Neural Networks II

Professor Michael C. Mozer

CSCI 3202

model selection

What's my rule?

1 2 3	yes (fits rule)
4 5 6	yes
6 7 8	yes
9 2 41	no

Plausible rules

- three consecutive single digits
- three consecutive integers
- three numbers in ascending order
- three numbers whose sum is less than 25
- three numbers < 10
- 1, 4, or 6 in first column
- "yes" to first 3 sequences, "no" to all others

No single correct rule given the data

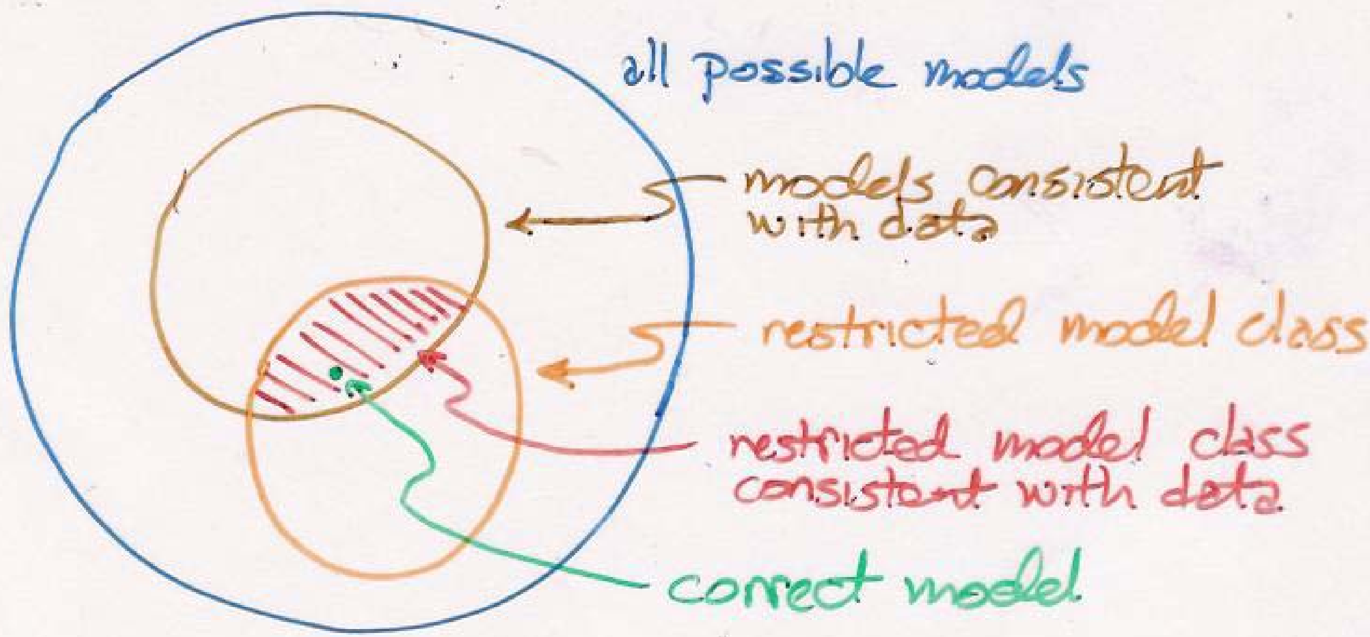
How do we select from among the many possible rules consistent with the data?

"What's my rule?" for machine learning

x_1	x_2	x_3	d
0	0	0	1
0	1	1	0
1	0	0	0
1	1	1	1
0	0	1	?
0	1	0	?
1	0	1	?
1	1	0	?

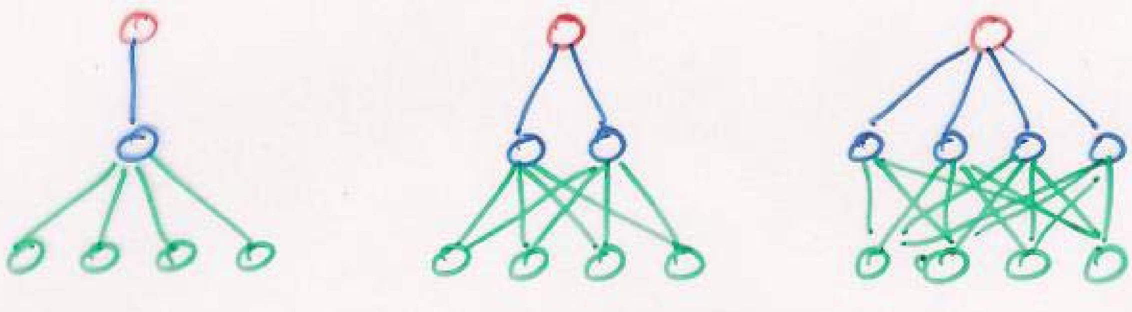
2^4 different rules (models)

with $\underline{a}$ binary inputs and p training examples, there are $2^{(2^a - p)}$ possible models



