Connectionist and Statistical Language Processing

Course Review

Matthew Crocker and Frank Keller

crocker@coli.uni-sb.de keller@coli.uni-sb.de

Course Overview

- Machine learning of natural language:
  - ❑ Cognitive models of language learning/development
  - ❑ Data-intensive learning of linguistic knowledge
  - ❑ Machine learning for applications
- Kinds of learning:
  - ❑ Supervised
  - ❑ Unsupervised
- Kinds of algorithms/techniques:
  - ❑ Connectionist modelling
  - ❑ Statistical learning methods

Exam details

- Date/Time: 12 February, 14:15-15:45 (90 minutes)
- Location: Konferenzraum (2.11)
- Format: Answer 5 of 6 questions
  - 1 obligatory question (connectionism)
  - 2 further questions (connectionism)
  - 1 obligatory question (machine learning)
  - 2 further questions (machine learning)
- Each question will consist of 3-5 sub-parts
- Each question has equal value (20%)

Course contents: connectionism

- Introduction: Stochastic Language Learning
  - Connectionism and the brain
  - The appeal of connectionism
  - Overview of connectionism in language processing
  - Basic connectionist models: nodes and activations
- Foundations of Connectionist Models
  - Simple connectionist models and their properties: The perceptron
  - Multi-layer perceptrons: feed-forward networks and internal representations
  - The encoding problem: Localist and distributed representations
  - Generalisation, association, and auto-association
- Connectionist Models of Language
  - Modelling acquisition of the English past-tense and reading aloud
  - Processing sequences: Simple recurrent networks
  - Modelling acquisition of hierarchical syntactic knowledge

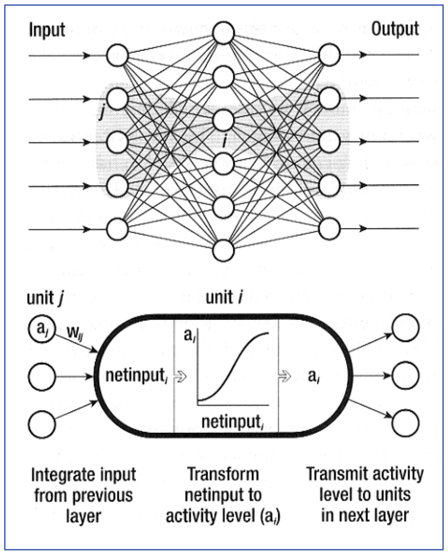
Summary of network architecture

- The activation of a unit *i* is represented by the symbol a_i.
- The extent to which unit *j* influences unit i is determined by the weight w_{ij}
- The input from unit *j* to unit *i* is the product: $a_j * w_{ij}$
- For a node *i* in the network:

$$netinput_i = \sum_j w_{ij} a_j$$

- The output activation of node *i* is determined by the activation function, e.g. the logistic:

$$a_i = f(netinput_i) = \frac{1}{1 + e^{net_i}}$$


Input
Output
j
i
unit j
unit i
a_j
w_{ij}
netinput$_i$
a_i
netinput$_i$
a_i
Integrate input from previous layer
Transform netinput to activity level (a_i)
Transmit activity level to units in next layer

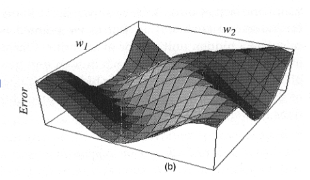
Learning in connectionist networks

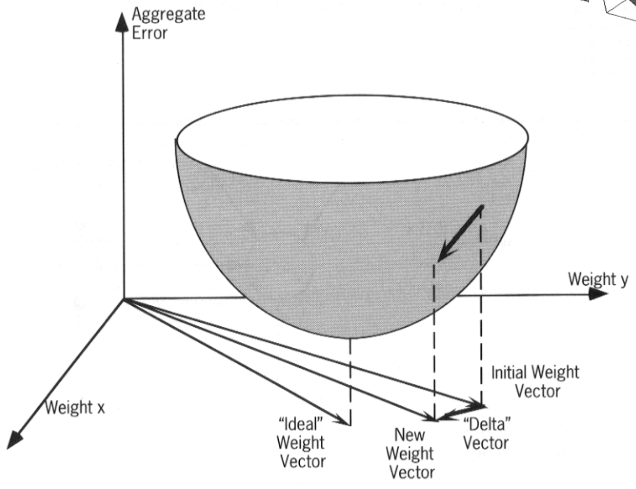
- **Supervised learning** in connectionist networks involves successively adjusting connection weights to reduce the discrepancy between the *actual output activation* and the *correct output activation*
  - An input is presented to the network
  - Activations are propagated through the network to its output
  - Outputs are compared to "correct" outputs: difference is called *error*
  - Weights are adjusted
- The Delta Rule:

$$\Delta w_{ij} = [a_i(\text{desired}) - a_i(\text{obtained})] a_j \eta$$

  - [a_i(desired)-a_i(obtained)] is the difference between the desired output activation and the actual activation produced by the network
    - What is the "error"?
  - a_j is the activity of the contributing unit *j*
    - How much activation is this unit responsible for?
  - η is the learning rate parameter.
    - How rapidly do we want to make changes?

Visualising the error „surface“





Gradient descent and the delta rule

- The perceptron convergence rule: $\Delta w = \eta \delta a_{in}$
- Our revised learning rule, based on gradient descent is:

$$\Delta w = 2\eta\delta F^* a_{in}$$

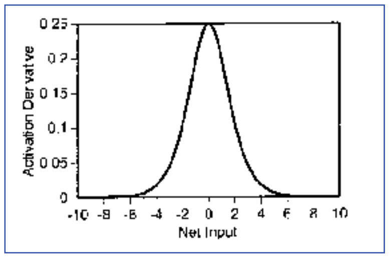
  - where F^* is the slope of the activation function
- If the activation function is linear, the slope is constant:

$$\Delta w = k\delta a_{in}$$

  - where k is a constant representing the learning rate and slope
- This corresponds to the original Delta rule:
  - It is straight-forward to calculate
  - Performs gradient descent to the bottom of an the error curve
  - Δw is proportional to $(t_{out}-a_{out})$, so changes get smaller as error is reduced
  - In 2-layer networks, there is a single minimum which gradient descent learning is guaranteed to find a solution, it one exists.

The dynamics of weight changes

- Learning rate: predetermined constant
- The error: large error = large weight change
- The momentum: how much of the previous weight change affects the current weight change
- The slope of the activation function:
  - Is largest for netinputs = 0, and for activations = .5
  - Small netinputs co-occur with small weights
  - Small weights tend to occur early in training



- The result: bigger changes during early stages of learning
  - More resilience in older network: harder to teach new tricks!

Network Training & Performance

- The training phase involves
  - Presenting an input pattern, and computing the output for the network using the current connection weights: $a_{out}=f(\sum_{in} w_{out,in} \times a_{in})$
  - Calculating the error between the desired and the actual output $(t_{out} - a_{out})$
  - Using the Delta rule (appropriate for the activation function):

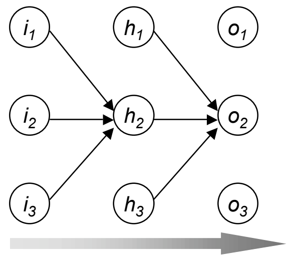
$$\Delta w = \eta(t_{out} - a_{out})a_{out}(1 - a_{out})a_{in}$$

- One such cycle is called a <u>sweep</u>
- A sweep through each patter is called an <u>epoch</u>
- We can define the <u>global error</u> of the network, as the average error across all input patterns, *k:*
  - One common measure is the square root of mean error

$$\text{rms error} = \sqrt{\frac{\sum_k (\vec{t}_k - \vec{o}_k)^2}{k}}$$

  - Squaring avoids positive and negative error cancelling each other out

Backpropagation of Error



(a) Forward propagation of activity:

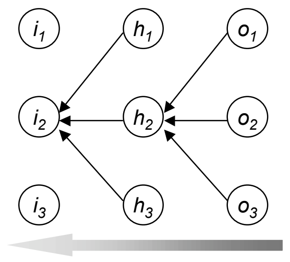
$$\text{netinput}_{out} = \sum w \cdot a_{hidden}$$

$$a_{out} = F(\text{netinput}_{out})$$

(b) Backward propagation of error:

$$\text{netinput}_{hidden} = \sum w \cdot \delta_{out}$$

$$\delta_{hidden} = F(\text{netinput}_{hidden})$$



Learning in Multi-layer Networks

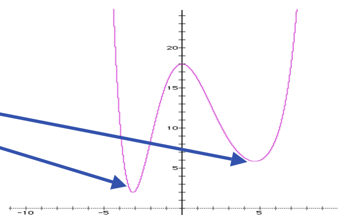
- The generalised Delta rule: $\Delta w_{ij} = \eta \delta_{ip} a_j$

