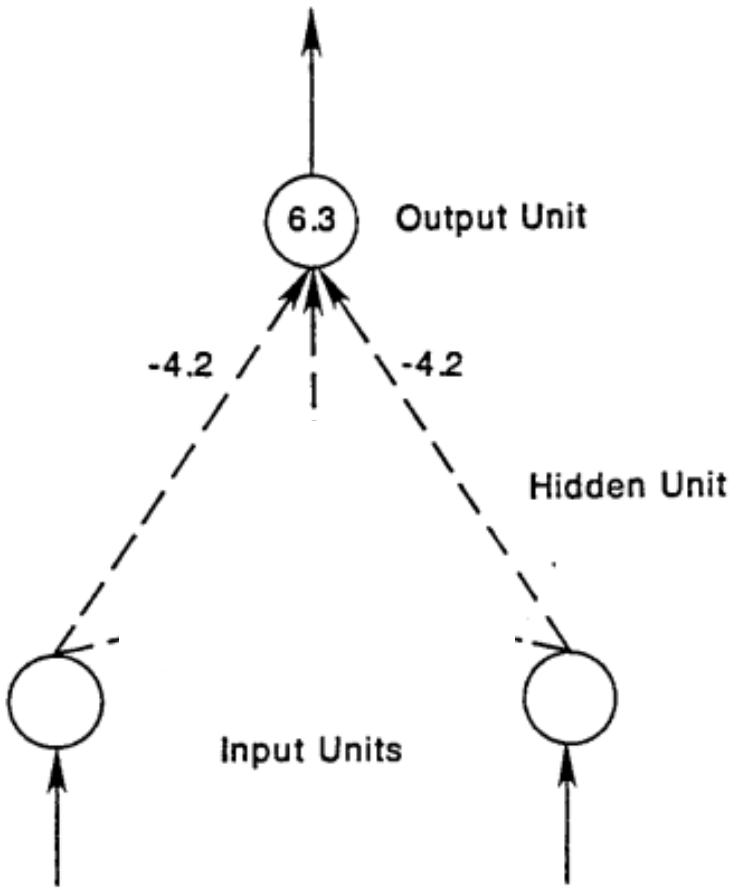
- Note here that the number in the unit is the *bias*: so think of this as what the unit "likes to do" in the absence of any input.



XOR

- Note here that the number in the unit is the *bias*: so think of this as what the unit "likes to do" in the absence of any input.

- What Boolean function is the network *without* the hidden unit computing?