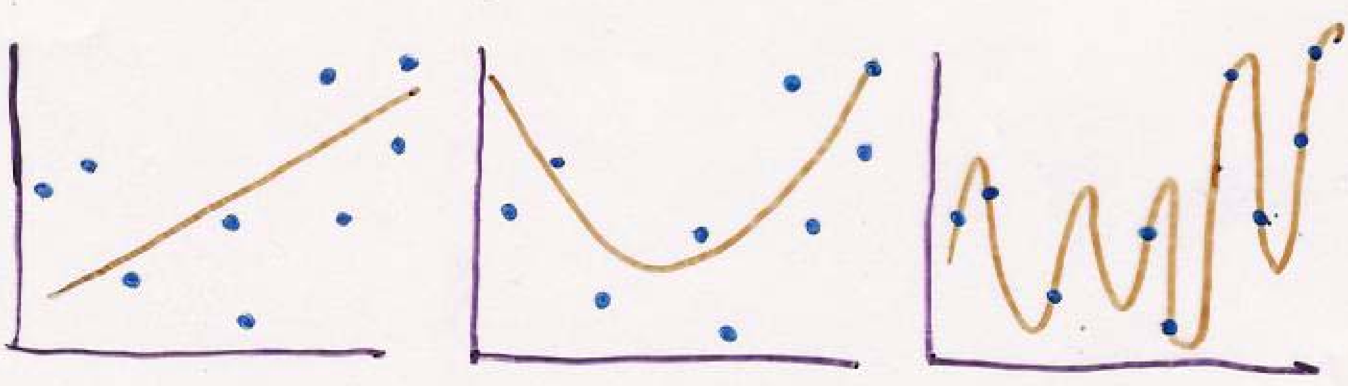
Challenge for machine learning approaches: Find restricted model class appropriate for problem at hand a.k.a. model selection

Model complexity continuum for neural nets

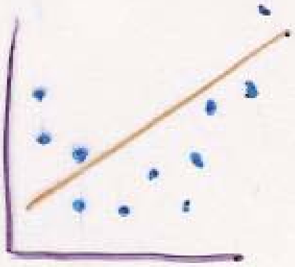
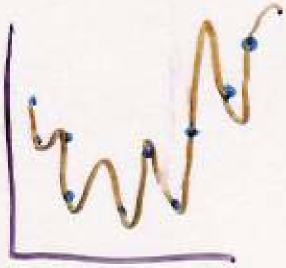


simple ⟷ complex

Model complexity continuum for polynomial regression



simple ⟷ complex

Simple model may be too inflexible to capture structure in data HIGH BIAS

Complex model may be so flexible that it captures noise in data HIGH VARIANCE

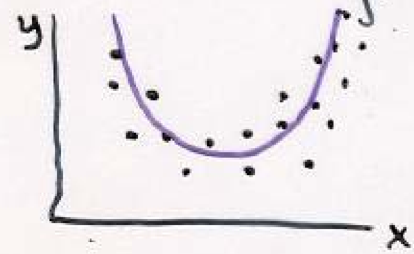
Key questions:

* Is variation in data due to structure/ regularity or noise?
* What degree of complexity do we need to capture structure without also picking up noise?

Statistical perspective

parametric statistical inference — curve fitting, e.g., linear or quadratic model

estimator → $y = K_1 + K_2 x + K_3 x^2$



nonparametric statistical inference — no parametric model, i.e., makes no assumptions as to form of function

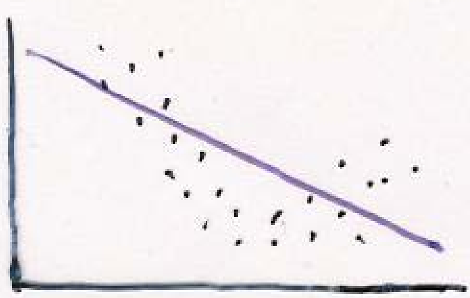
For a training set T of fixed size N and a particular input X and target output d (assume scalar), $(f_T(x) - d)^2$ tells us performance of net

How does this error vary as a function of T?

$$\underbrace{E_T\left[(f_T(X) - d)^2\right]}_{\text{expected error over ensemble of possible } T} = \underbrace{\left(E_T[f_T(X)] - d\right)^2}_{\text{bias}} + \underbrace{E_T\left[\left(f_T(X) - E_T[f_T(X)]\right)^2\right]}_{\text{variance}}$$

bias ~ degree to which model (network) is constrained

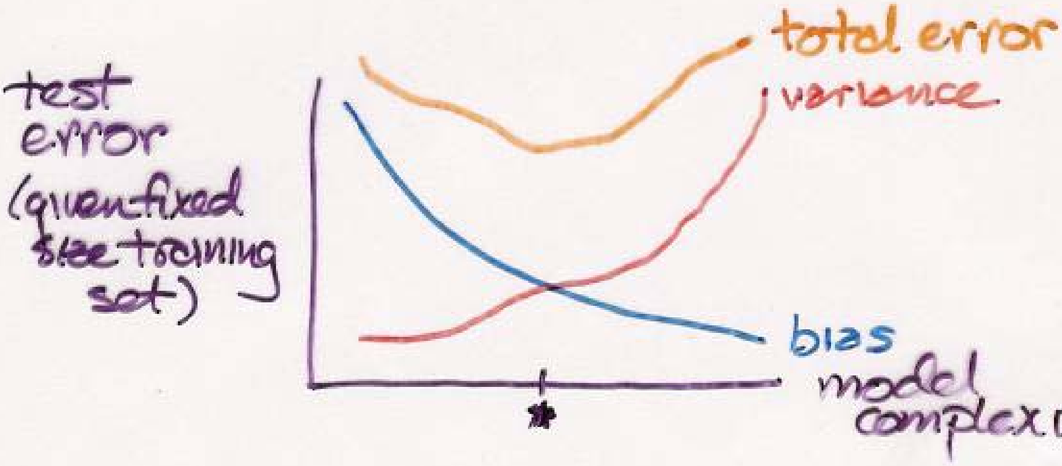
variance ~ degree to which model is dependent on the particular training set T



high bias

high variance

Bias - Variance Trade Off



Ways of performing model selection

(1) heuristics based on training set size & # variables (e.g. for selecting # of hidden units in a neural net)