  For output nodes: $\delta_{ip} = f'(net_{ip})(t_{ip} - a_{ip})$

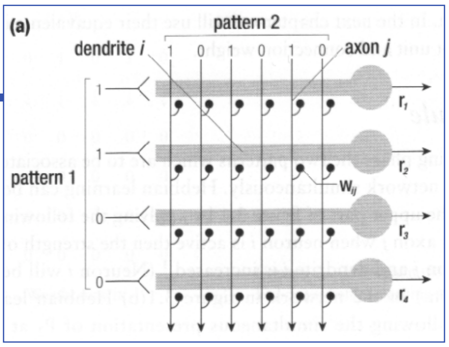
  For hidden nodes: $\delta_{ip} = f'(net_{ip}) \sum_k \delta_{kp} w_{ki}$

  where, $f'(net_{ip}) = a_{ip}(1 - a_{ip})$

- Multi-layer networks can, in principle, learn any mapping function:
  - Not constrained to problems which are linearly separable
- While there exists a solution for any mapping problem, backpropagation is not guaranteed to find it
  - Unlike the perceptron convergence rule
- Why? Local minima:
  - Backprop can get trapped here
  - Global minimum (solution) is here



Learning: Hebb's rule



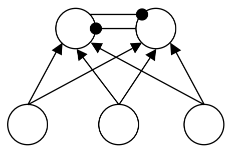
- The idea behind Hebbian learning is simple:
- The two patterns to be associated are presented simultaneously
- If there is activity on input axon *j*, when neuron *i* is active, then the connection weight w_{ij} (between axon *j* and dendrite *i*)is increased
- The Hebb rule: $\boxed{\square w_{ij} = \square a_i a_j}$
  - a_i is the activity of element *i* in P_1
  - a_j is the activity of element *j* in P_2
  - □is the learning rate parameter

Summary of Pattern Associators

- Associate multiple stimulus-response patterns in a single network
  - Networks can be represented as a weight matrix
- Weights are sensitive to similarity
  - The more similar, the higher the netinput; the *dot product* of P and W
- Important properties
  - Generalisation: robust to noisy input
  - Fault tolerance: robust to loss/damage
  - Prototype extraction & noise reduction
- Biologically Plausible:
  - Learning is strictly local
  - Reinforcement based
- Auto Association
  - We can also train a network to associate a given pattern with itself
    - Noise reduction, prototype extraction
    - = category formation (unsupervised)

Architecture of Competitive Networks

- A simple network:
  - Inputs are fully connected to outputs by feed-forword connections
  - Outputs may be connected to each other by *inhibitory* connections



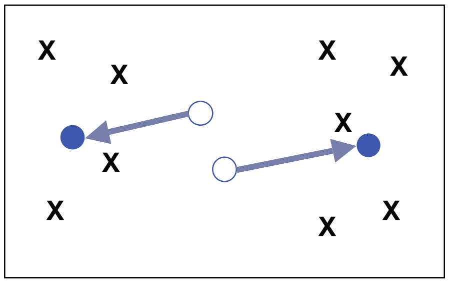
- Outputs compete until only one remains active
  - Or, simply the unit with highest activation wins
- Excitation of outputs: $\text{netinput}_i = \sum_j a_j w_{ij}$
  - Dot product of input activations and the weight vector the the output
- Competition:
  - Output activations are compared, unit with highest activation wins
  - Or, direct competition among outputs, via inhibitory connections:
    - Active units force other units to become inactive

Overall behaviour

- Netinput to an output unit is greatest when it's weight vector is most similar to the input vector
- Training makes the weight vector for a particular winning unit more similar to the input pattern
- It is therefore also likely to be the "winning unit" for similar patterns, and therefore learn to respond to those patterns as well
- The weight vector for a particular output unit learns to respond to similar input patterns
  - Because these patterns are all slightly different, the learned weights cannot exactly mimic the associated inputs
  - Rather, the learned weights will be an average of the patterns, based on the frequency of presentation during training
- The competitive network can therefore learn to categorise similar inputs without any "teacher": unsupervised learning

