Backprop, 25 years later…

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

But first…

Hal White passed away March 31st, 2012

- Hal was "our theoretician of neural nets," and one of the nicest guys I knew.
- His paper on "A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity" has been cited 15,805 times, and led to him being shortlisted for the Nobel Prize.
- But his paper with Max Stinchcombe: "Multilayer feedforward networks are universal approximators" is his second most-cited paper, at 8,114 cites.

But first…

- In yet another paper (in *Neural Computation,* 1989), he wrote

"The premise of this article is that learning procedures used to train artificial neural networks are inherently statistical techniques. It follows that statistical theory can provide considerable insight into the properties, advantages, and disadvantages of different network learning methods…"

This was one of the first papers to make the connection between neural networks and statistical models - and thereby put them on a sound statistical foundation.

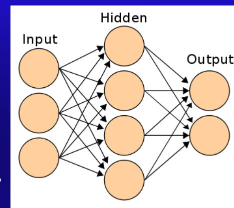
We should also remember…

Dave E. Rumelhart passed away on March 13, 2011

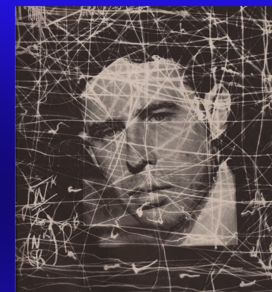
- Many had invented back propagation; few could appreciate as deeply as Dave did what they had when they discovered it.

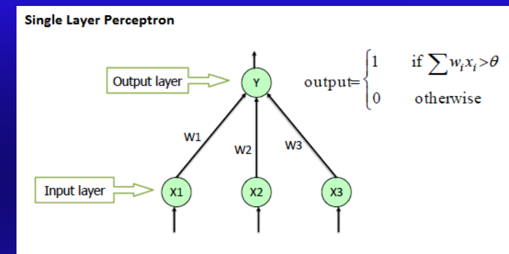
What is backpropagation, and why is/was it important?

- We have billions and billions of neurons that somehow work together to create the mind.
- These neurons are connected by 10^{14} - 10^{15} synapses, which we think encode the "knowledge" in the network - too many for us to explicitly program them in our models
- Rather we need some way to ***indirectly*** set them via a procedure that will achieve some goal by changing the synaptic strengths (which we call weights).
- This is called ***learning*** in these systems.



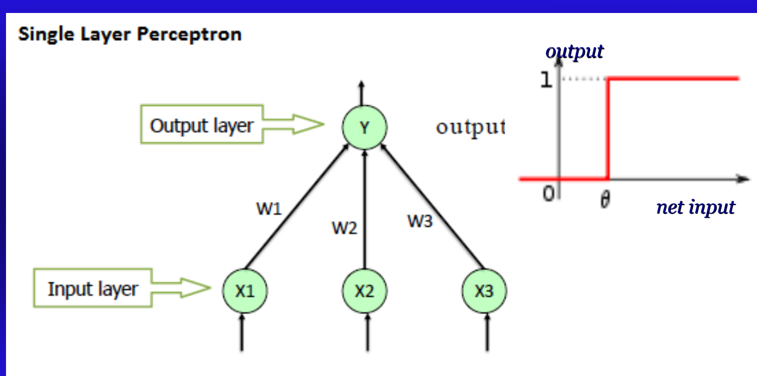
Learning: A bit of history



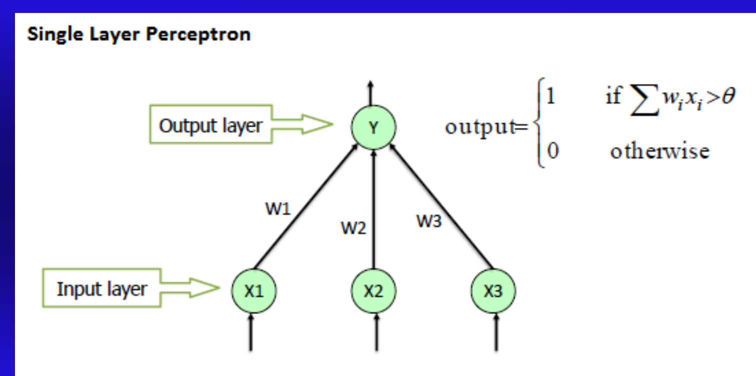


- Frank Rosenblatt studied a simple version of a neural net called a *perceptron:*
  - A single layer of processing
  - Binary output
  - Can compute simple things like (some) boolean functions (OR, AND, etc.)

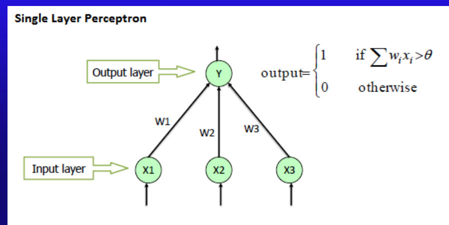
Learning: A bit of history



Learning: A bit of history



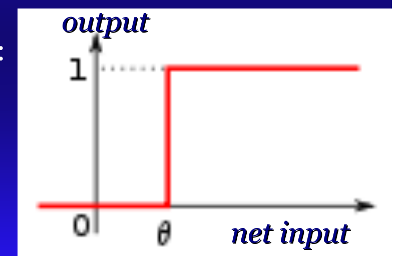
Learning: A bit of history



- Rosenblatt (1962) discovered a learning rule for perceptrons called the *perceptron convergence procedure*.
- Guaranteed to learn anything computable (by a two-layer perceptron)
- Unfortunately, not everything was computable (Minsky & Papert, 1969)

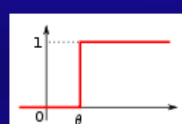
Perceptron Learning Demonstration

- Output activation rule:
  - First, compute the *net input* to the output unit:
    $\sum w_i x_i = net$
  - Then, compute the output as:
    If $net \geq \theta$ then output = 1
    else output = 0



Perceptron Learning Demonstration

- Output activation rule:
  - First, compute the *net input* to the output unit:
    $\sum w_i x_i = net$
    If $net \geq \theta$ then output = 1
    else output = 0