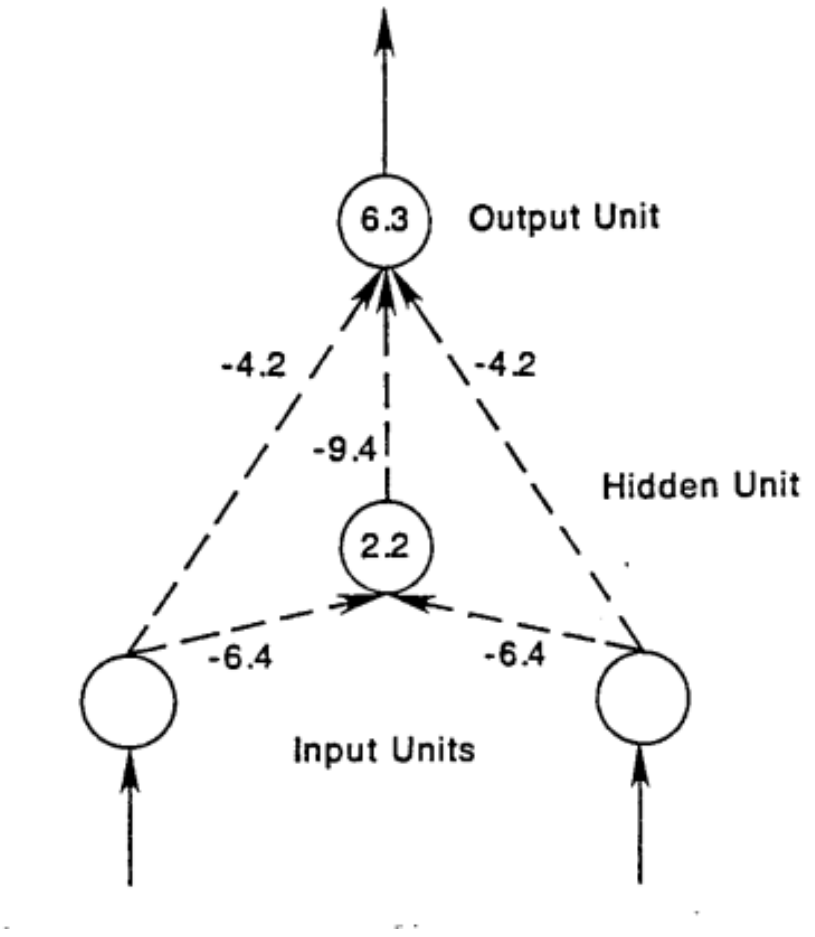


X1	X2	Y
0	0	1
0	1	1
1	0	1
1	1	0

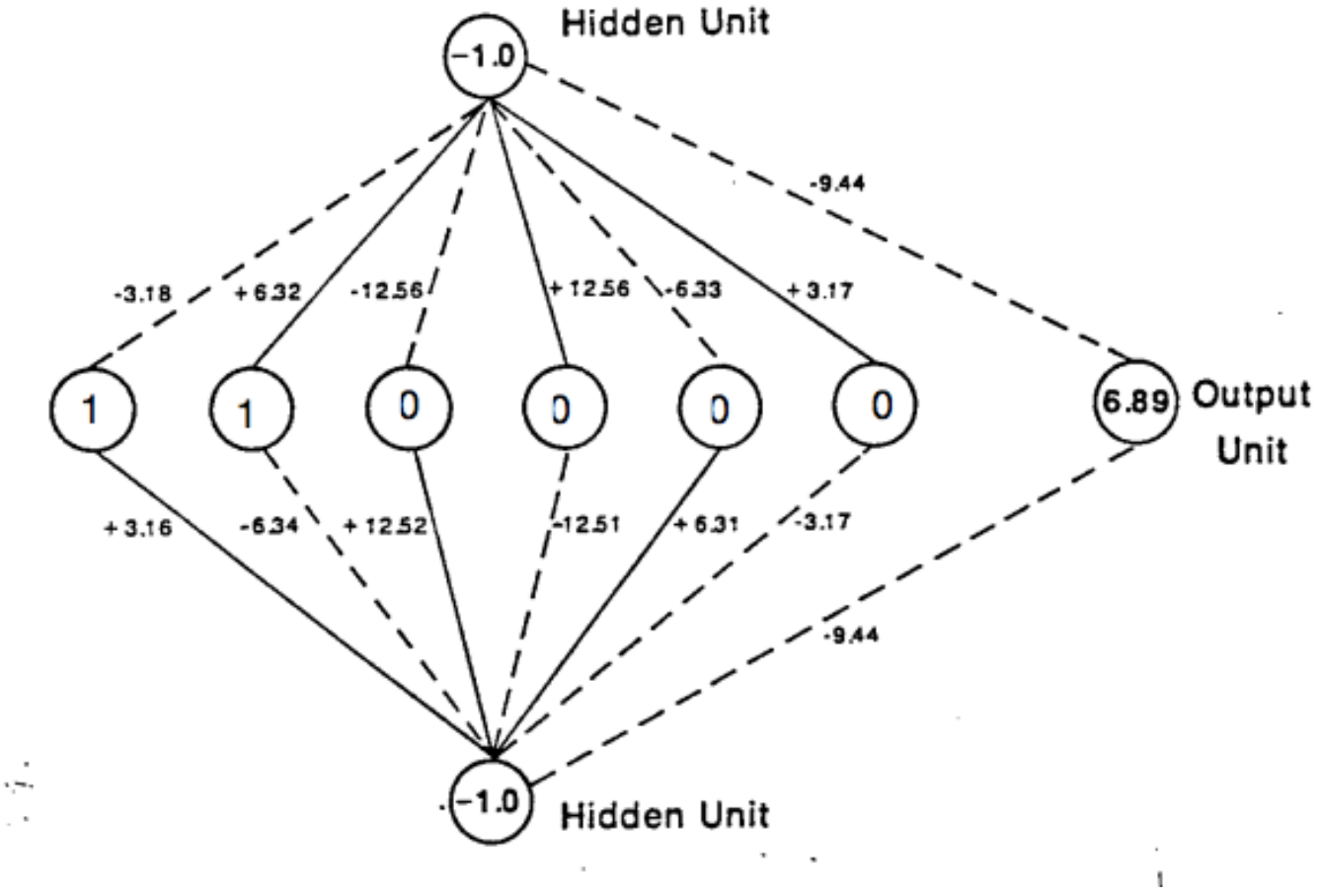
XOR

- Note here that the number in the unit is the *bias*: so think of this as what the unit "likes to do" in the absence of any input.
- What function is the hidden unit computing?

- It only turns on for (0,0)!

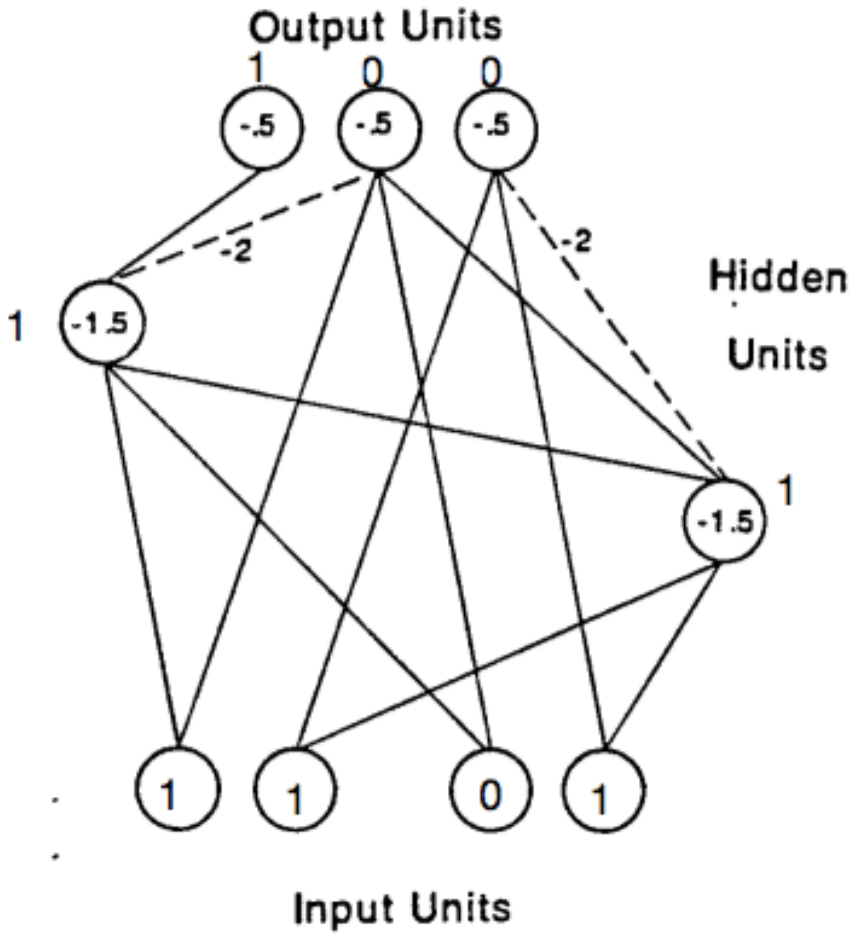


Symmetry


Hidden Unit
-1.0
-9.44
-3.18
+6.32
-12.56
+12.56
-6.33
+3.17
1
1
0
0
0
0
6.89
Output Unit
+3.16
-6.34
+12.52
-12.51
+6.31
-3.17
-9.44
-1.0
Hidden Unit