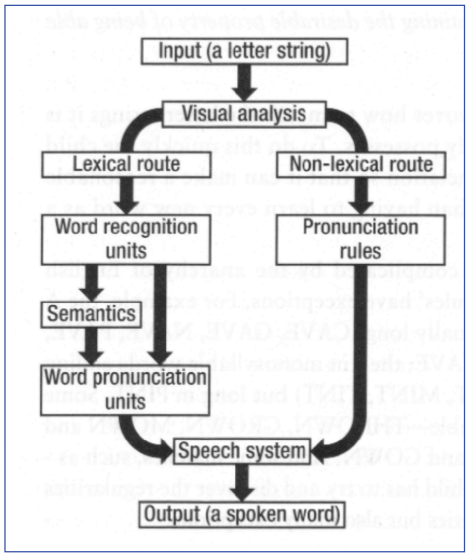
Visualising competitive learning

- Represent input patterns & weight vectors in multi-dimensional space
  - weight vectors for the output units have a random relation to the input patterns
  - Competitive learning changes the weight vector for a particular output so that it becomes the average for a subset of inputs
  - More outputs enable the network to more finely categorise the inputs



Reading Aloud: Dual Route Model

- The standard model of reading posits two independent routes leading to pronunciation of a word, because …
  - People can effortless pronounce words they have never seen:
    - SLINT or MAVE
  - People can pronounce words which break the “rules”:
    - PINT or HAVE
- One mechanism uses general rules for pronunciation
- The other mechanism stores pronunciation information with specific words

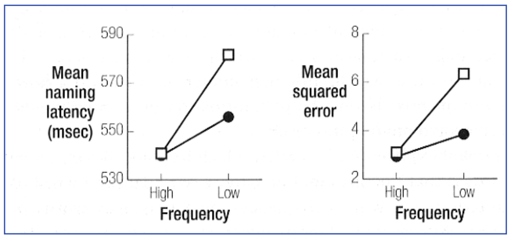


Word Frequency Effects

- Common words are pronounced more quickly than uncommon words
  - This is true for most almost all aspects of human information processing
- Conventional (localist) explanation:
  - Frequent words require a lower threshold of activity for "the word recognition device" to "fire"
  - Infrequent words require a higher threshold of activity
- In the Seidenberg & McClelland model, naming latency is modelled by the error:
  - Word frequency is reflected in the training procedure
  - Phonological error is reduced by training, and therefore lower for high frequency words
- The explanation of latencies in terms of error follows directly from the network's architecture and the training regime

Frequency x Regularity

- In addition to faster naming of frequent words, human subjects exhibit:
  - Faster pronunciation of regular words (e.g GAVE or MUST) than irregular words(e.g. HAVE or PINT)
  - But, this effect interacts with frequency: it is only observed with low frequency words
- For regulars (filled circle) we observe a small effect of frequency
  - It takes slightly longer to pronounce the low frequency regulars
- For irregulars (open square) we observe a large effect of frequency
- The model precisely mimics this pattern of behavior in the error
- 2-route: the confusion of the lexical and rule outcome requires resolution
  - Lexical route wins faster for high frequency words

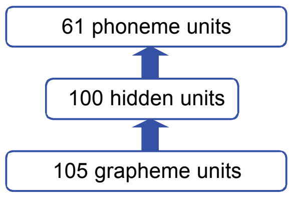


Summary of Seidenberg & McClelland (1989)

- What has the model achieved
  - The model is a single mechanism with no lexical entries or explicit rules
  - Response to an input is a function of the networks entire experience
    - Reflects previous experience on a particular word
    - Experience with words resembling that string
  - E.g. <u>specific</u> experience with HAVE is sufficient to overcome the <u>general</u> information that _AVE is usually a long vowel
  - The network can produce a plausible pronunciation for MAVE, but error is introduced by experience with inconsistent words like HAVE
- Performance
  - 97% accuracy on pronouncing learned words
  - Models: frequency & interaction with regularity, neighborhood, consistency
- Limitations: It is not as good as humans at
  - Reading non-words (model get 60%, humans 90%)
  - Lexical decision (FRAME is a word, but FRANE is not)

The Plaut et al (1996) architecture

- The architecture of the Plaut *et al* network:
  - The are a total 105 possible orthographic onsets, vowels, and codas
  - The are 61 possible phonological onsets, vowels and codas



- What is the performance on non-words?
  - For consistent words (HEAN/DEAN): model (98%) *versus* human (94%)
  - For inconsistent words (HEAF/DEAF/LEAF): model (72%), human (78%)
- Representations:
  - The right encoding scheme is essential for modelling the findings
  - They assume this knowledge could be partially acquired prior to reading
  - Doesn’t scale to polysyllabic words
- Doesn’t not explain the double dissociation:
  - ✓ Surface dyslexics (can read exceptions, but not non-words)
  - ✗ Phonological (can pronounce non-words, but not irregulars)