- Learning rule:
  If output is 1 and should be 0, then *lower* weights to active inputs and *raise* the threshold (θ)
  If output is 0 and should be 1, then *raise* weights to active inputs and *lower* the threshold (θ)

  ("active input" means $x_i = 1$, not 0)

- STOP HERE FOR DEMO

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 Made Simple

- Output activation rule:
  - First, compute the *net input* to the output unit:
    $\sum w_i x_i = net$
    If $net \geq \theta$ then output = 1
    else output = 0
- Learning rule:
  If output is 1 and should be 0, then *lower* weights to active inputs and *raise* the threshold (θ)
  If output is 0 and should be 1, then *raise* weights to active inputs and *lower* the threshold (θ)

Perceptron Learning Made Simple

- Learning rule:
  If output is 1 and should be 0, then *lower* weights to active inputs and *raise* the threshold (θ)
  If output is 0 and should be 1, then *raise* weights to active inputs and *lower* the threshold (θ)
- Learning rule:

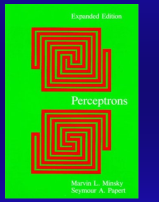
$$w_i(t+1) = w_i(t) + \eta*(\text{teacher - output})*x_i$$

(η 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)

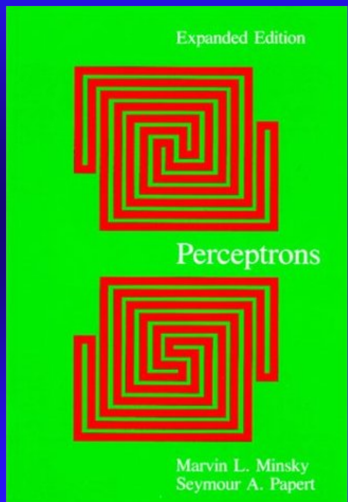
Problems with perceptrons

- 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



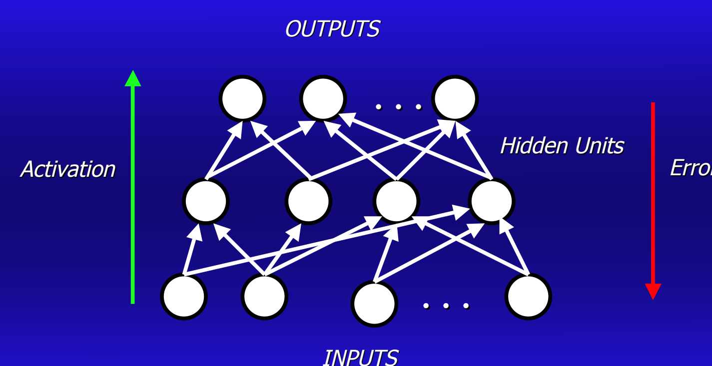
Aside about perceptrons

- 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)

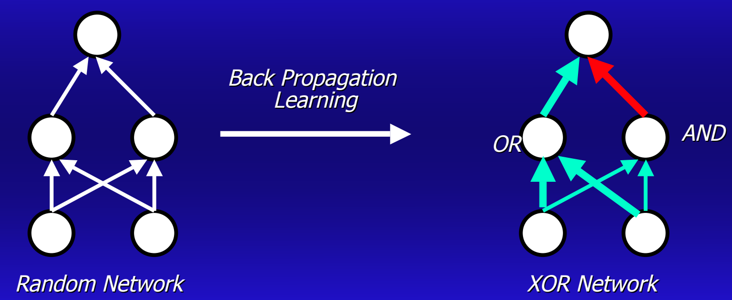
- Discovered a learning rule for networks with hidden units.
- Works a lot like the perceptron algorithm:
  - Randomly choose an input-output pattern
  - present the input, let activation propagate through the network
  - give the *teaching signal*
  - propagate the error back through the network (hence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

Enter Rumelhart, Hinton, & Williams (1985)



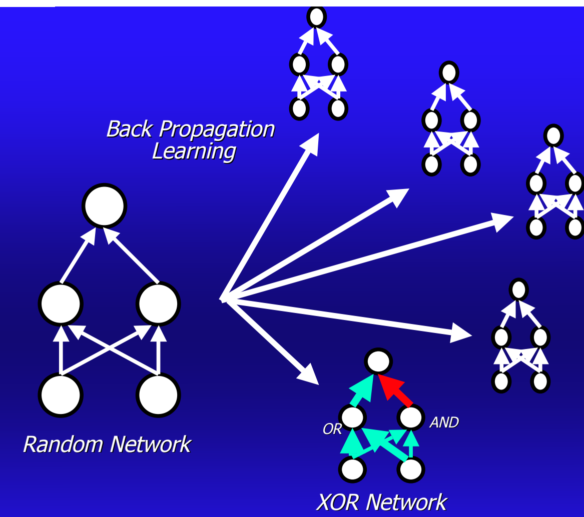
- The actual algorithm uses the chain rule of calculus to go *downhill* in an error measure with respect to the weights
- The hidden units must learn features that solve the problem

XOR



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

XOR



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

Why is/was this wonderful?

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

Hinton's Family Trees example

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

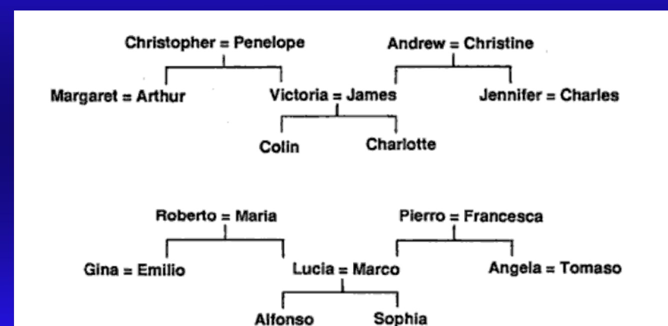
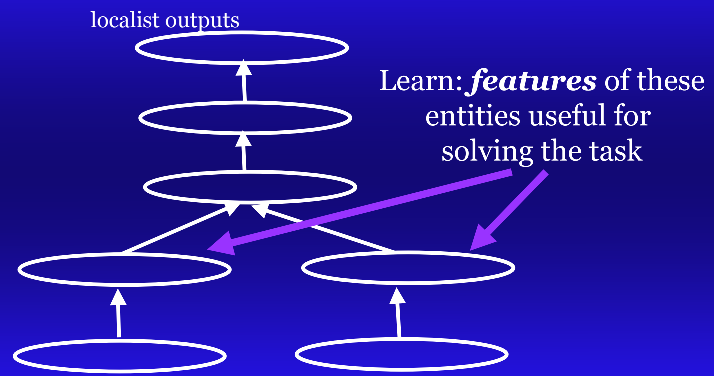


Figure 2: Two isomorphic family trees. The symbol "=" means "married to".