(2) validation / cross validation / bootstrapping

(3) model pruning / growing heuristics

(4) Bayesian methods

(5) regularization

(6) designing appropriate bias into model architecture and representations

Hinton's heuristic

$$\boxed{H = \frac{p \log_2 p}{2(I+O)}}$$

p = training set size
I = # inputs
O = # outputs
H = # hidden in 3 layer net

Task difficulty ~ entropy of output patterns (in bits)

$$= -\underbrace{p}_{\substack{\#\text{ patterns} \\ \text{in training} \\ \text{set}}} \sum_{\substack{\text{all possible} \\ \text{output} \\ \text{patterns}}} q(d^\alpha) \log_2 \underbrace{q(d^\alpha)}_{\substack{\text{relative probability of} \\ \text{desired output } d^\alpha}}$$

E.g., 2 training patterns, $x^1 - d^1$, $x^2 - d^2$

$\Rightarrow q(d^1) = q(d^2) = .5$

entropy in a single pattern $= -\sum_{\alpha=1}^{2} q(d^\alpha) \log_2 q(d^\alpha)$

$= -(.5 \log_2 .5 + .5 \log_2 .5)$

$= -(.5 \times -1 + .5 \times -1)$

$= 1$ <u>bit</u>

entropy in training set $= 1$ bit $\times 2 = 2$ bits

Assuming each weight contains about 2 bits of information, we should find # weights in network such that

information contained in weights = information contained in output patterns

2 × # weights in net = entropy of output patterns

If all output patterns appear just once, entropy $= p \log_2 p$

$\therefore \quad 2(IH + HO) = p \log_2 p$

$$H = \frac{p \log_2 p}{2(I+O)}$$

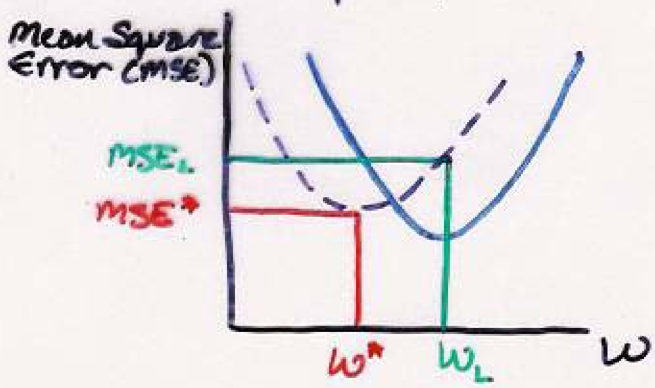
I: # inputs
H: # hidden
O: # output

Widrow's heuristic

$$\boxed{H \doteq \frac{.1P}{I+O}}$$

P = size of training set
I = # inputs
O = # outputs
H = # hidden units in a 3 layer net

Consider a 2 layer net (A inputs, 1 output) trained w/ LMS. Assume input patterns are independent, zero mean.



— = MSE based on training set
$$= \frac{1}{P}\sum_{\alpha=1}^{P}(y^\alpha - d^\alpha)^2$$

--- = MSE based on environment
$$= \lim_{P\to\infty} \frac{1}{P}\sum_{\alpha=1}^{P}(y^\alpha - d^\alpha)^2$$

w^* = optimal weights

MSE^* = best possible generalization performance

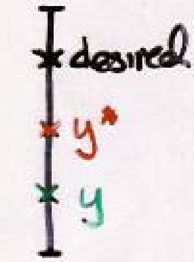
w_L = weights obtained by learning

MSE_L = expected generalization performance

The closer w_L approaches w^*, the better generalization performance will be. This depends on training set size, P.

Define $MSE_{excess} = \frac{1}{P}\sum_{\alpha=1}^{P}(y^\alpha - y^{*\alpha})^2$

output obtained with w^*
output obtained with w_L



$$MSE_{excess} = \frac{A}{P} MSE^* \quad \text{(Widrow \& Walach, 1984)}$$

If we generalize this result to multi-layered nets and treat A as the total number of weights in network:

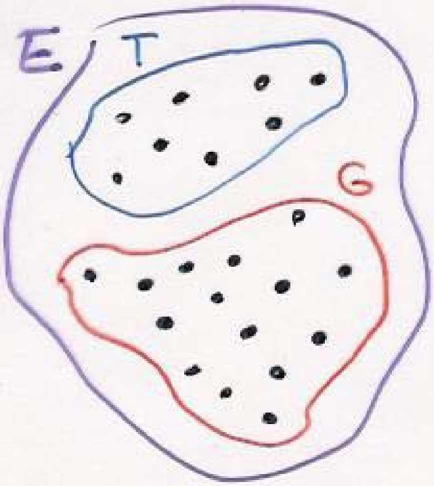
$$A = P \cdot \frac{MSE_{excess}}{MSE^*} = (I+O)H$$

H = # hidden
I = # input
O = # output

Allowing $MSE_{excess}/MSE^* = 10\%$

$$(I+O)H = .1P \longrightarrow H = \frac{.1P}{I+O}$$

Validation method



E: pattern environment (all possible assoc.)