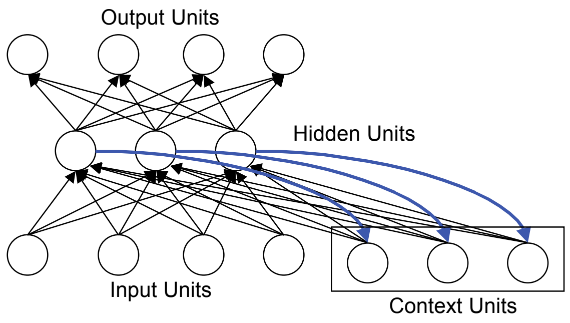
Simple Recurrent Networks

- Recurrent networks are powerful for executing and learning complex sequences, but difficult to design
- Simple recurrent networks can learn any sequence given as input
- We can tell they've learned by training them to predict the next item
- Hidden units are connected to "context" units:

  These correspond to "state" units: they remember the state of the network on the previous time step

  The hidden units are able to recycle information over multiple time steps

  Dynamic memory: Identical inputs can be treated differently depending on context



Discovering word boundaries

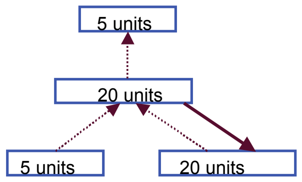
- We often take for granted the existence of words, and yet for the child language learner, input is largely in the form of an unsegmented acoustic stream.
- How do children learn to identify word boundaries in such a signal?

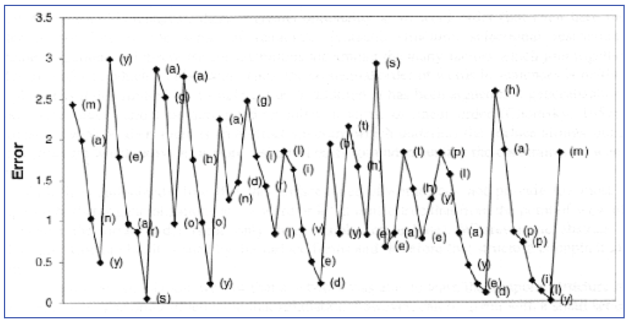
- Example: Predicting the next sound
  - ❑ Problem: discovering word boundaries in continuous speech
    - ✚ Approximated by a corpus of continuous phonemes
  - ❑ Task: network is presented with one phoneme and attempts to predict the next one
  - ❑ *Manyyearsagoaboyandgirllivedbytheseatheyplayedhappily*

- At time *t*: the network knows both the current input (phoneme at time *t*) and the results of processing at time *t-1* (context units)
  Problem: discovering word boundaries in continuous speech

Predicting the next sound

- We can examine the error:
  - High error at the onset of words
  - Decreases during a word, as the sequence is increasingly predictable
  - High error at word onset demonstrates the network has discovered word boundaries





Structure of Training Environment

- Categories of lexical items

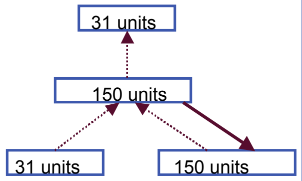
Category	Examples
NOUN-HUM	man,woman
NOUN-ANIM	cat,mouse
NOUN-INANIM	book,rock
NOUN-AGRESS	dragon,monster
NOUN-FRAG	glass,plate
NOUN-FOOD	cookie,sandwich
VERB-INTRAN	think,sleep
VERB-TRAN	see,chase
VERB-AGPAT	move,break
VERB-PERCEPT	smell,see
VERB-DESTROY	break,smash
VERB-EAT	eat