Addition

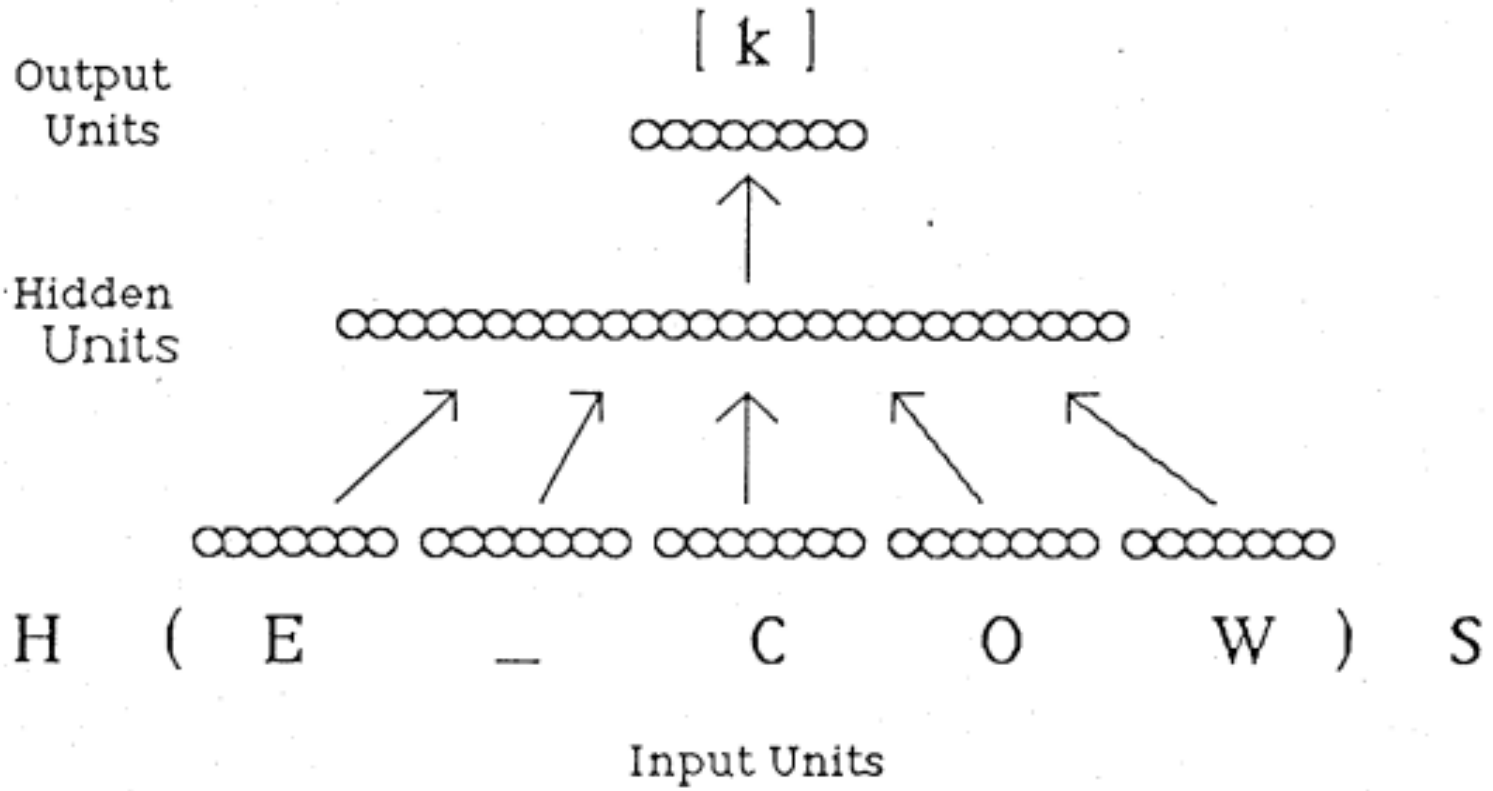
- Here – think of these as binary threshold units.
- Solid lines are =1
- What is the hidden unit on the right doing?
- The one on the left?



NetTalk
(Sejnowski & Rosenberg, 1987)

- A network that learns to read aloud from text
- Trained on a corpus of 500 words, overnight on a Vax
- Business Week: It learns the abilities of a six-year old child overnight! (a lot of hype…)
- Maps time into *space* – the text is stepped through the input (a lot of alignment issues)

NetTalk Architecture



Training corpus

You mean uh um like England or something. When we walk home from school I walk home with two friends and sometimes we cant run home from school though. Because um one girl where every time she wants to runs she gets the wheezes and stuff. And then she cant breathe very well and she gets sick. Thats why we cant run.

I like to go to my grandmothers house. Well because she gives us candy. Well um we eat there sometimes. Sometimes we sleep over night there.

Sometime when I go to go to my cousins I get to play soft ball or play badminton and all that.

Thing I hate to play is doctor. Oh. I hate to play doctor or house or that. Dont like it or stuff.

Weve been learning a lot of Spanish words. Our teacher speaks Spanish sometimes.

So does my father.

Well my father doesnt know very much Spanish but he doesnt know what gray is in Spanish and its gris and he doesnt and he knows what blue is in Spanish and he knows what um red is. In Spanish. And sometimes I like to go to Mexico but Ive never been there before. Only when I was a little teeny baby I been there and I dont even remember it.

There this one night I couldnt get any food. I mean there was this one day I couldnt get any food at home unless I asked it for Spanish.

My um my mother and father is going to pretty soon take us to Philadelphia. And were going to see our grandmother there.

I wish we went to uh we went to Mexico. Not Mexico San Diego once and they had a little um pool that was full of water and it was two feet. And then they and then they had another pool it was five feet eight feet. Randy my brother went in eight feet and I went in five feet and I think there was a three feet. There was. And I jumped off and I uh and I jumped off the edge

Test set

dont go in spring or winter because its too cold. My my brother can go swimming in the winter though because he gots his tonsils out you know and he and he gets sick uh sick um once in a few years. I get sick just about every day.

Theres just one thing I cant stand in my family. My baby makes too much noise I cant even get get to sleep for a minute. He wont stop jumping around in the bathtub.

In the bathtub.

No. In the crib. He he keeps jumping around gets tired then he goes to bed then he finally gets to sleep. Cant go to sleep in about a hour. Not with that in the house.

It would just take two minutes to get to sleep. Just about two minutes. If you just um why dont you get some cotton and plug it in your ears and then you cant hear him.

He makes so much noise he makes so much noise it probably sound effect through it.

Well what does the baby do. Come out get out crawl out of his crib and then come along in your bed and pull out your earplugs.

Once once he keep jump jumping jumping and then this thing slide down and then he fell over to the other bed and he start crying and I couldnt get to bed so I I have to wake up put him back in my crib.