Hinton's Family Trees example

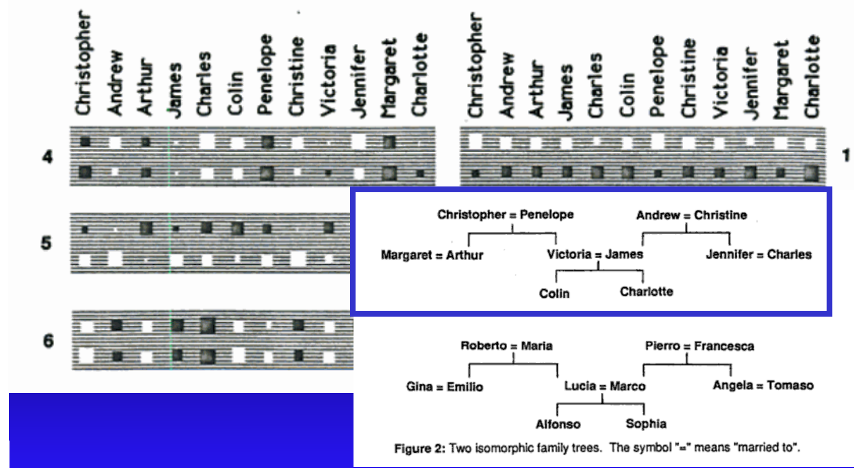
- Idea 2: Learn *distributed* representations of concepts:



Input: localist people   localist relations

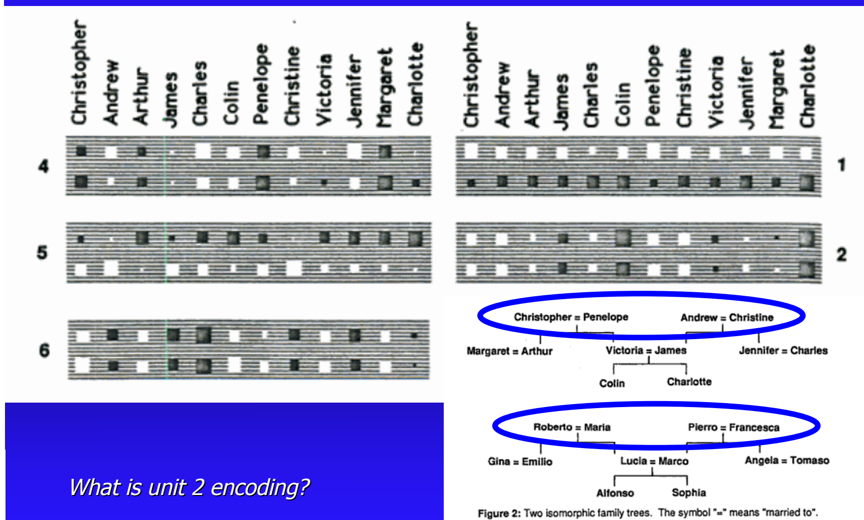
Localist: one unit "ON" to represent each item

People hidden units: Hinton diagram



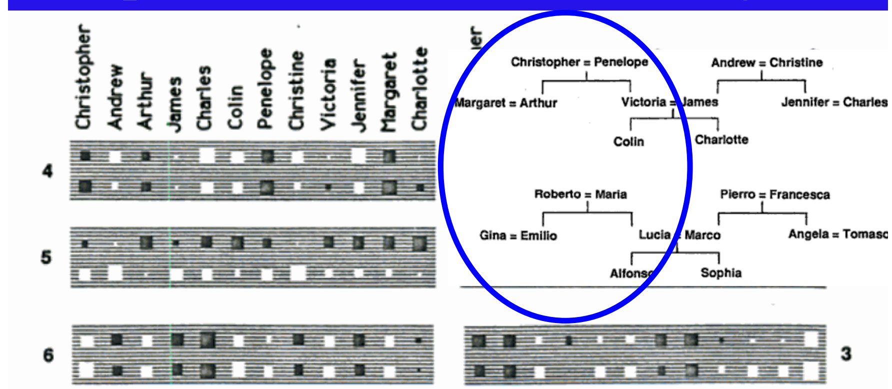
What is unit 1 encoding?

People hidden units: Hinton diagram



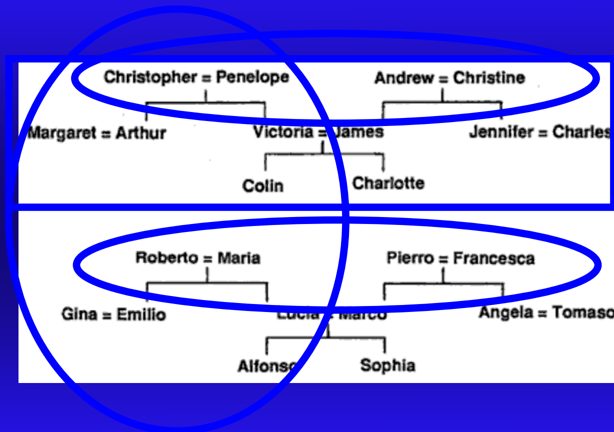
What is unit 2 encoding?

People hidden units: Hinton diagram



What is unit 6 encoding?