- Template for sentence generator

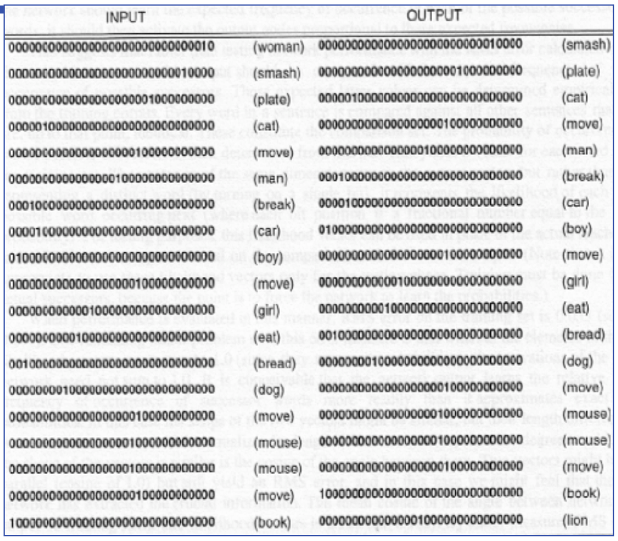
WORD 1	WORD 2	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-ANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTROY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

Input encoding & training

- Localist representation of each word (31 bits)
  - ❑ Nothing of the word class is reflected
- 10000 random 2-3 word sentences
  - ❑ 27,354 sequence of 31 bit vectors
- Architecture:



- Trained on 6 complete passes through the sequence



INPUT		OUTPUT	
0000000000000000000000000000010	(woman)	0000000000000000000000000010000	(smash)
0000000000000000000000000010000	(smash)	0000000000000000000001000000000	(plate)
0000000000000000000001000000000	(plate)	0000010000000000000000000000000	(cat)
0000010000000000000000000000000	(cat)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000001000000000000000	(man)
0000000000000001000000000000000	(man)	0001000000000000000000000000000	(break)
0001000000000000000000000000000	(break)	0000100000000000000000000000000	(car)
0000100000000000000000000000000	(car)	0100000000000000000000000000000	(boy)
0100000000000000000000000000000	(boy)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000001000000000000000000	(girl)
0000000000001000000000000000000	(girl)	0000000000100000000000000000000	(eat)
0000000000100000000000000000000	(eat)	0010000000000000000000000000000	(bread)
0010000000000000000000000000000	(bread)	0000000010000000000000000000000	(dog)
0000000010000000000000000000000	(dog)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000001000000000000	(mouse]
0000000000000000001000000000000	(mouse)	0000000000000000001000000000000	(mouse]
0000000000000000001000000000000	(mouse)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	1000000000000000000000000000000	(book)
1000000000000000000000000000000	(book)	0000000000000000100000000000000	(lion

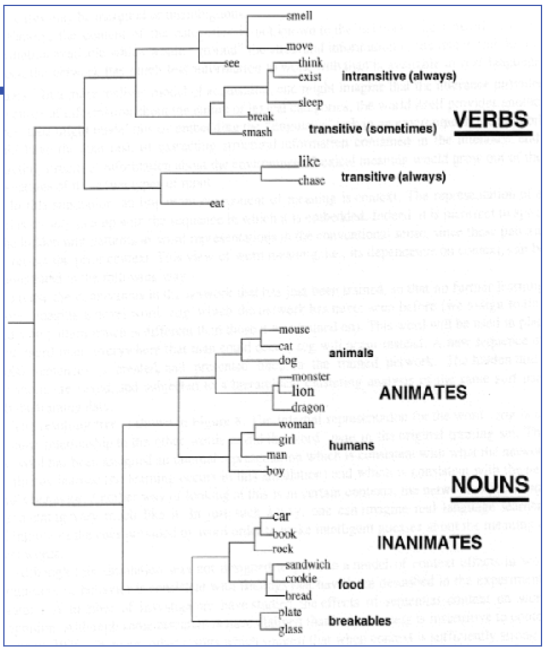
Performance

- Training yields an RMS error of 0.88
- RMS error rapid drops from 15.5 to 1, by simply learning to turn all outputs off (due to sparse, localist representations)

- Prediction is non-deterministic: next input cannot be predicted with absolute certainty, but neither is it random
  - ❑ Word order and selectional restrictions partially constrain what words are likely to appear next, and which cannot appear.
  - ❑ We would expect the network to learn the frequency of occurrence of each possible successor, for a given input sequence
- Output bit should be activated for all possible following words
  - ❑ These output activations should be proportional to frequency
- Evaluation procedure:
  - ❑ Compare network output to the vector of probabilities for each possible next word, given the current word and context …

Cluster analysis:

- Lexical items with similar properties are grouped lower in the tree
- The network has discovered:
  - Nouns vs. Verbs
  - Verb subcategorization
  - Animates/inanimates
  - Humans/Animals
  - Foods/Breakables/Objects
- The network discovers ordering possibilities for various work categories and “subcategories”

