CogSci 109: Lecture 18

Tues Nov 28, 2006

Resolving Training Issues in artificial neural networks, hopfield networks, associative memory

Outline for today

- Announcements
- Review of last time - training issues in ANN’s
  - Avoiding overfitting
    - Regularization
    - Early stopping
    - Bayesian regularization
- Improvements on weight update algorithms
  - Conjugate gradient
- Networks, a contextual example
  - Hopfield networks and associative memory
  - Hebbian learning
- Quick mentioning of lyapunov functions

Announcements

- Final hw 6 part of final
  - Due date
  - General final exam structure
    - Bring
      - Scantron
      - Calculator
      - Pencils
      - BYOB ('Bring your own brain!!!')
      - Up to 3 handwritten note sheets (so you have 6 sides of pages, 8.5x11)!!!
- Grades dealing with issues

Training issues mentioned last time

- Last time we defined, and mentioned some strategies for
  - **Overfitting**
  - **Generalization**
  - We want to reduce overfitting and increase generalization of our fits

Techniques to Prevent Overfitting

- Regularization
  - Reduction of hidden units
    - Only fit simpler functions
  - Weight decay
- Early stopping
  - Using validation sets
- Bayesian regularization
  - (see the MacKay Book)

Technique 1: Reduce number of layers to prevent overfitting

- *Note: Remember that overfitting is a problem when fitting many parameters to small amounts of data*
  - *Infinite data would be then no problem*
- Simplify the function you are fitting by reducing the number of network hidden layers - similar to using a lower degree polynomial to fit data
  - Limits the capability of your network
- But ahead of time we may not know the complexity of the function we want to fit, so how do we deal with this?

Technique 2: Regularization to prevent overfitting

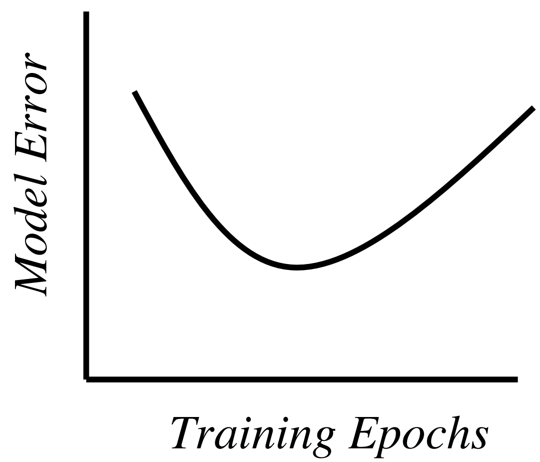
- ***Regularization*** - adding a penalty to the usual error function to encourage smoothness

$$E_{new} = E + \nu * \omega$$

- Here ν is the regularization parameter and ω is the smoothness penalty
- Weight decay sets $\omega = \frac{1}{2}\sum_i w_i^2$
  - Note that when you then take the partial derivative of E_{new} with respect to a weight the update rule will now include a term that is -w_i.
  - This will encourage the weights to decay to zero (hence the name)

Technique 3: Early stopping to prevent overfitting

- Start the weights very small
  - Then the neural network starts by behaving fairly linearly
  - The weights gradually increase to handle nonlinearities
- Split the data into a **validation set** and a **training set**
  - Use the *training set* to adjust the weights
  - Use the *validation set* to compute model error
  - As the fit improves the error will decrease, when the error starts to increase again, you are fitting the noise in the training set



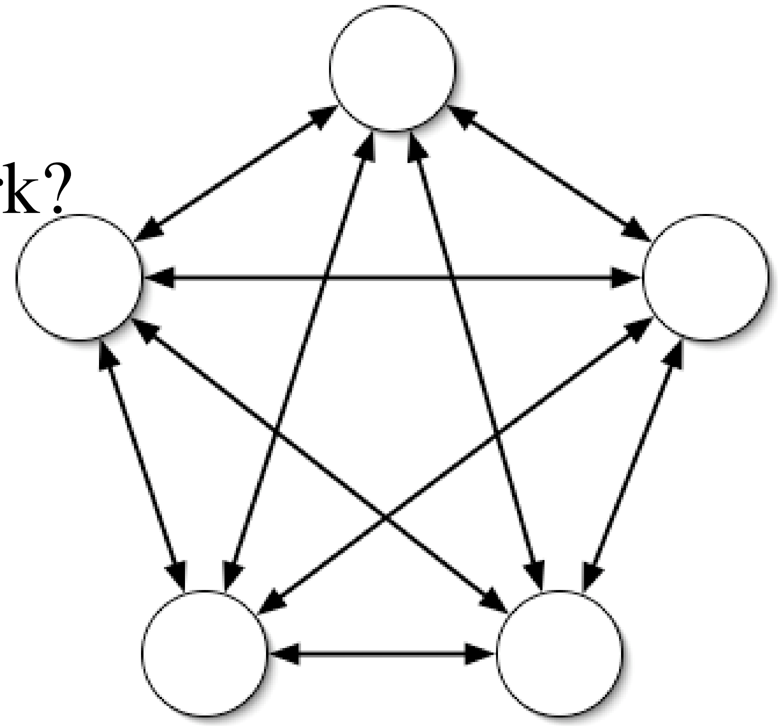
Technique 4: Bayesian regularization to prevent overfitting

- The Bayesian neural network formalism of David MacKay and Radford Neal, considers neural networks not as single networks but as distributions over weights (and biases)
- The output of a trained network is thus not the result of applying one set of weights but an average over the outputs from the distribution.
- This can be computationally expensive but MacKay and Neal have developed approximations and the approach leads to automatic regularization that is very effective.