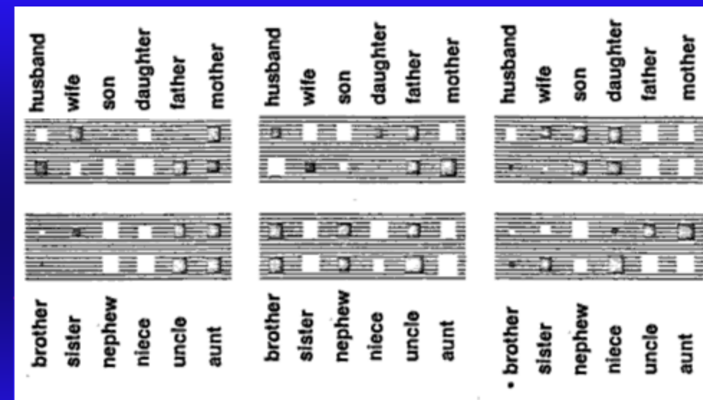
People hidden units: Hinton diagram



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?

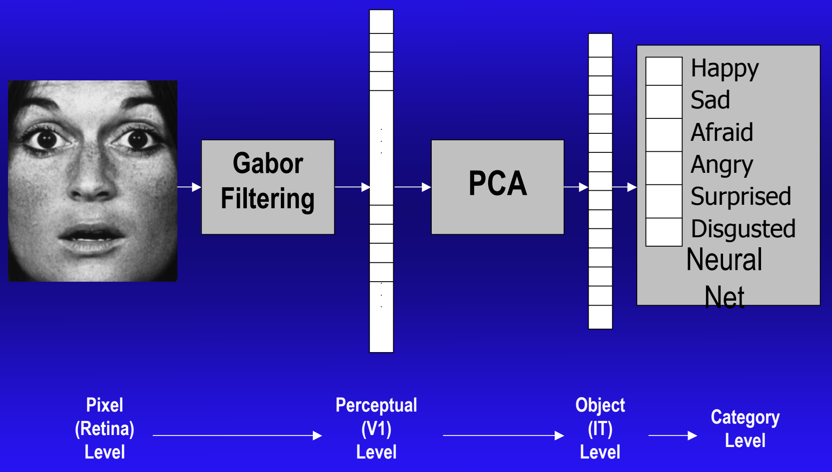
Lessons

- 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

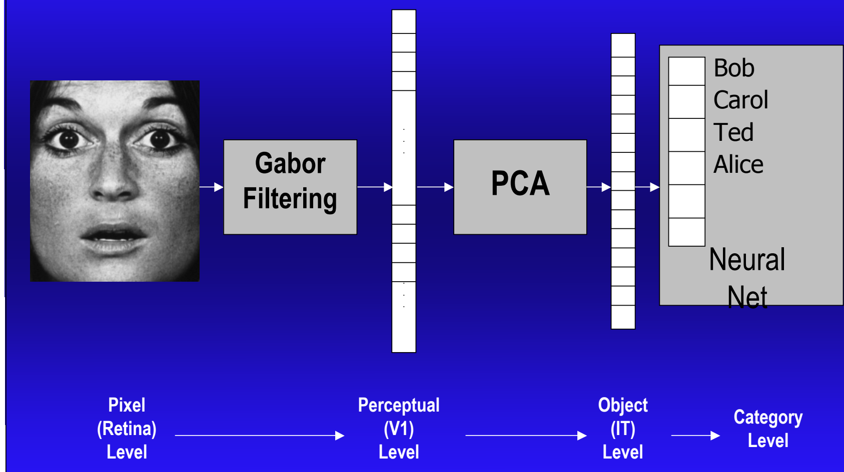
Another example

- In the next example(s), I make two points:
  - The perceptron algorithm is still useful!
  - Representations learned in the service of the task can explain the "Visual Expertise Mystery"