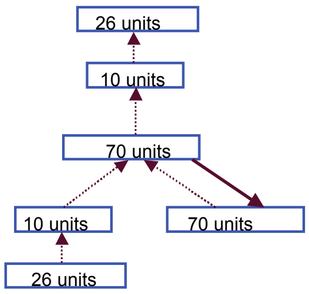


General discussion

- The network learns hierarchical categories and classes
  - Such classes are determined from word order/co-occurrence
  - Learning takes place purely on the basis of observable data
    - No pre-specified localist representations, etc.
- Predicts “context” effects in processing:
  - Consistent with findings that human lexical access is sensitive to context
    - Controversial: there is evidence both for (Tabossi) and against (Swinney) immediate context effects in lexical access
  - And that it is word classes that are predicted, not individual words

Learning Constituency: Elman (1991)

- So far, we have seen how SRNs can find structure in sequences
- How can complex structural relationships such as constituency be represented?
- The Stimuli:
  - Lexicon of 23 items
  - Encoded orthogonally, in 26 bit vector
- Grammar:
  - S ➔ NP VP "."
  - NP ➔ PropN | N | N RC
  - VP ➔ V (NP)
  - RC ➔ who NP VP |who VP (NP)
  - N ➔ boy | girl | cat | dog | boys | girls | cats | dogs
  - PropN ➔ John | Mary
  - V ➔ chase | feed | see | hear | walk |live | chases | feeds | sees | hears | walks | lives
  - Number agreement, verb argument patterns



Training

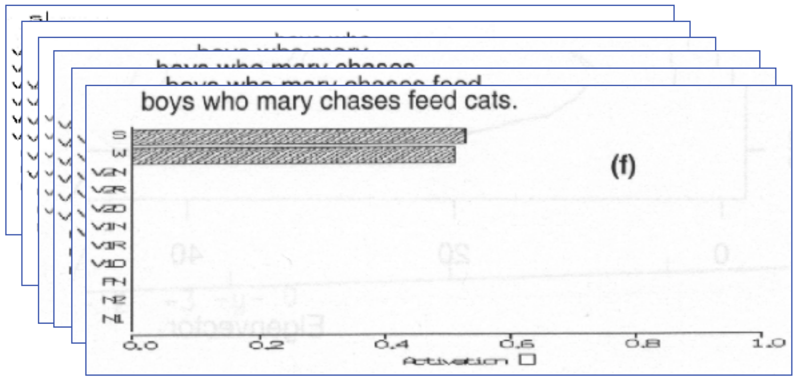
- Verb subcategorization
  - Transitives: hit, feed
  - Optional transitives: see, hear
  - Intransitives: walk, live
- Interaction with relative clauses:
  - *Dog who chases cat sees girl*
  - *Dog who cat chases sees girl*
  - Agreement can span arbitrary distance
  - Subcategorization doesn't always hold (superficially)
- Recursion: Boys who girls who dogs chase see hear
- Viable sentences: where should end of sentence occur?
  - Boys see (.) dogs (.) who see (.) girls (.) who hear (.) .
- Words are not explicitly encoded for number, subcat, or category

Training

- At any given point, the training set contained 10000 sentences, which were presented to the network 5 times
- The composition of sentences varied over time:
  - Phase 1: Only simple sentence (no relative clauses)
  - Phase 2: 25% complex and 75% simple
  - Phase 3: 50/50, mean sentence length 4.38
  - Phase 4: 75% complex, 25% simple, max: 16, mean: 6
- WHY?: Pilot simulations showed the network was unable to learn the task when given the full range of complex data from the beginning.
- Focussing on simpler data first, the network learned quickly, and was then able to learn the more complex patterns.
- Earlier simple learning, usefully constrained later learning

Processing complex sentences

- "Boys who mary chases feed cats"
  - Long distance
    - Agreement: Boys … feed
    - Subcategorization: chases is transitive but in a relative clause
    - Sentence end:all outstanding "expectations" must be resolved



Results

- Learning was only possible when the network was forced to begin with simpler input
  - This effectively restricted the range of data to which the networks were exposed during initial learning
  - Contrasts with other results showing the entire dataset is necessary to avoid getting stuck in local minima (e.g. XOR)
- This behaviour partially resembles that of children:
  - Children do not begin by mastering language in all its complexity
  - They begin with simplest structures, incrementally building their “grammar”
- But the simulation achieves this by manipulation the environment:
  - This does not seem an accurate model of the situation in which children learn language
  - While adults do modify their speech, it is not clear they make such grammatical modifications
  - Children hear all exemplars of language from the beginning