T: training set

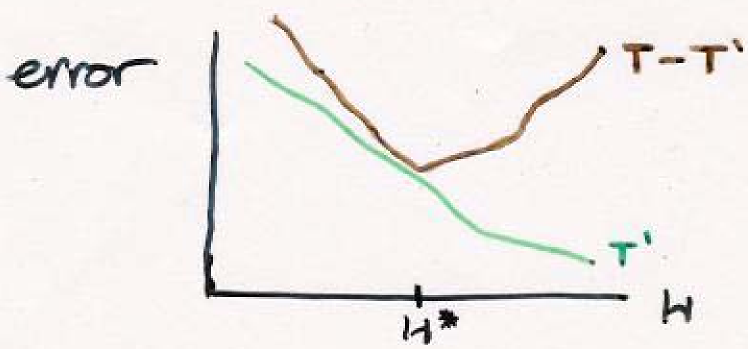
G: generalization set

$E = T \cup G$

Problem: Given T, find the number of hidden units, H, that will maximize performance on G.

Approach: Train net on only a part of T. Use remainder of T to estimate generalization performance.

Generate T' by selecting, say, 80% of examples in T at random. Train net on T' and test on T−T' for various values of H (and potentially, for various T')



Find value of H, H*, such that performance on T−T' is maximized. When net is trained on T with H* hidden, can expect comparable generalization performance

Neural net regularization

In statistics & computer vision, regularization is used to achieve a "smooth" interpolation surface.

Smoothness criterion $S = \int \left| \frac{d^m}{dx^m} f(x) \right|^2 dx$

Incorporate into objective function: $E = \sum (d_i - f(x_i))^2 + \lambda S$

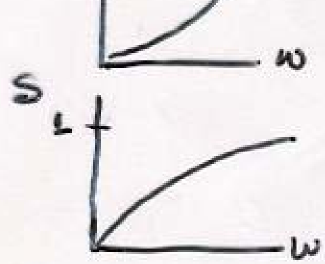
regularization parameter

There are methods for literally smoothing the function computed by a neural net, but regularization can also be used to bias the network in other ways

E.g., to encourage networks with fewer weight parameters

weight decay $S = \sum_{i,j} w_{ij}^2$

weight elimination $S = \sum_{i,j} \frac{w_{ij}^2}{w_0^2 + w_{ij}^2}$

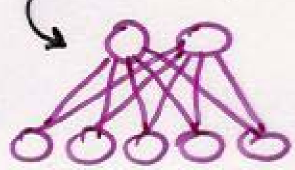


Unless weight is serving a useful purpose — as determined by back prop error forcing it away from zero — disable the connection.

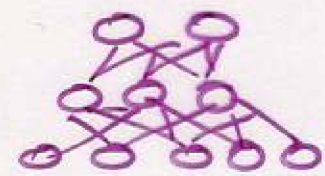
Designing bias into network architecture

Direct I/O connections to learn easy parts of task

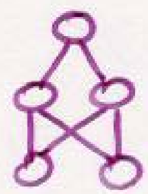
- Nettalk performs at about $\underset{\text{(guess)}}{70\%}$ without hidden units



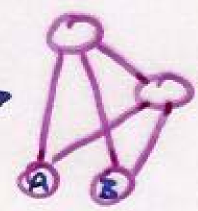
Performance up to 100% with hidden units



- Hidden units useful for handling exceptions
- E.g., XOR



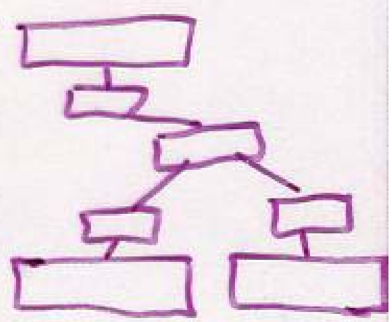
for easy parts of task



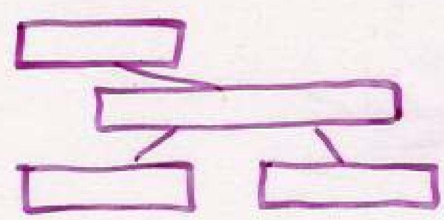
hidden units discover higher order features critical to performance (here, A & B)

Modular architectures

- Reserving pools of hidden units for particular aspects of task
- E.g., family trees task
  Pool of hidden units associated with each input and output element