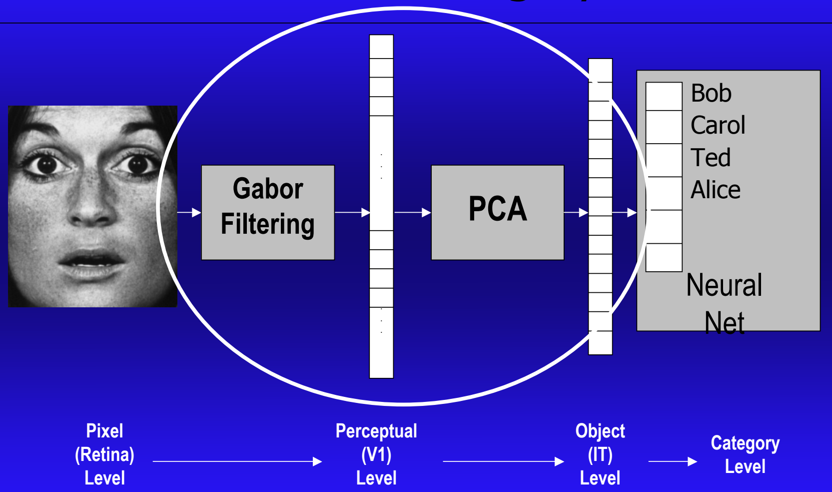
A Face Processing System



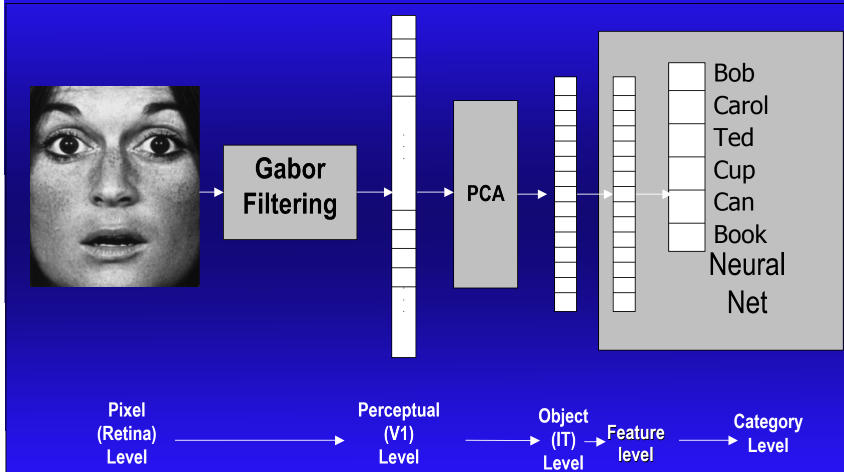
The Face Processing System



The Face Processing System

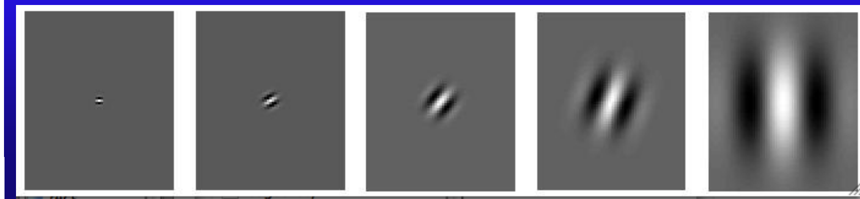


The Face Processing System

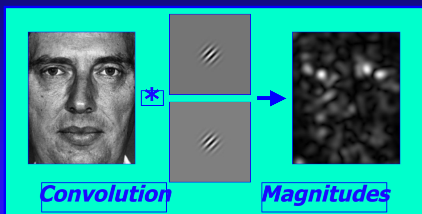


The Gabor Filter Layer

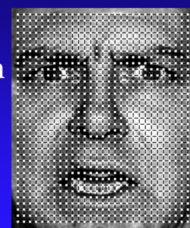
- Basic feature: the 2-D Gabor wavelet filter (Daugman, 85):



- These model the processing in early visual areas



Subsample in a 29x36 grid



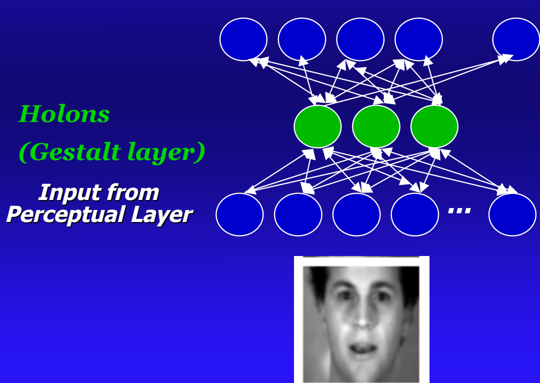
Principal Components Analysis

- The Gabor filters give us 40,600 numbers
- We use PCA to reduce this to 50 numbers
- PCA is like Factor Analysis: It finds the underlying directions of Maximum Variance
- PCA can be computed in a neural network through a competitive Hebbian learning mechanism
- Hence this is also a *biologically plausible* processing step
- We suggest this leads to representations similar to those in Inferior Temporal cortex

How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

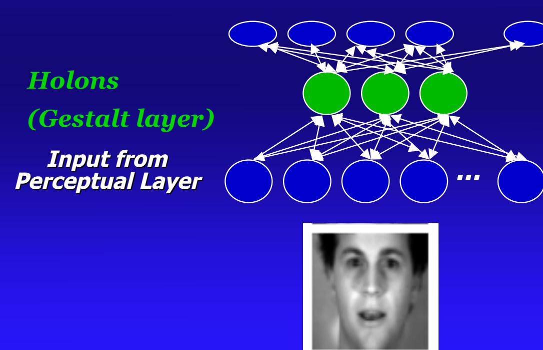
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

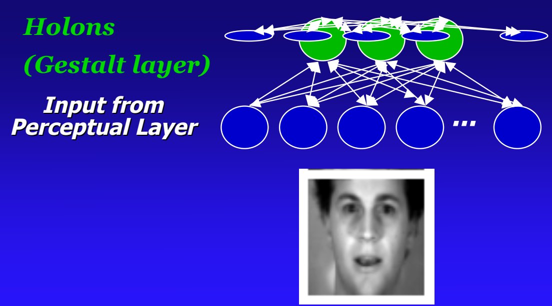
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

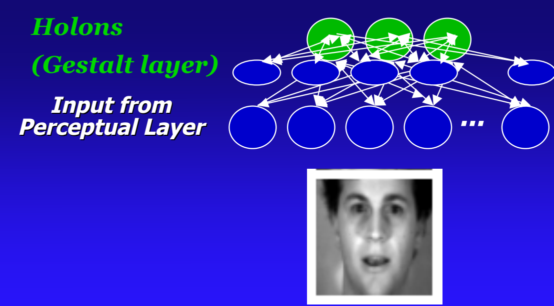
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

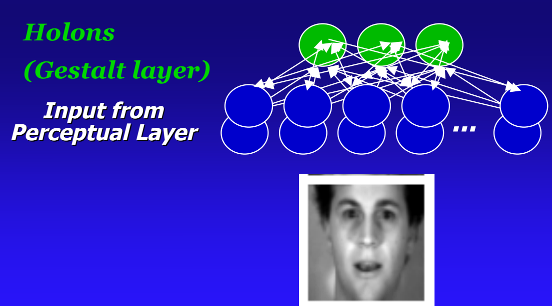
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

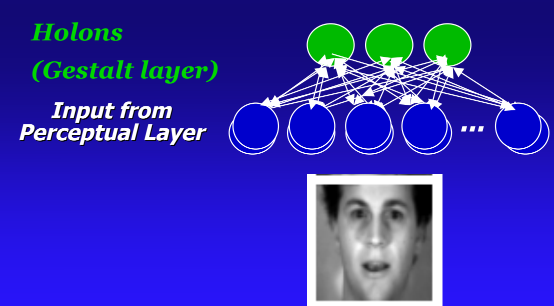
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

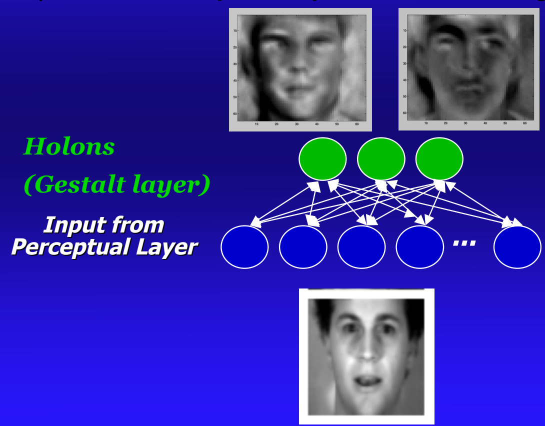
- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



