Outline for Today

- A case study: Time delay neural nets
- Recurrent networks
- Dynamically changing network structure
- Alternative error functions

Case Study: Speech Recognition

[Lang, Waibel, Hinton, 1990]

- Task: Recognize b,d,e,v from different speakers
- Preprocess signal to extract 16 freqency bands over time
- Network input: 16 freqs x 12 time segments
- Network output: one each for b,d,e,v
- 91% Recognition accuracy (comparable to best achievable)

Design Choices for Neural Net Speech Recognition

1. Which frequency bands?
   - 16 bands. allocate linearly? logarithmically? ...
2. How to map energy level to input value?
   - linearly? log? square? clip values?
3. What temporal resolution
   - 24msec, 12, 6, 3?

Try alternatives, using 2-layer network

Performance varied between 82% - 86%

Resulting choices:

- Mel scale assignment of frequencies to 16 bands
- Scale inputs to (0, 1) by
  - Mapping -5 dB to 0, +105dB to 1, clipping other
  - Using squares of input values
- Temporal resolution 12 msec

Design Choices: Alternative Net Architectures

1. Fully connected, feed forward, 2 layer

2. Fully connected, feed forward, 3 layer

3. Allocate 3 hidden units to each 3-step time window

4. As above, but tie together hidden unit weights
   - Effectively only 3 hidden units, duplicated for 10 input windows

5. As above, but allocate outputs to 5-step time windows and tie together weights

6. As above, but 6 hidden units with 4-step time windows, 4 outputs with 4-step time windows

Performance varied between 86.9% and 91.4%

Final choice: 6

Performance comparable to best known methods

Case study: Key Ideas

1. Sparsely connected units
2. Tying weights together
3. Temporally replicated “time delay” architecture
4. Train/test scheme
5. Many simple experiments to choose representation, parameters

Recurrent Networks

The issue: networks with cycles

- Useful for time series data
- Provides a learnable memory!

One special case: output $O_i(t)$ feeds back to next input $C_i(t+1)$

- *Context* units:

$$C_i(t+1) = \alpha C_i(t) + O_i(t)$$

$$C_i(t+1) = O_i(t) + \alpha O_i(t-1) + \alpha^2 O_i(t-2) + ...$$

- fix $\alpha < 1$ to set time constant
- Can train using Backprop rule

Backpropagation Through Time

Key idea: handle general recurrent networks by unfolding them

- Allows simple training rule
- Update using Backprop, with weights tied together

Dynamically Altering Net Structure

The issue: Might not know optimal structure in advance

- Completely connected?
- How many layers?
- How wide are hidden layer(s)?

Approach 1: Start big, then shrink

- Use large initial network
- Remove edges found to have lasting low weights

Sometimes this improves generalization

Dynamically Altering Net Structure

Approach 2: Start small, then grow [Fahlman 92] Cascade Correlation

- Start with one unit
- Train for a while, then freeze unit weights permanenently
- Add new unit, allowing all existing units to be its inputs

Alternative Error Functions

- Add penalty for large weights:

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{p \in P} \sum_{k \in outputs} (t_{k,p} - o_{k,p})^2 + \gamma \sum_{i,j} w_{i,j}^2$$

- Minimize error in learned function value, plus its slope (if you know target slopes)

$$\begin{aligned} E[\vec{w}] \quad \equiv \quad & \frac{1}{2} \sum_{p \in P} \sum_{k \in outputs} (t_{k,p} - o_{k,p})^2 \\ & + \frac{1}{2} \sum_{p \in P} \sum_{k \in outputs} \left(\frac{\partial H}{\partial \vec{x}}_{k,p} - \frac{\partial \widehat{H}}{\partial \vec{x}}_{k,p} \right)^2 \end{aligned}$$