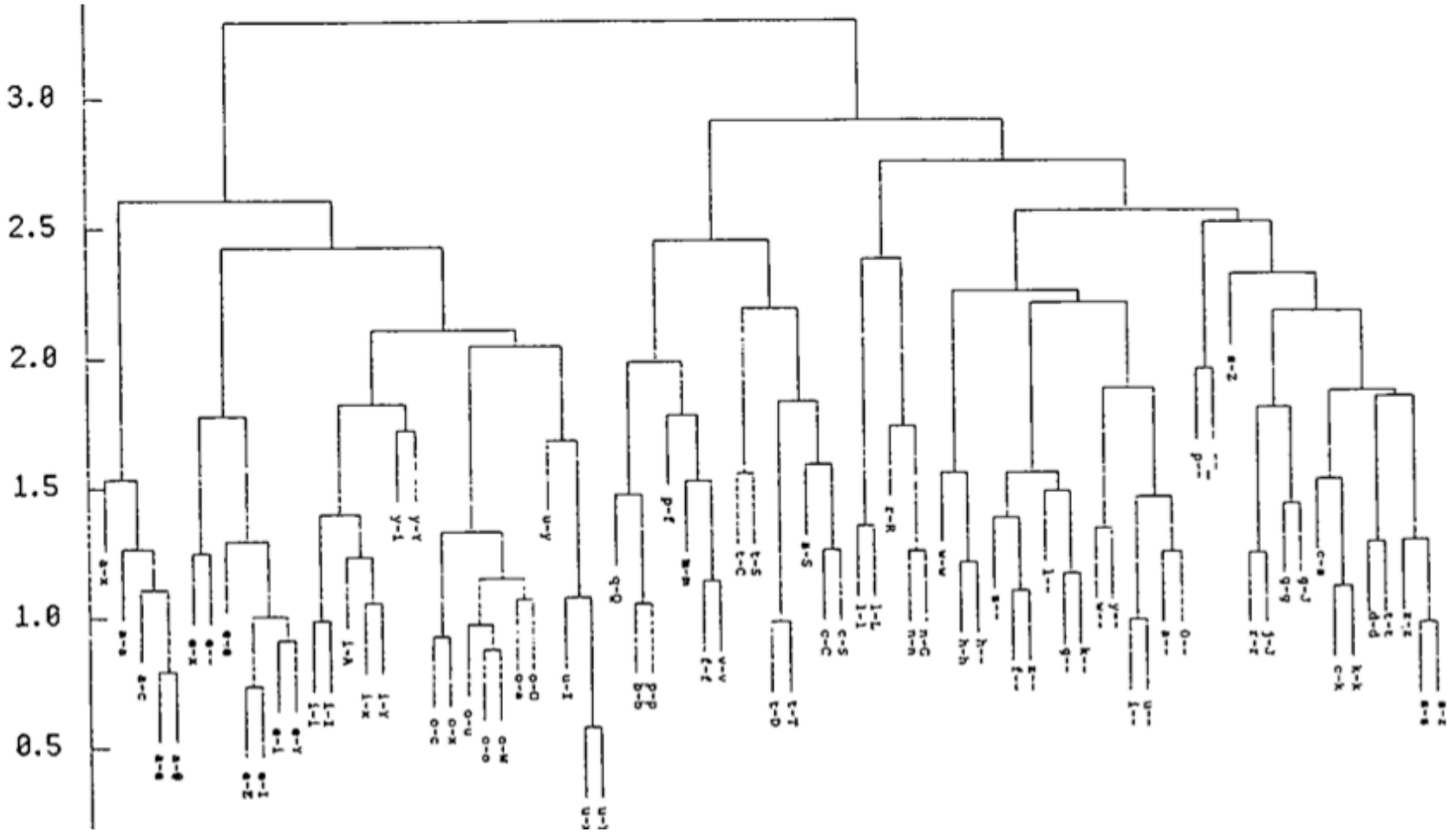
In your crib.

No not in my crib. I dont have a crib.

You said put him back in your crib.

I mean in his crib. I dont have a crib.

NetTalk hidden unit analysis



Hinton's Family Trees example

- Idea: Learn to represent relationships between people that are encoded in a family tree:

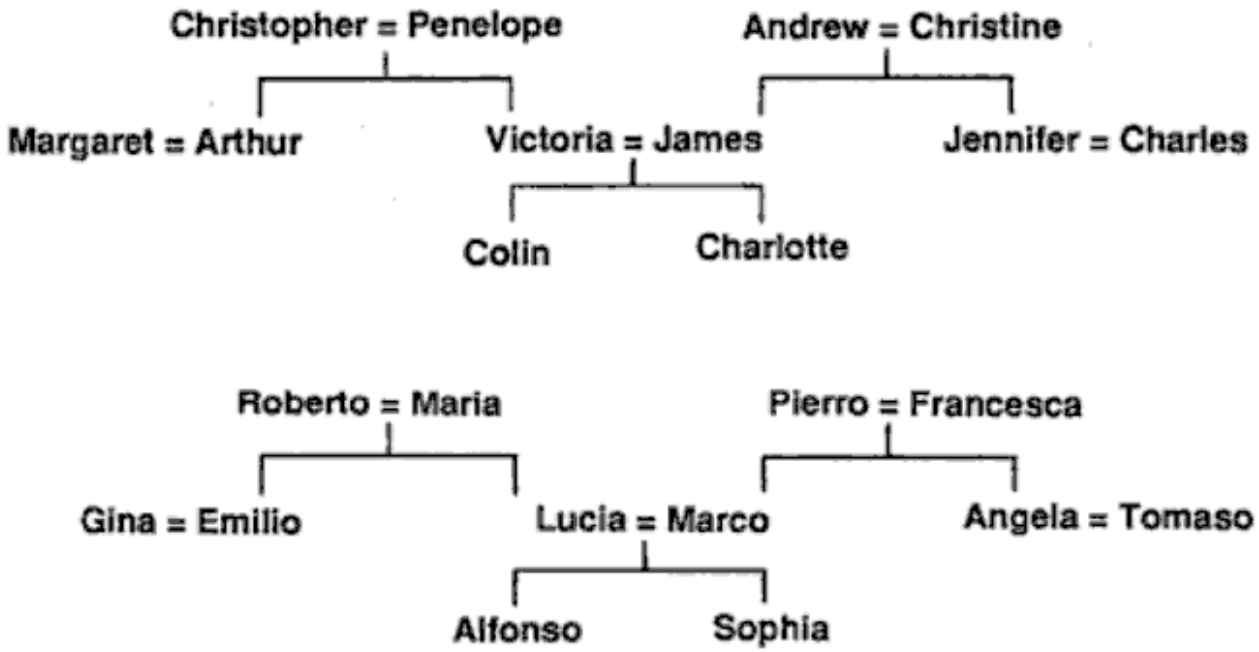
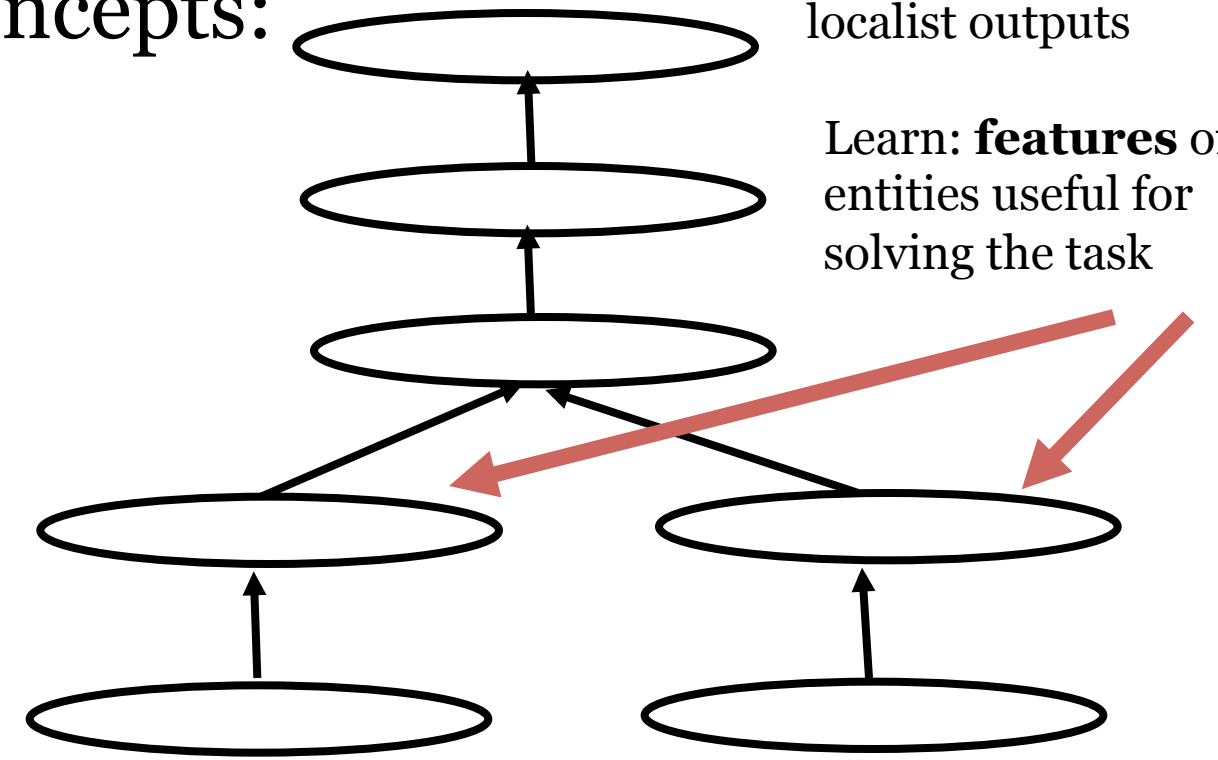


Figure 2: Two isomorphic family trees. The symbol "=" means "married to".

Hinton's Family Trees example

- Idea 2: Learn *distributed* representations of concepts:



Input: localist people localist relations

Localist: one unit "ON" to represent each item (AKA "one hot" encoding)