How to do PCA with a neural network

(Cottrell, Munro & Zipser, 1987; Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; O'Toole et al. 1991)

- **A self-organizing network that learns whole-object representations**
  (features, Principal Components, Holons, eigenfaces)



Holons

- They act like face cells (Desimone, 1991):
  - Response of single units is strong despite occluding eyes, e.g.
  - Response drops off with rotation
  - Some fire to my dog's face
- A novel representation: ***Distributed templates*** --
  - each unit's optimal stimulus is a ghostly looking face (template-like),
  - but many units participate in the representation of a single face (distributed).
  - For this audience: Neither exemplars nor prototypes!
- Explain holistic processing:
  - ***Why? If stimulated with a partial match, the firing represents votes for this template:***

    Units "downstream" don't know what caused this unit to fire.

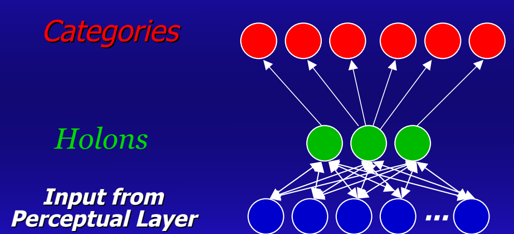
    (more on this later…)

The Final Layer: Classification

(Cottrell & Fleming 1990; Cottrell & Metcalfe 1990; Padgett & Cottrell 1996; Dailey & Cottrell, 1999; Dailey et al. 2002)

The holistic representation is then used as input to a categorization network trained by supervised learning.

Output: Cup, Can, Book, Greeble, Face, Bob, Carol, Ted, Happy, Sad, Afraid, etc.



- Excellent generalization performance demonstrates the sufficiency of the holistic representation for recognition

The Final Layer: Classification

- Categories can be at different levels: basic, subordinate.
- Simple learning rule (~delta rule). It says (mild lie here):
  - ***add*** inputs to your weights (synaptic strengths) when you are supposed to be ***on***,
  - ***subtract*** them when you are supposed to be ***off***.
- This makes your weights "look like" your favorite patterns – the ones that turn you on.
- When no hidden units => No back propagation of error.
- When hidden units: we get task-specific features (most interesting when we use the basic/subordinate distinction)

Facial Expression Database

- Ekman and Friesen quantified muscle movements (Facial Actions) involved in prototypical portrayals of happiness, sadness, fear, anger, surprise, and disgust.
  - Result: the Pictures of Facial Affect Database (1976).
  - 70% agreement on emotional content by naive human subjects.
- 110 images, 14 subjects, 7 expressions.

Anger, Disgust, Neutral, Surprise, Happiness (twice), Fear, and Sadness
This is actor "JJ": The easiest for humans (and our model) to classify

Results (Generalization)

Expression	Network % Correct	Human % Agreement
Happiness	100.0%	98.7%
Surprise	100.0%	92.4%
Disgust	100.0%	92.3%
Anger	89.2%	88.9%
Sadness	82.9%	89.2%
Fear	66.7%	87.7%
Average	89.9%	91.6%

- Kendall's *tau* (rank order correlation): .667, p=.0441
- Note: This is an *emergent property* of the model!

Correlation of Net/Human Errors

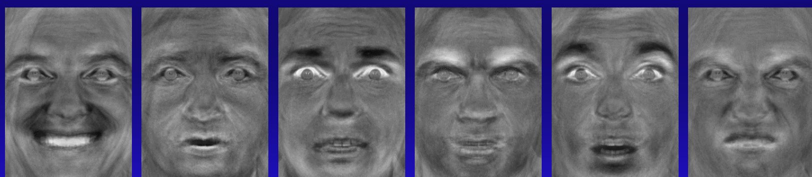
- Like all good Cognitive Scientists, we like our models to make the same mistakes people do!
- Networks and humans have a 6x6 confusion matrix for the stimulus set.
- This suggests looking at the off-diagonal terms: The errors
- Correlation of off-diagonal terms: r = 0.567. [$F_{(1,28)}$ = 13.3; p = 0.0011]
- Again, this correlation is an *emergent property* of the model: It was not told which expressions were confusing.

Examining the Net's Representations

- We want to visualize "receptive fields" in the network.
- But the Gabor magnitude representation is noninvertible.
- We can *learn* an approximate inverse mapping, however.
- We used linear regression to find the best linear combination of Gabor magnitude principal components for each image pixel.
- Then projecting each unit's *weight vector* into image space with the same mapping visualizes its "receptive field."

Examining the Net's Representations

- The "y-intercept" coefficient for each pixel is simply the average pixel value at that location over all faces, so subtracting the resulting "average face" shows more precisely what the units attend to:



- Apparently local features appear in the global templates.

Morph Transition Perception

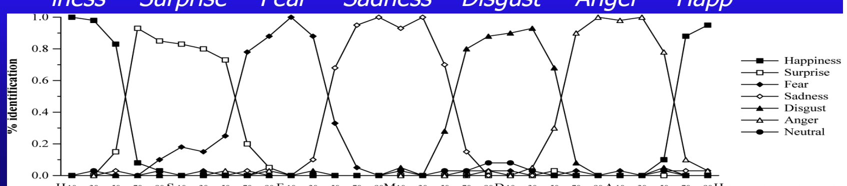
- Morphs help psychologists study categorization behavior in humans
- Example: JJ Fear to Sadness morph:

0% 10% 30% 50% 70% 90% 100%

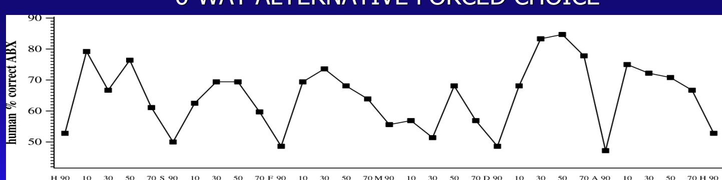
- Young et al. (1997) Megamix: presented images from morphs of all 6 emotions (15 sequences) to subjects in random order, task is 6-way forced choice button push.