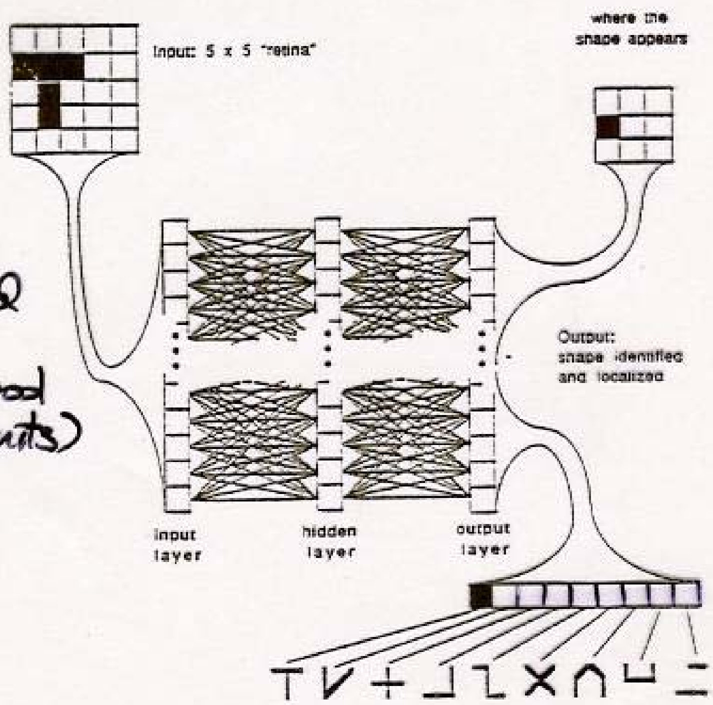


- Compare to undifferentiated architecture

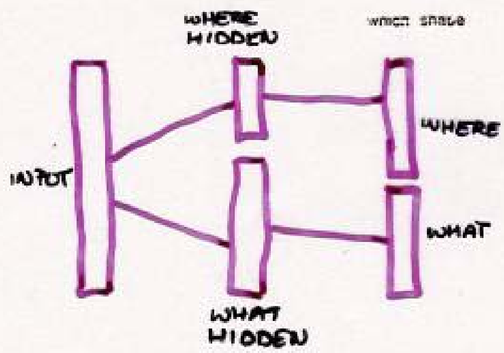


- E.g., Rueckl's What/Where network

Rueckl's what/where network

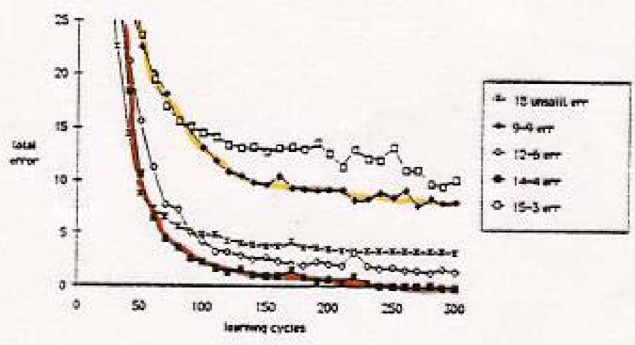


undifferentiated architecture (just 1 big pool of hidden units)



specialized architecture

specialized architecture has same # of hidden units but fewer connections



= undifferentiated (18 hidden)

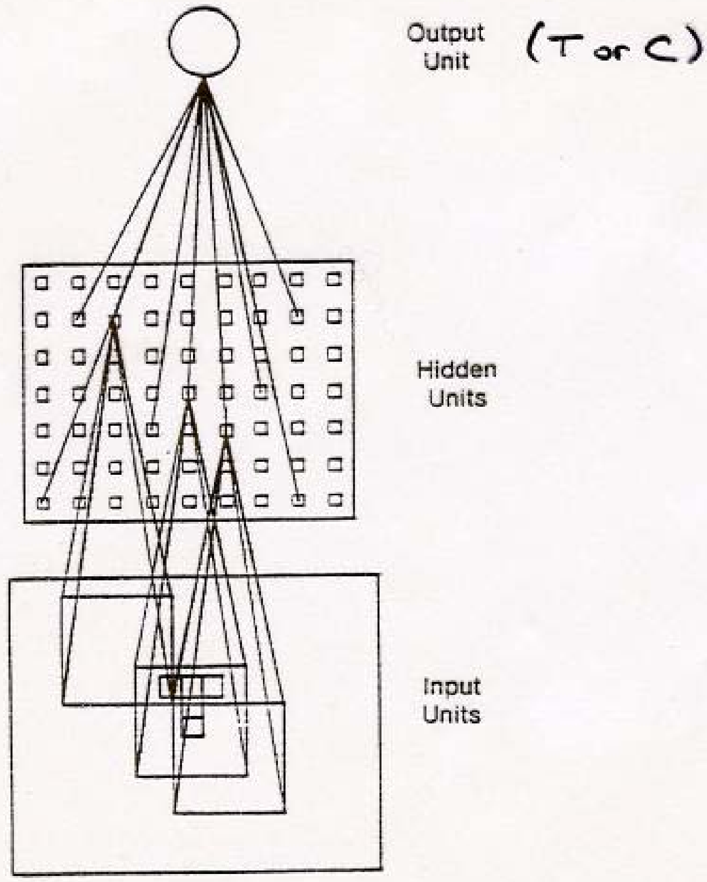
= specialized (4 hidden in where, 14 in what)

- By splitting hidden units, Rueckl tells system how difficult each task is and that information relevant to one task is not relevant to the other
- Relation to two cortical visual systems (parietal & temporal pathways)

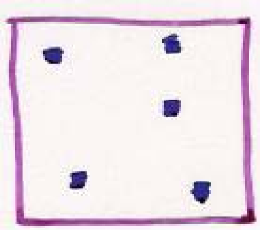
Designing bias into network architecture (contd.)

Local receptive fields

- For problems where input units have a topographic organization
- E.g., T-C network from PDP1:8



- Activities of nearby input units are likely to be jointly related to task; activities of distant units unlikely to be related
- I.e., task will not require higher order features consisting of distant inputs

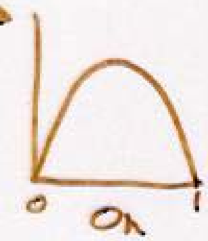


Designing bias into network architecture, dynamics, & learning

Constraints on activities

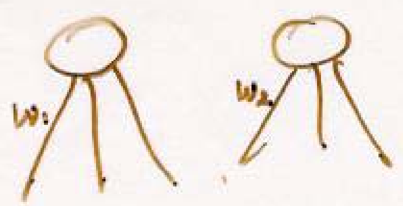
e.g. reduce amount of information flowing through net by encouraging binary-valued hidden units

$$E = \sum_p \sum_i (d_i^p - o_i^p)^2 + \underbrace{\sum_{h \in hidden} o_h(1-o_h)}$$



Constraints among weights

E.g., T-C problem: Each hidden unit should detect the same feature, but shifted in position



