Receptive fields

spatially localized learning & radial basis functions

Radial Basis Functions

- In the sigmoidal activation functions we've seen before, the "basis" of activation is the weighted sum of the inputs
- In self-organizing maps, the "winner" is the node whose weight vector is least distant from the input
- Now consider nodes whose activation is a direct function of radial distance between the weight vector and the input...

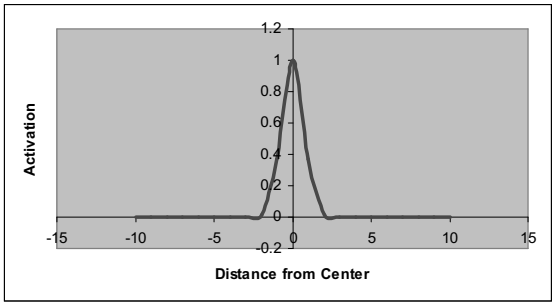
Consider Gaussian Units

- These units make literal "activation bumps" of sets of sigmoids

$$f\left(\vec{x}\right) = ae^{-\sum_i \left[\frac{(x_i - c_i)}{\sigma_i}\right]^2}$$

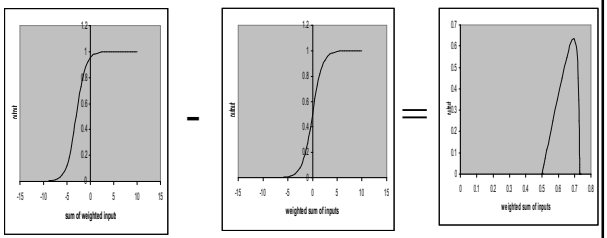
- where c is a "center", and s is a "scale factor"
- These are the node's parameters ("weights")

Pictorially



Recall...

- Combinations of sigmoids...



Somewhat similar effect, but over a different "basis"

Training

- Clearly, these "radial basis function" or "Gaussian" networks can be updated via some sort of delta rule, by calculating:

$$\frac{\partial E}{\partial c_i}, \frac{\partial E}{\partial \sigma_i}$$

and taking negative gradient steps

Comparison

- RBFs are sometimes superior to sigmoids...

Desired Sigmoids RBFs

Using *Spatial Localization* for Speedup

- In RBF nets, only nodes (and related weights) whose centers are near the current inputs are (substantially) involved in output, and update
- Consider thresholding the activations
- The area around the nodes center which gives non-zero output is that node's "receptive field"
- Only nodes that are near the input require calculations and updating!

Downside of Spatial Localization

- Note that with limited receptive fields, the network can work itself into situations where no node responds, or gets updated, for certain inputs.
- This causes delta rule updates to fail
- "Clustering" algorithms and other training techniques have been suggested for nets with limited receptive fields

Spatially Localized, Online Learning

- Spatially localized learning has obvious advantages in computation and update speed
- It also has advantages in avoiding oversimplification and "forgetting" in certain situations

Forgetting Online

- Let's say we have an online application
- and the system "dwells" in a region of the state (*x)* space for a long time
- A regular, fully connected, sigmoid net will learn that region
- But if it moves onto another region, it will relearn, and forget all previous learning

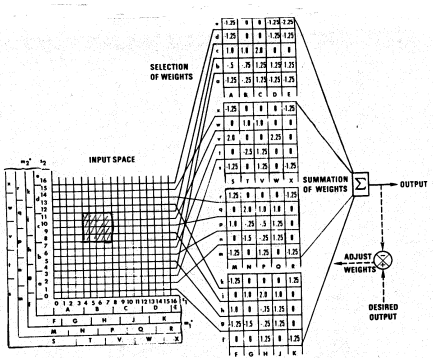
Spatial Localization Online

- If some limited receptive field scheme (e.g., RBF) is employed
- Not every node is updated at every time step
- Nodes can "specialize" in regions of the input space
- Avoiding forgetting

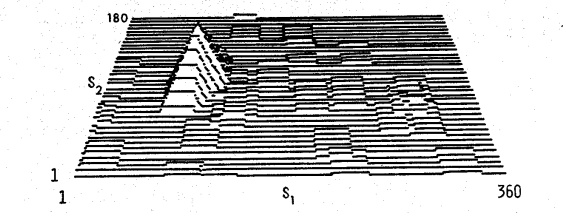
Another aspect of receptive fields

- Consider CMAC (Albus)
- Cerebellar Model Arithmetic Computer (1975), or
- Cerebellar Model Articulation Controller (1979)
- This is a biological model that concentrates on overlapping receptive fields...