People hidden units: Hinton diagram

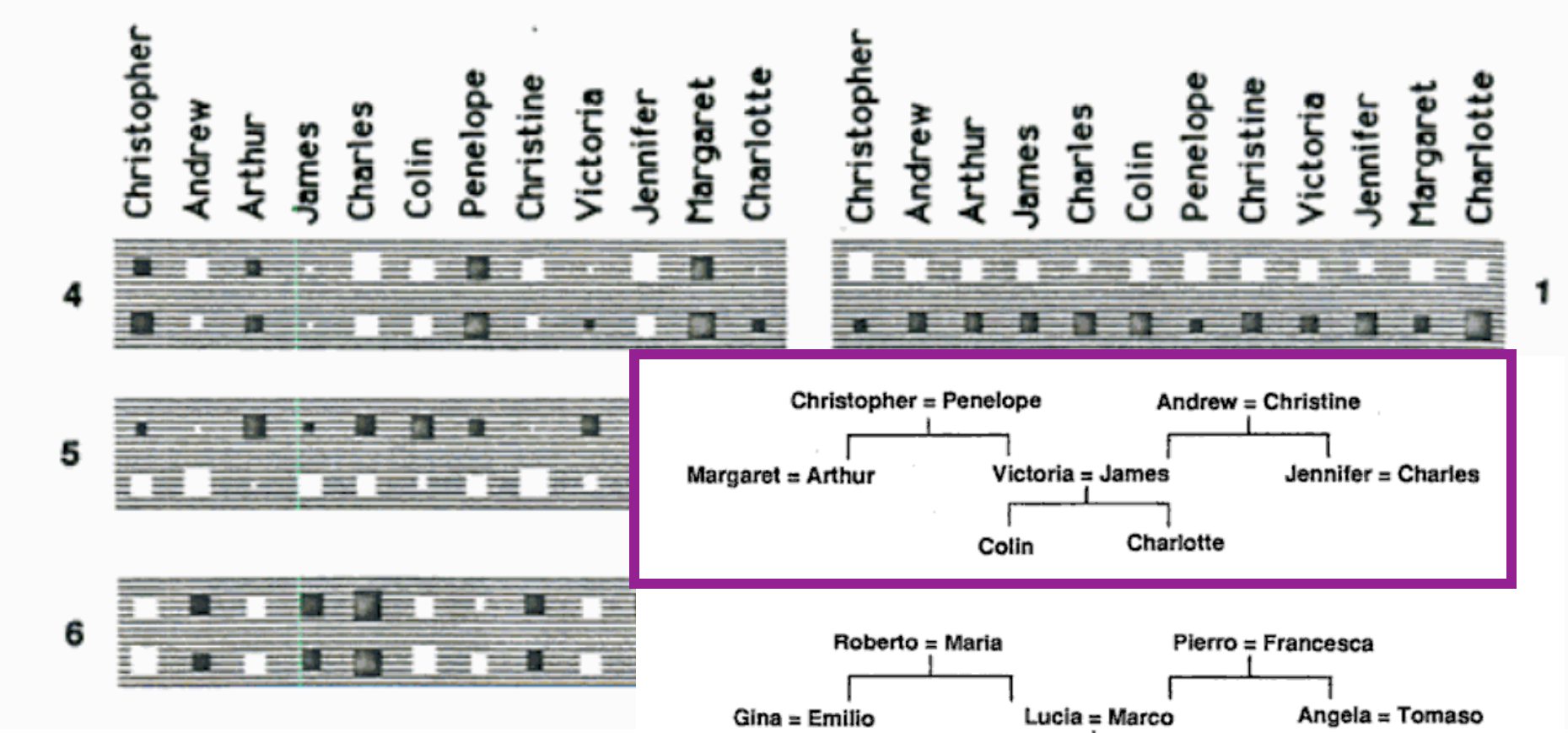
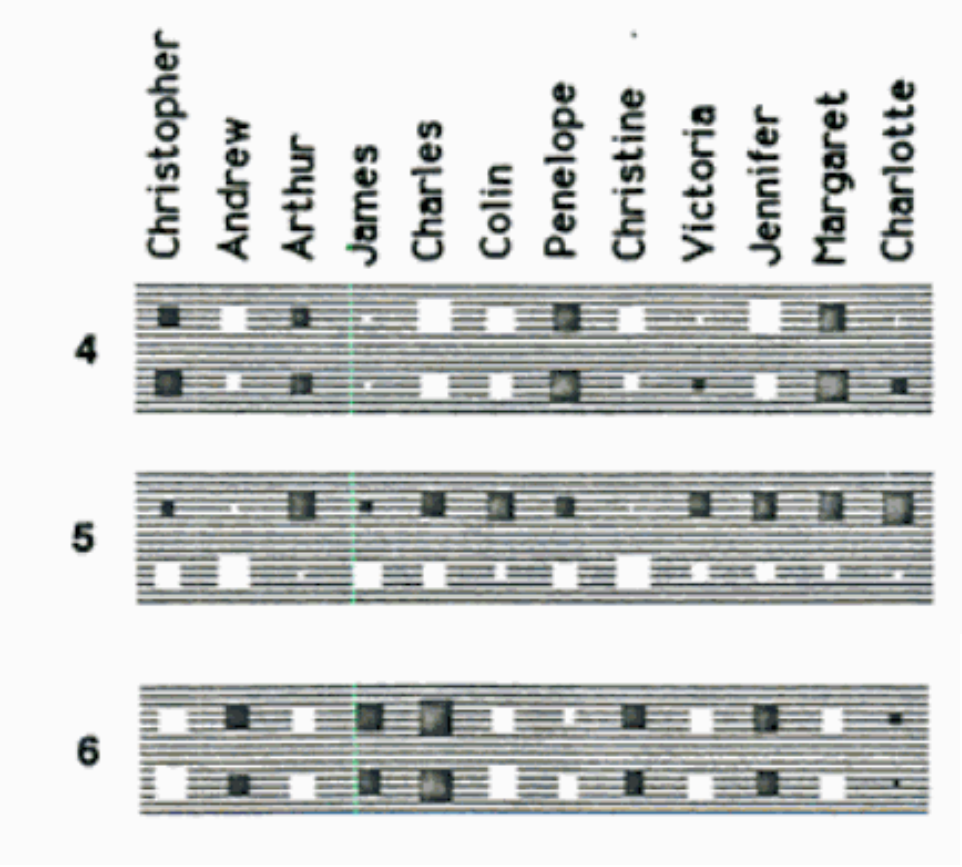


Figure 2: Two isomorphic family trees. The symbol "=" means "married to".

What is unit 1 encoding?