Set $w_1 = w_2$ initially

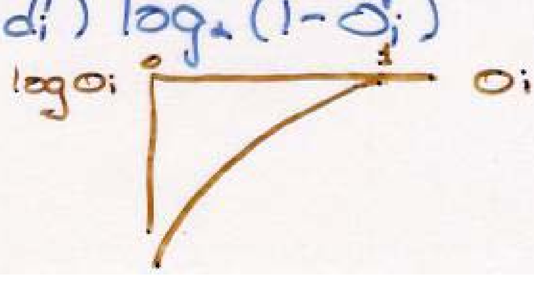
$\Delta w_1 = \Delta w_2 = -\varepsilon(\partial E/\partial w_1 + \partial E/\partial w_2)$

Variations in the error function

E.g., minimize cross entropy between desired & actual conditional probability distributions (instead of LMS)

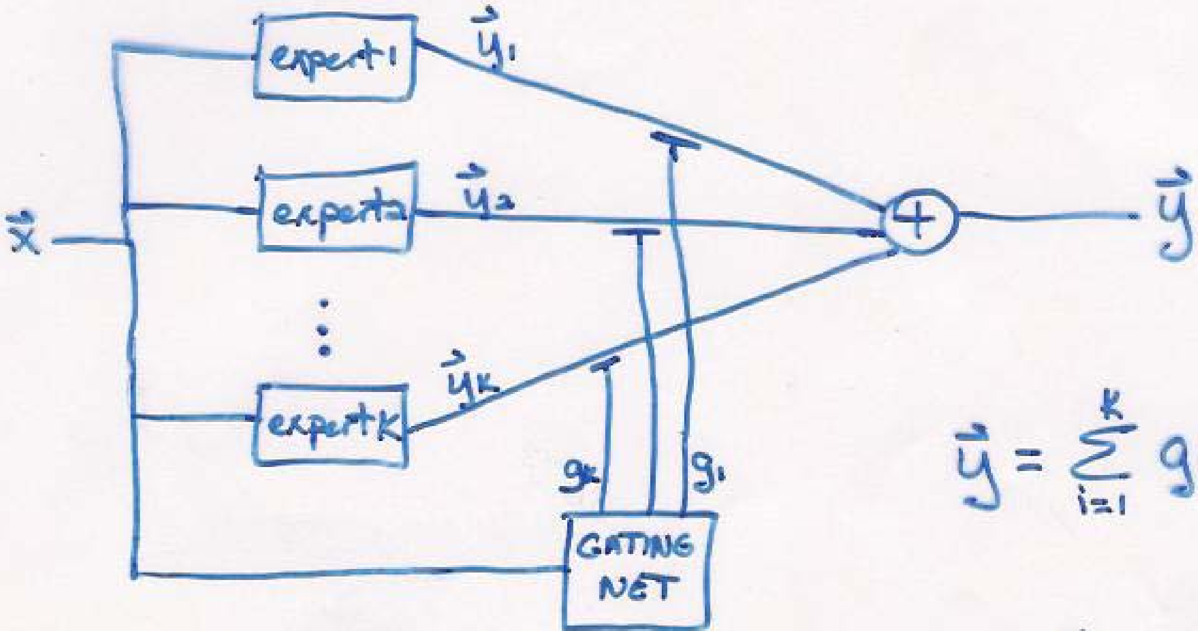
Interpret d_i and o_i as probability that output is true.

$$E = -\sum_p \sum_i d_i^p \log_2 o_i^p + (1-d_i^p) \log_2 (1-o_i^p)$$



Specialized Activation Functions

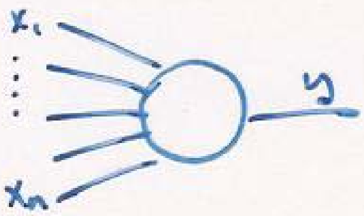
Mixture of Experts (Jacobs, Jordan, Nowlan, & Hinton 91)



$$\vec{y} = \sum_{i=1}^{K} g_i \vec{y}_i$$

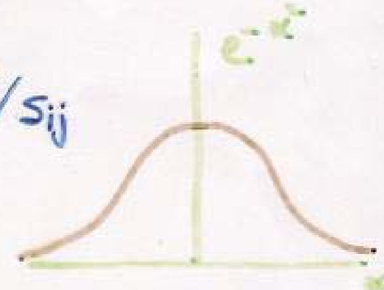
$$E = -\ln \sum_{i=1}^{K} g_i e^{-\frac{1}{2}|d - y_i|^2}$$

Radial Basis Functions

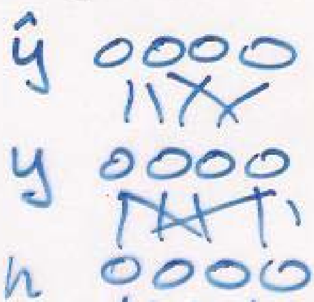


$$net_i = \sum (x_j - w_{ij})^2 / s_{ij}$$

$$y_i = e^{-net_i}$$



Normalized Exponential Transform, a.k.a. softmax (Bridle 91)



$$y_i = \sum w_{ij} h_j \quad \text{(linear)}$$

$$\hat{y}_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

for classification:

$$E = -\ln \hat{y}_d$$

index of desired output

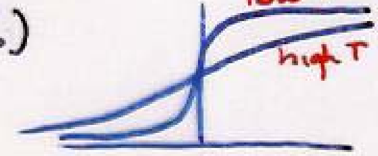
NOTE: $0 \le \hat{y}_i \le 1$

$\sum \hat{y}_i = 1$

Designing bias into the net (contd.)

Introduce parameters other than weights/biases and perform gradient descent in these parameters as well.