CMAC

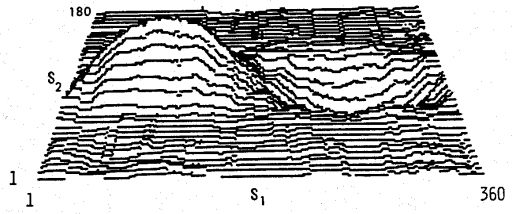


After one training step

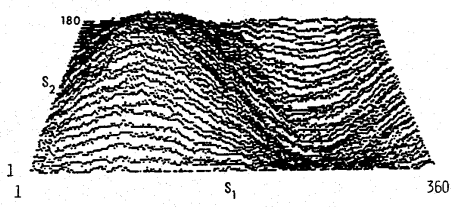


Learning sin(S1)sin(S2)

After 16 training steps



After 175 training steps



CMAC seems entirely linear....

- But is is capable of nonlinear mappings due to its use overlapping receptive fields
- This is a form of switching nonlinearity
- It yields a output surface that varies relatively smoothely
- Weights are updated via a delta-rule-like procedure
- It involves only a single layer of weights

CMAC advantages

- It trains relatively quickly
- It requires few weight updates per training cycle
- It is "spatially localized"
- It requires less memory than a one-to-one mapping

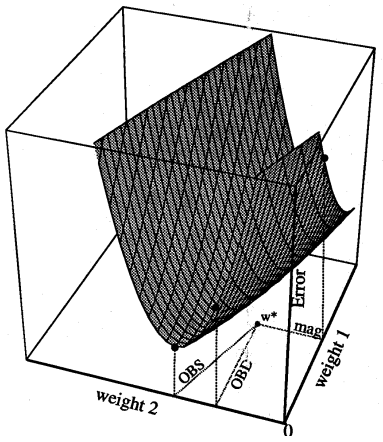
CMAC difficulties...

- For large input spaces, it requires a large amount of memory
- Albus uses "hash codings" to reduce CMAC memory requirements
- In a more general context, it would be nice to use as few overlapping receptive fields as possible for a desired level of accuarcy
- Methods for determining this connectivity are still being developed

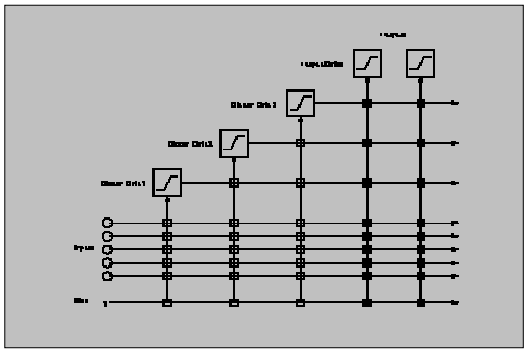
On another topic:

- Consider "pruning" away weights in a BP network
- Several methods have been suggested:
  - Magnitude based (Hertz, et. al 1991)
  - Optimal Brain Damage (Le Cun, et. al, 1990)
  - Optimal Brain Surgery (Hassibi & Stork,1992)
- The first method is not generally effective, and the later two are somewhat non-neural

Pruning



Adding Nodes Incrementally: Cascade Correlation (Fahlman, 91)



Cascade Correlation Procedure

- Start with no hidden layer nodes, and repeat
  - Train output weights to minimize error
  - Add node(s) with inputs from all existing nodes (cascaded)
  - Train new node input weights until output of the new node correlates to existing error
  - Fully connect output of the new node
  - Continue

Why correlate new nodes to existing error?

- Because that means the “receptive field” of the new node is aligned with the area of the input space that is causing most error
- Then, adjustments of this node will be able to correct existing error

Note...

- Each of the things we’ve covered:
  - sigmoidal nodes, linear nodes, BP, SOM, RBF, CMAC, CC
- are all design tools, not necessarily end products