People hidden units: Hinton diagram



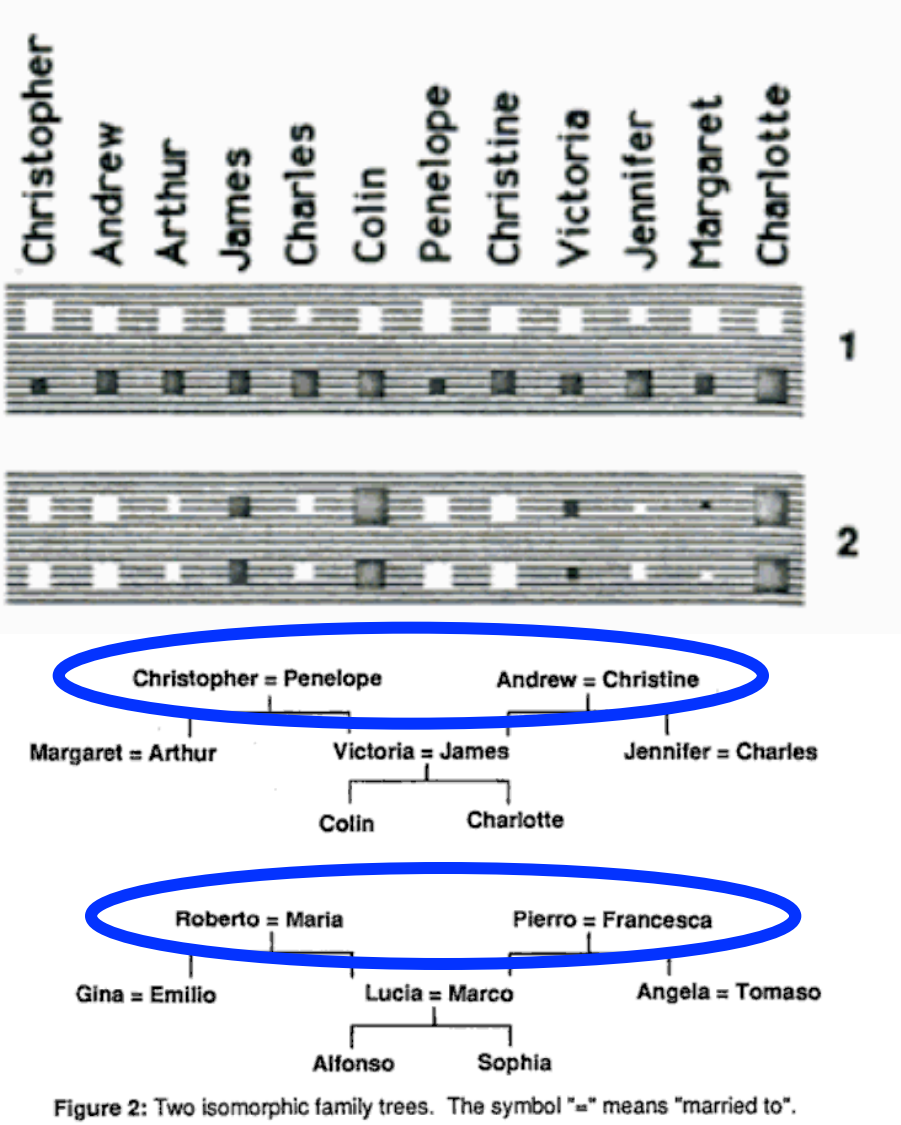
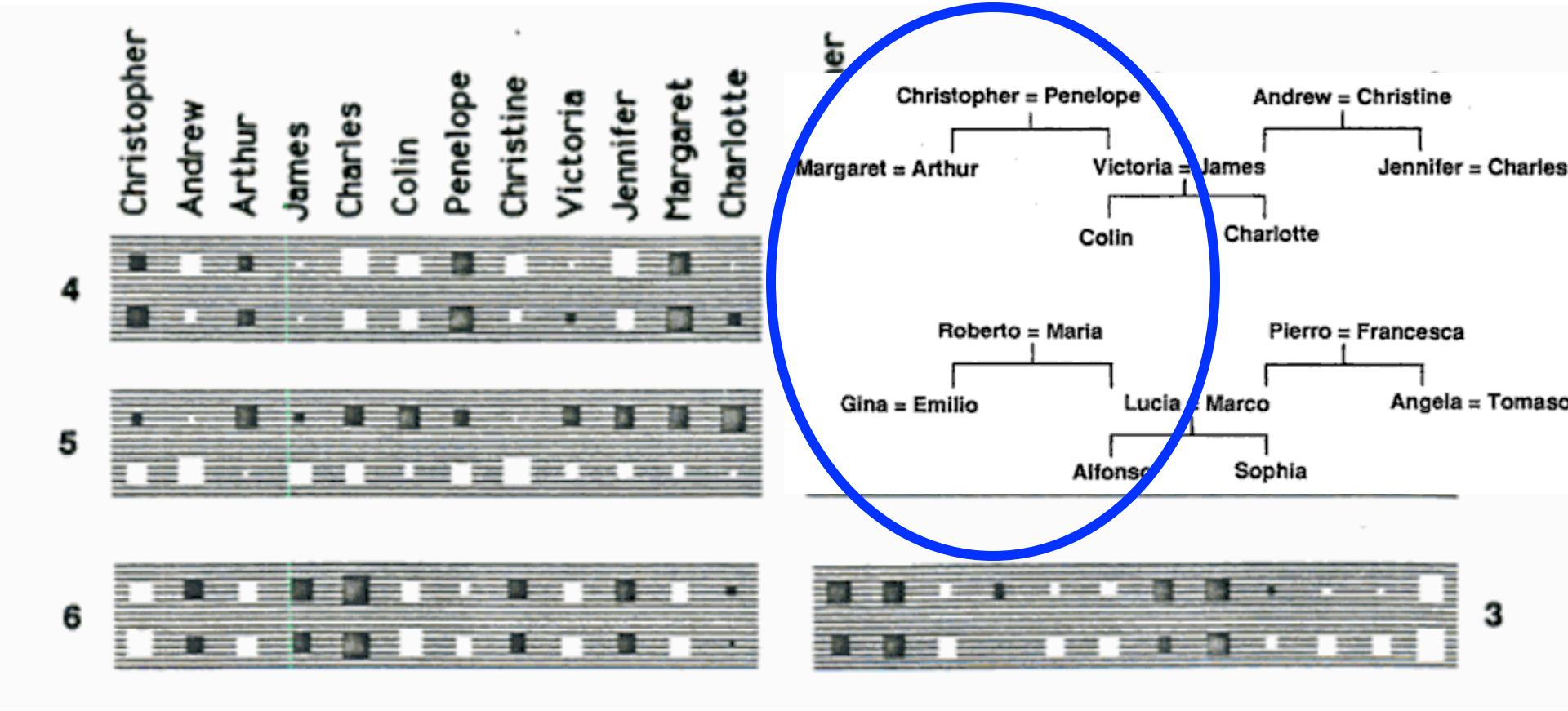


Figure 2: Two isomorphic family trees. The symbol "=" means "married to".