Results: classical Categorical Perception: sharp boundaries…

iness Surprise Fear Sadness Disgust Anger Happ-



6-WAY ALTERNATIVE FORCED CHOICE



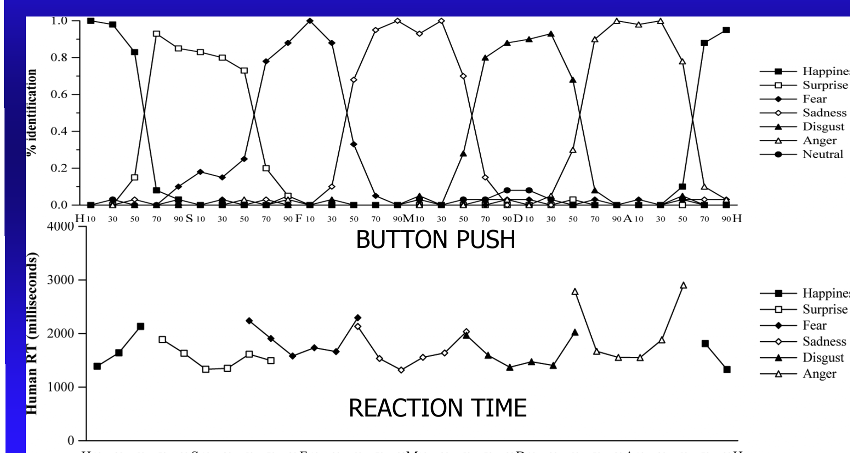
PERCENT CORRECT DISCRIMINATION

…and higher discrimination of pairs of images when they cross a perceived category boundary

Results: Non-categorical RT's

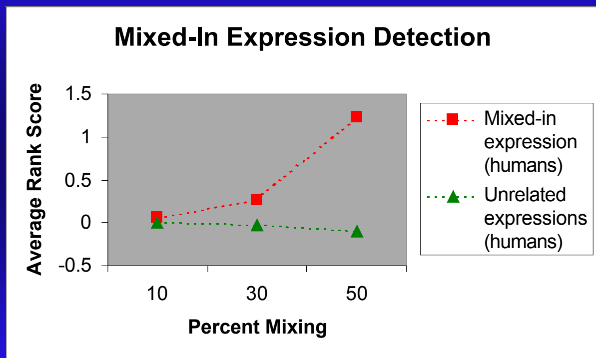
- "Scalloped" Reaction Times

iness Surprise Fear Sadness Disgust Anger Happ-



Results: More non-categorical effects

- Young et al. Also had subjects rate 1st, 2nd, and 3rd most apparent emotion.

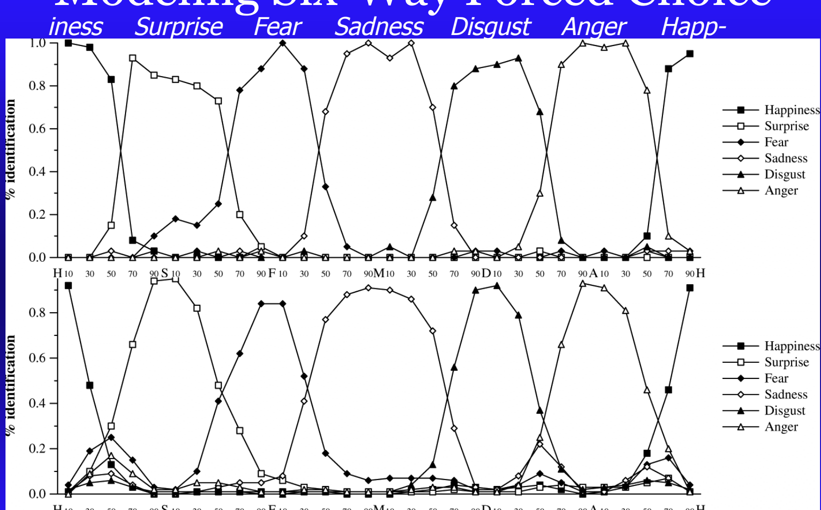


- At the 70/30 morph level, subjects were above chance at detecting mixed-in emotion. These data seem more consistent with *continuous* theories of emotion.

Modeling Megamix

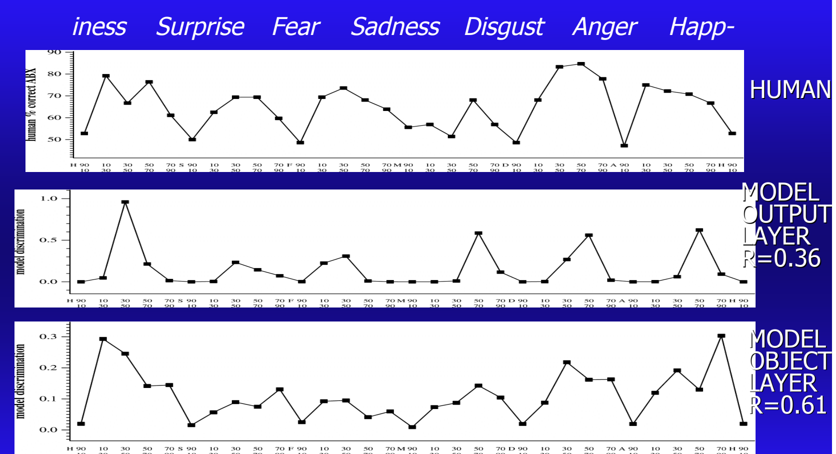
- 1 trained neural network = 1 human subject.
- 50 networks, 7 random examples of each expression for training, remainder for holdout.
- Identification = average of network outputs
- Response time = uncertainty of maximal output (1.0 - y_{max}).
- Mixed-in expression detection: record 1st, 2nd, 3rd largest outputs.
- Discrimination: 1 – correlation of layer representations
- We can then find the layer that best accounts for the data

Modeling Six-Way Forced Choice



- Overall correlation r=.9416, with NO FIT PARAMETERS!