- E.g., "temperature" (steepness of sigmoid)



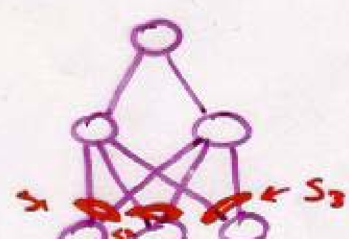
$$o_i = \frac{1}{1 + e^{-net_i / T_i}}$$

Compute $\partial E / \partial T_i$ $\qquad \Delta T_i = -\epsilon \frac{\partial E}{\partial T_i}$

- E.g., input salience term

In the "real world" many inputs are irrelevant to task at hand. Would like to suppress them.

$$\underset{\text{layer 2}}{net_i} = \sum w_{ij}\, \underset{\substack{\text{activity}\\ \text{of unit } j\\ \text{in layer 1}}}{o_j}\, \underset{\substack{\text{salience of}\\ \text{unit } j \text{ in}\\ \text{layer 1}\\ (0-1)}}{s_j}$$



Compute $\partial E / \partial s_j$ $\qquad \Delta s_j = -\epsilon\, \partial E / \partial s_j$

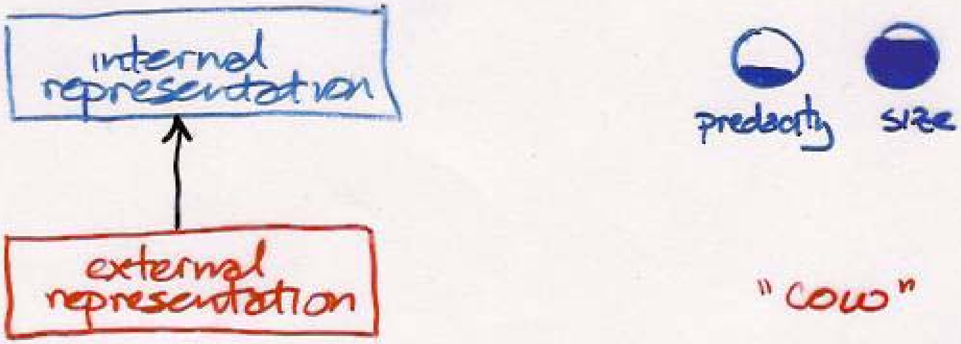
Equivalent to changing all outgoing weights from input unit simultaneously

- These parameters allow you to cut across weight space diagonally (low $T \equiv$ turning up all weights coming into unit; low $s \equiv$ turning down all weights coming from unit)

Example of task-specific architecture & activation fn.

Similarity network (Todd & Rumelhart)

Problem: what internal representations do people develop for stimuli and objects in their world?

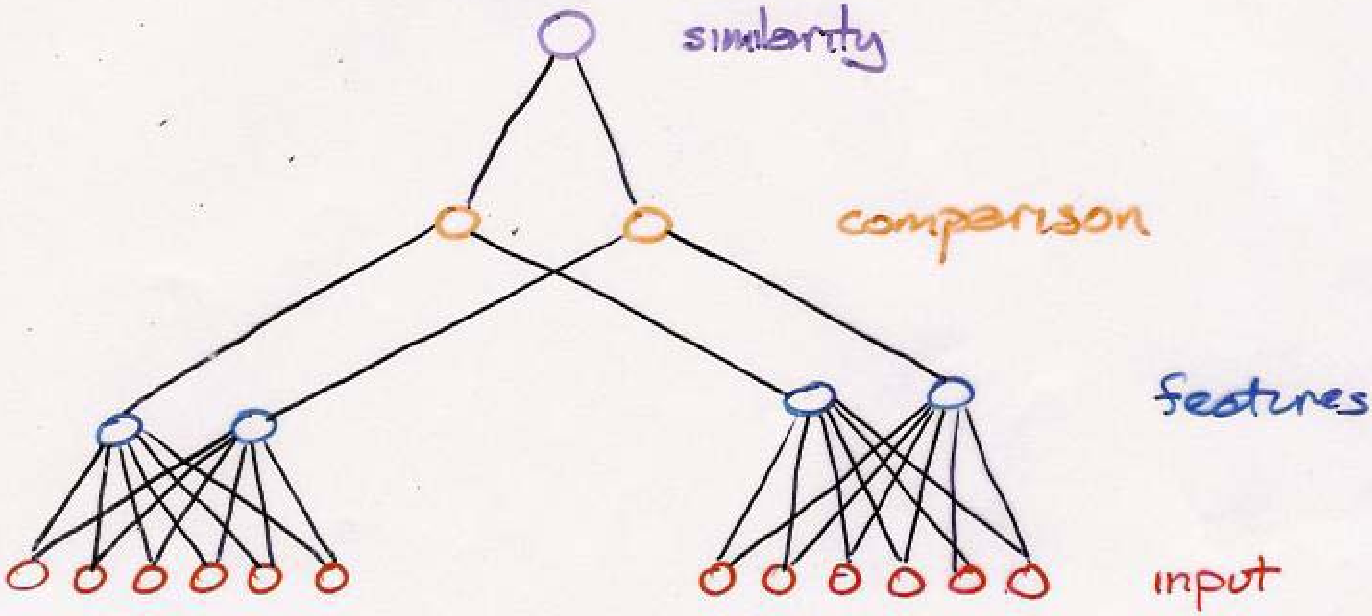


multi-dimensional scaling: method to discover underlying internal representations from human similarity judgements

similarity(lion, bear) = .9 similarity(pig, bear) = .1
similarity(cow, lion) = .2 similarity(pig, cow) = .7

If each animal corresponds to a point in n-dimensional semantic feature space, can we lay the points out such that the distance between points is inversely proportional to the similarity judgements?