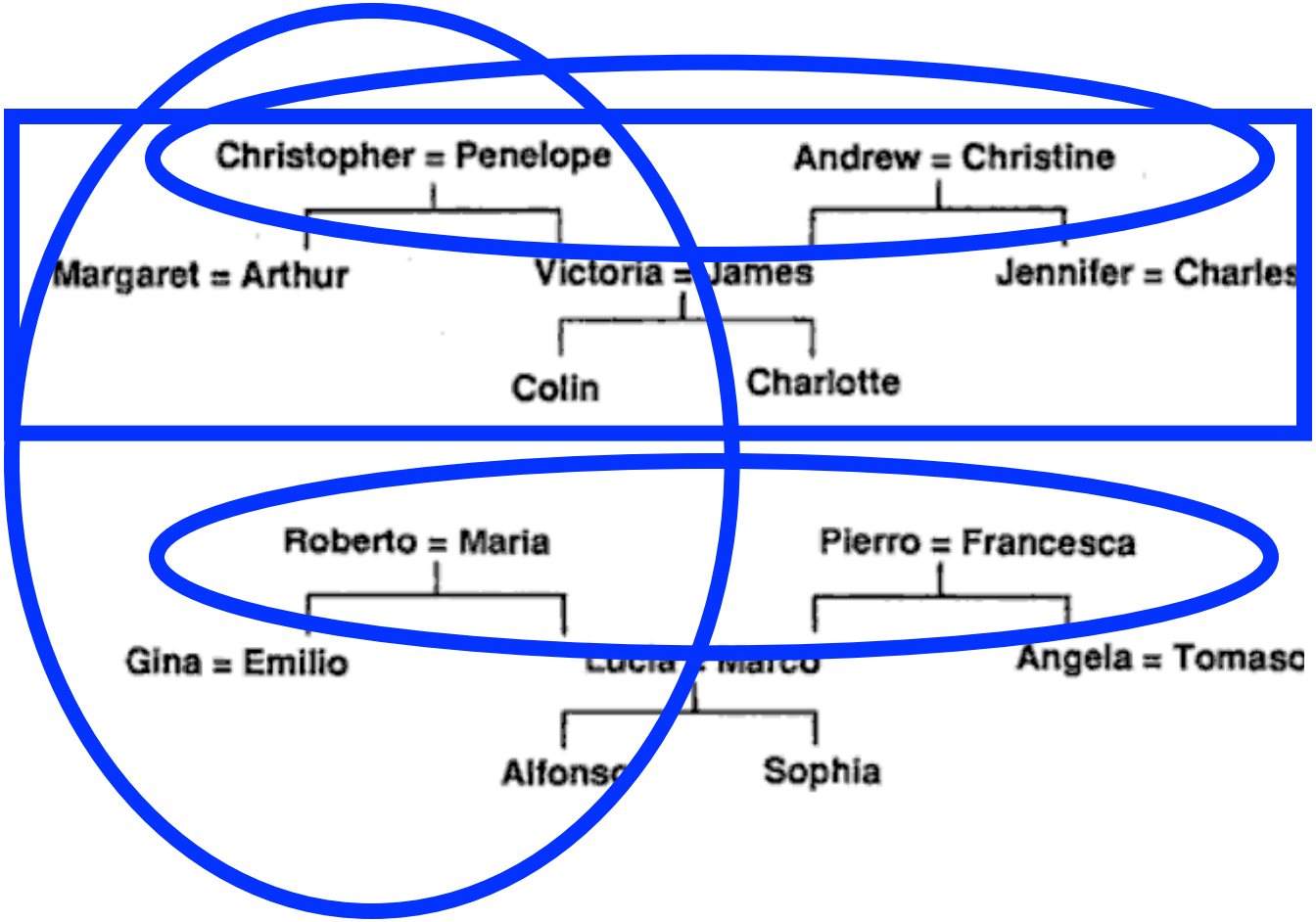
What is unit 2 encoding?

People hidden units: Hinton diagram



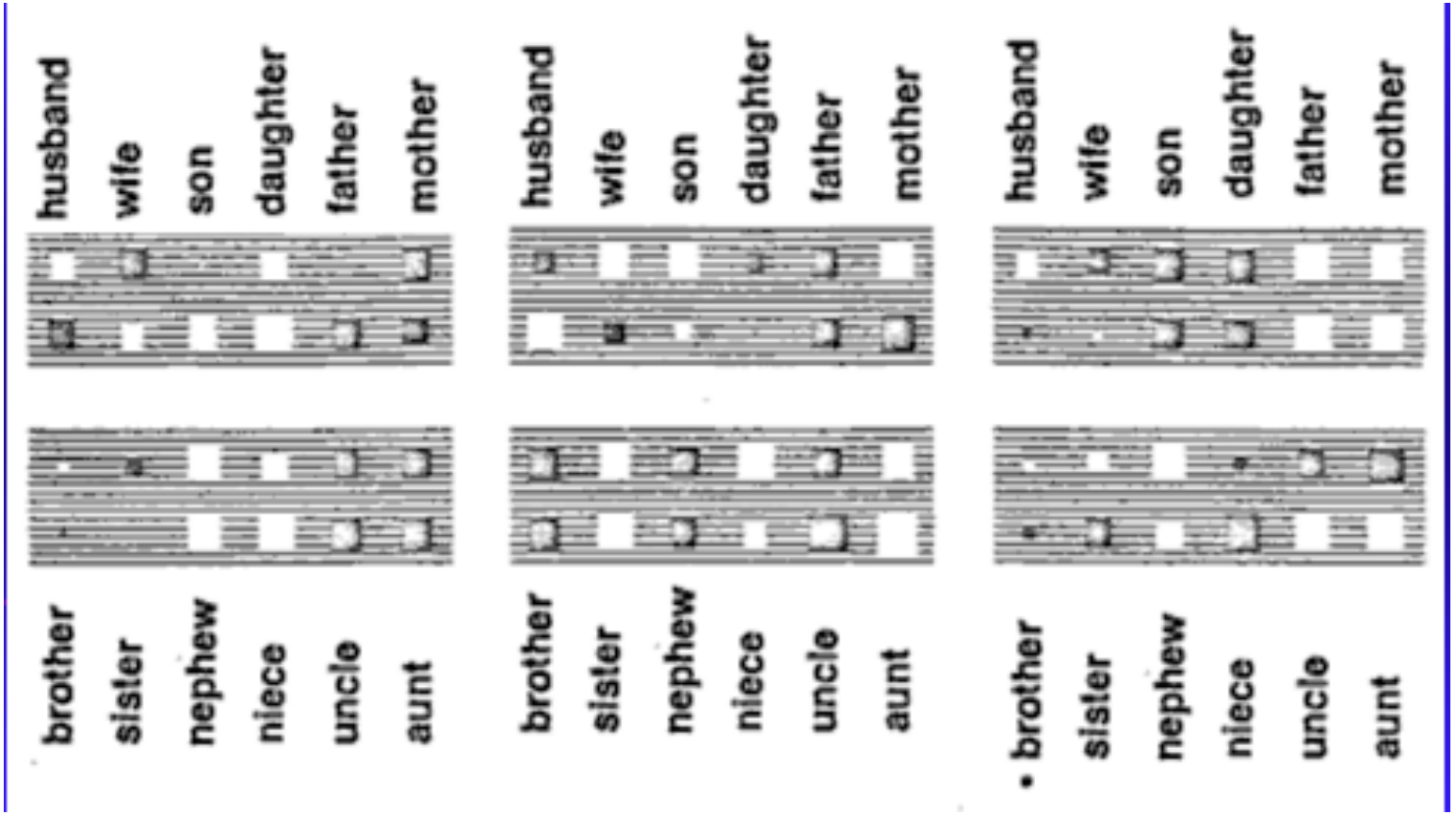
What is unit 6 encoding?

People hidden units: Hinton diagram



When all three are on, these units pick out Christopher and Penelope: Other combinations pick out other parts of the trees

Relation units



What does the lower middle one code?

Switch to Demo

- This demo is downloadable from my website under “Resources”
- About the middle of the page:
- “A matlab neural net demo with face processing.”

Lessons

- The network learns features *in the service of the task* - i.e., it learns features on its own.

- This is useful if we don't know what the features ought to be.

- Can explain some human phenomena