Model Discrimination Scores



- The model fits the data best at a precategorical layer: The layer we call the “object” layer; NOT at the category level

Discrimination

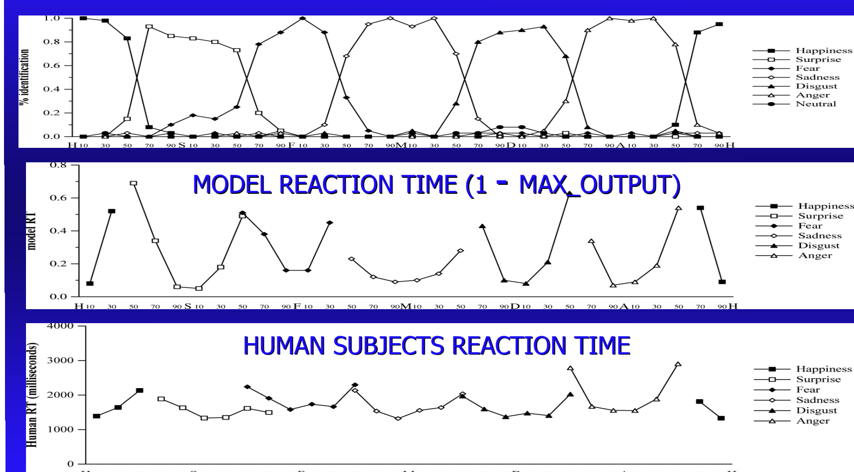
- Classically, one requirement for “categorical perception” is higher discrimination of two stimuli at a fixed distance apart when those two stimuli cross a category boundary
- Indeed, Young et al. found in two kinds of tests that discrimination was highest at category boundaries.
- The result that we fit the data best at a layer before any categorization occurs is significant: In some sense, the category boundaries are “in the data,” or at least, in our representation of the data.

Outline

- An overview of our facial expression recognition system.
- The internal representation shows the model’s prototypical representations of Fear, Sadness, etc.
- How our model accounts for the “categorical” data
- How our model accounts for the “non-categorical” data
- Discussion
- Conclusions for part 1

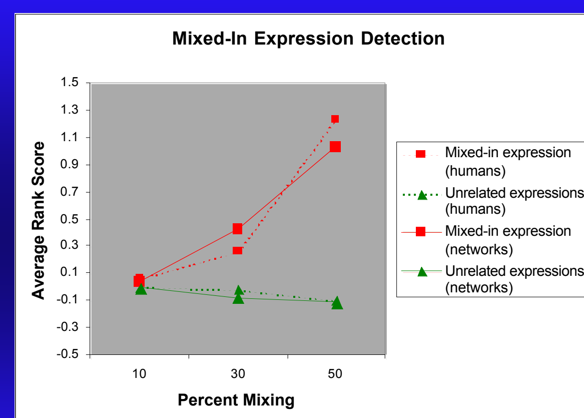
Reaction Time: Human/Model

iness Surprise Fear Sadness Disgust Anger Happ-



Correlation between model & data: .6771, p<.001

Mix Detection in the Model



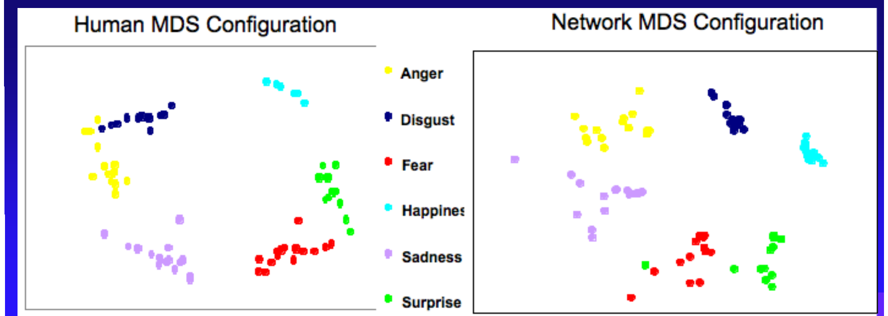
Can the network account for the continuous data as well as the categorical data? YES.

Human/Model Circumplexes

These are derived from similarities between images using non-metric Multi-dimensional scaling.

For humans: similarity is correlation between 6-way forced-choice button push.

For networks: similarity is correlation between 6-category output vectors.



Outline

- An overview of our facial expression recognition system.
- How our model accounts for the "categorical" data
- How our model accounts for the "two-dimensional" data
- The internal representation shows the model's prototypical representations of Fear, Sadness, etc.
- ➡ Discussion
- Conclusions for part 1

Discussion

- Our model of facial expression recognition:
  - Performs the same task people do
  - On the same stimuli
  - At about the same accuracy
- Without actually "feeling" anything, without any access to the surrounding culture, it nevertheless:
  - Organizes the faces in the same order around the circumplex
  - Correlates very highly with human responses.
  - Has about the same rank order difficulty in classifying the emotions

Discussion

- The discrimination correlates with human results most accurately at a *precategorization* layer: The discrimination improvement at category boundaries is in the representation of data, not based on the categories.
- These results suggest that for expression recognition, the notion of "categorical perception" simply is not necessary to explain the data.
- Indeed, most of the data can be explained by the interaction between the similarity of the representations and the categories imposed on the data: Fear faces are similar to surprise faces in our representation – so they are near each other in the circumplex.

Conclusions from this part of the talk

- The best models perform the same task people do
- Concepts such as "similarity" and "categorization" need to be understood in terms of models that do these tasks
- Our model simultaneously fits data supporting both categorical and continuous theories of emotion
- The fits, we believe, are due to the interaction of the way the categories slice up the space of facial expressions,
- and the way facial expressions inherently resemble one another.
- It also suggests that the continuous theories are correct: "discrete categories" are not required to explain the data.
- We believe our results will easily generalize to other visual tasks, and other modalities.

Backprop, 25 years later

- Backprop is important because it was the first relatively efficient method for learning internal representations
- Recent advances have made deeper networks possible
- This is important because we don't know how the brain uses transformations to recognize objects across a wide array of variations (e.g., the Halle Berry neuron)

END

- E.g., the "Halle Berry" neuron…