Training with Incremental Memory

- While it’s not the case that the environment changes, it true that the child changes during the language acquisition period
- Solution: keep the environment constant, but allow the network to undergo change during learning
  - Phase 1:
    - Recurrent feedback was eliminated after every 3 or 4 words, by setting all context units to 0.5
  - Phase 2:
    - Memory window increased to 4-5 words
  - Phase 3: 5-6 word window
  - Phase 4: 6-7 word window
  - Phase 5: no explicit memory limitation implemented
- Performance: as good as on the previous simulation

The importance of starting small

- The representation of experience:
  - Exemplar-based learning models store all prior experience, and such early data can then be re-accessed to subsequently help form new hypotheses
  - SRNs do not do this: each input has it's relatively minor effect on changing the weights (towards a solution), and then disappears.
- Constraints on new hypotheses, and continuity of search:
  - Changes in a symbolic systems may lead to suddenly different solutions
    - This is often ok, if it can be checked against the prior experience
  - Gradient descent learning makes it difficult for a network to make dramatic changes in its solution: search is continuous, along the error surface
    - Once in a local minima, the network might not recover
- Network are most sensitive during the early period of learning:
  - Thus most learning occurs when information is least reliable
  - Non-linearity (the logistic activation function) means that weight modifications are less likely as learning progresses

Conclusions

- Learning language is difficult because:
  - Learning linguistic primitives is obscured by the full complexity of grammatical structure
  - Learning complex structure is difficult because the network lacks knowledge of the basic primitive representations
- Incremental learning shows how a system can learn a complex system by having better initial data:
  - Initially impoverished memory provides a natural filter for complex structures early in learning so the network can learn the basic forms of linguistic regularities
  - As the memory is expanded, the network can use what it knows to handle increasingly complex inputs
  - Noise, present in the early data, tends to keep the network in a state of flux, helping it to avoid committing to false generalisations

Summary of SRNs …

- Finding structure in time/sequences:
  - Learns dependencies spanning more than a single transition
  - Learns dependencies of variable length
  - Learns to make partial predictions from structure input
    - Prediction of **consonants**, or particular lexical **classes**
- Learning from various input encodings:
  - Localist encoding: XOR and 1 bit per word
  - Distributed:
    - Structured: letter sequences where consonants have a distinguished feature
    - Random: words mapped to random 5 bit sequence
- Learns both general categories (types) and specific behaviours (tokens) based purely on distributional evidence
- What are the limitations of SRNs
  - Do they simply learn co-occurrences and contingent probabilities?
  - Can they learn more complex aspects of linguistic structure?

Properties of Connectionist Networks

- Learning
  - There is usually no predetermined (innate) knowledge of language, but ...
    - Input/output representation are often specified
    - The architecture of the network may be “suited” to a particular task
    - The learning mechanism and parameters provide degrees of freedom
  - Learning takes place in direct response to experience
    - Structure of the training environment is often important (e.g order and frequency of inputs)
- Generalisation
  - Networks are able to learn generalisations not just by rote
  - More efficient representation of information
  - Novel inputs can be processed
- Representation
  - Learned automatically, and typically distributed

Properties continued

- Rules versus exceptions
  - Single mechanism to explain both general rules and also exceptions
- Graded:
  - Can often give a useful output to new, partial, noisy input
  - Can yield non-deterministic outputs
  - Damage is distributed, and some performance is still possible:
    - Modelling of brain damage and neurological disorders in possible
- Frequency effects
  - Model response time behaviours where high frequency inputs are recognised faster than low frequency ones

Resources

- Main texts:
  - MacLeod, Rolls & Plunkett (1998). *Introduction to Connectionist Modelling of Cognitive Processes*. Oxford University Press.
  - Plunkett & Elman (1997). *Exercises in rethinking innateness*. MIT Press.
- Supplementary reading:
  - Elman, Bates, Johnson, Karmiloff-Smith, Parisi & Plunkett (1996). *Rethinking innateness.* MIT Press.
  - Manning and Schütze (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, MA: MIT Press.
- Selected Articles:
  - Elman (1990). Finding structure in time. *Cognitive Science*, 14, 179-211.
  - Elman (1991). Distributed Representations, simple recurrent networks, and grammatical structure. *Machine Learning*.
  - Elman (1993). Learning and development in neural networks: the importance of starting small. *Cognition*, 48:71-99.