Can we use a connectionist network to discover semantic features on the basis of similarity judgements?



Local input representation: inputs are just labels; makes no assumption about similarity of different objects

Input → feature mapping is the same for left & right inputs (enforced with weight constraints)

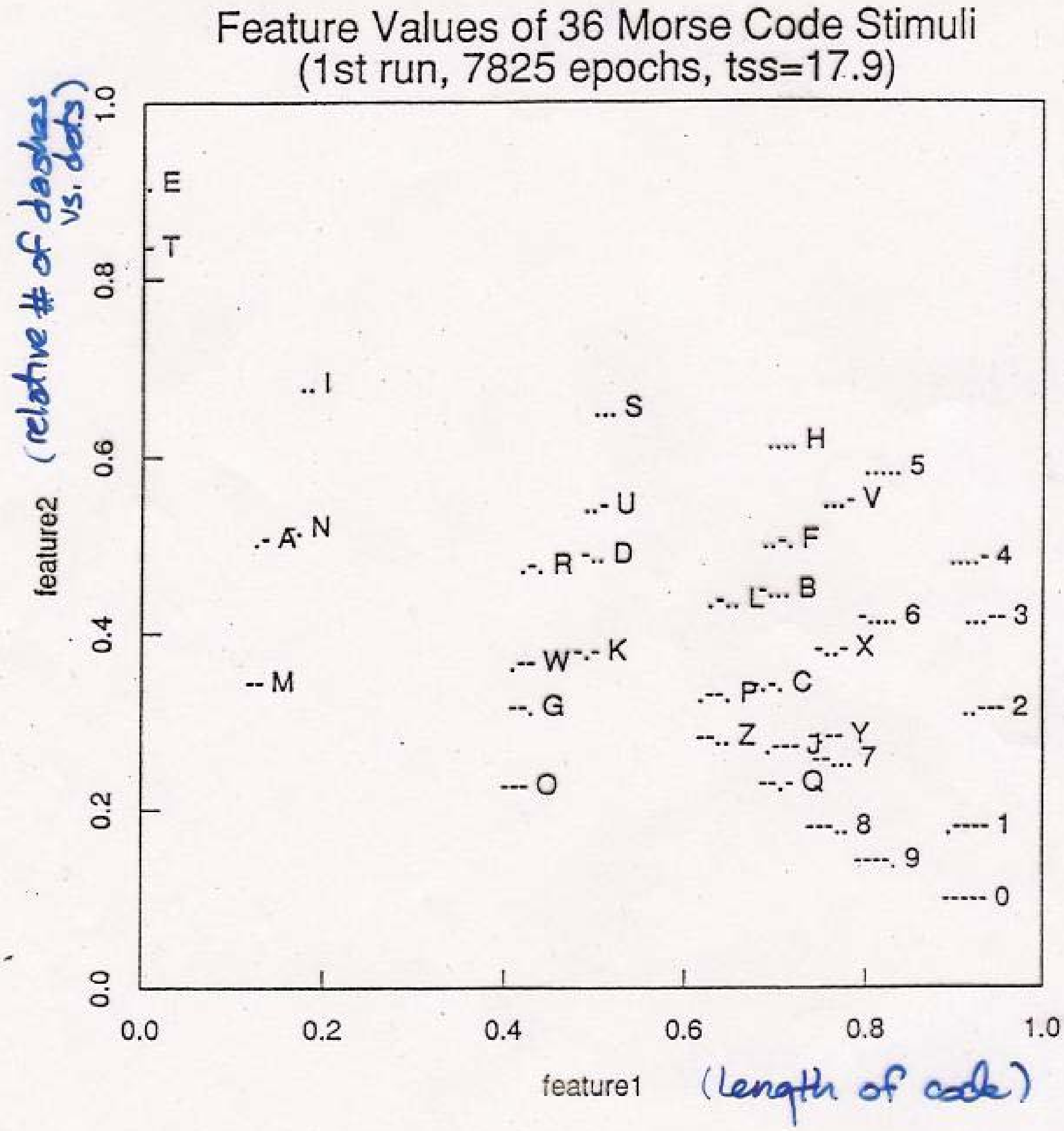
Activation functions of comparison units, c_i, and similarity unit, s, are fixed:

$$c_i = (\underbrace{f1_i}_{\text{feature unit } i \text{ of pool } 1} - \underbrace{f2_i}_{\text{feature unit } i \text{ of pool } 2})^2 \qquad s = \frac{1}{(\Sigma c_i)^{1/2}}$$

Training procedure:

present input pair; compute similarity measure
compute $\partial E/\partial s$, where $E = (s - \underbrace{d}_{\text{desired similarity}})^2$
compute $\partial E/\partial c_i = \partial E/\partial s \cdot \partial s/\partial c_i$
compute $\partial E/\partial f1_i$, $\partial E/\partial f2_i$ from $\partial E/\partial c_i$
adjust input → feature weights

Featural representation developed by network based on similarity judgements for morse code letters:



Connectionist approach allows a variety of similarity measures to be used (vs. traditional multi-dimensional scaling):

- weightings on individual features: $s = \frac{1}{(\sum w_i c_i)^{1/2}}$
- city block (Hamming) distance: $s = 1 / \sum c_i$

  $c_i = | f1_i - f2_i |$
- asymmetric similarity measure (e.g., Tversky): $s = \sum c_i - \sum d_i$

  $c_i = f1_i f2_i$

  $d_i = w_1 f1_i (1 - f2_i) + w_2 f2_i (1 - f1_i)$