More training issues mentioned last time

- Improvements on gradient descent
  - Gradient descent with momentum
  - *Conjugate gradient*
  - Variable learning rate
  - For nonquadratic functions, minimization (ie Nelder Mead, golden section line search, Brent's method, etc - See numerical methods book)
    - Demos:
      - nnd12sd1
      - nnd12sd2
      - nnd12mo
      - nnd12vl
      - nnd12ls
      - nnd12cg

A Network example - Associative Memory



- Associative memory sample
  - (Yellow)--(banana smell)
- What is a binary Hopfield network?
  - Weights are constrained to be
    - Symmetric $w_{kp} = w_{pk}$
    - Bidirectional
    - No self connections (w_ii = 0)
  - Activity rule

$$x(a) = \Theta(a) \equiv \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

    - We need to specify the order of updates as either
      - **Synchronous** $a_k = \sum_j w_{kj} x_j$, $x_k = \Theta(a_k)$
      - **Asynchronous** - each neuron sequentially (either fixed or random order) computes its activation then updates its output state and weights

Binary Network learning rule

- Learning rule
  - The problem - make a set of memories $\{\mathbf{x}^{(n)}\}$stable states of the network's activity rule
    - Each memory is a binary pattern $x_i \in \{-1, 1\}$
    - Setting the weights is done according to Hebb's rule:

$$w_{ij} = \eta \sum_n x_i^{(n)} x_j^{(n)}$$

    - We may set eta to prevent a particular weight from growing with N: $\eta = 1/N$

Associative memory example

Desired memories:

moscow------russia
lima----------peru
london-----england
tokyo--------japan
edinburgh-scotland
ottawa------canada
oslo--------norway
stockholm---sweden
paris-------france

- Pattern completion

```
moscow---::::::::: ⟹ moscow------russia
:::::::::--canada  ⟹ ottawa------canada
```

- Error correction

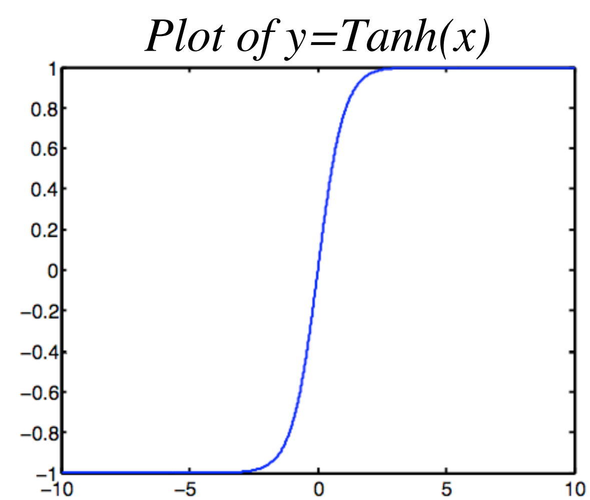
```
otowa-------canada ⟹ ottawa------canada
egindurrh-sxotland ⟹ edinburgh-scotland
```

Continuous form of the Hopfield network

- Similar rules, but instead of binary states, we have continuous states from (-1,1)

$$a_i = \sum_j w_{ij} x_j$$

$$x_i = \tanh(a_i)$$



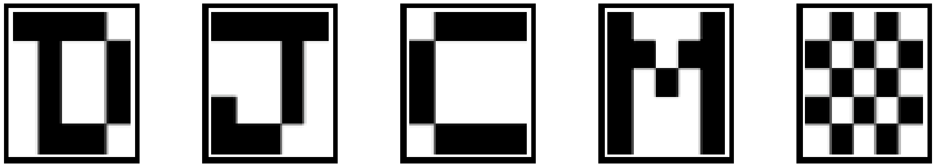
- Eta becomes more important

Stability of memories

- Lyapunov functions
  - If you can show that a lypapunov function exists for an ANN, then it's dynamics converge rather than diverge
  - Look up lyapunov functions for more info, some mentioning next lecture

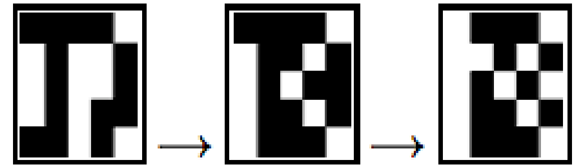
Brain damage (p. 511 in MacKay) - delete 26 weights, still converges

(a)

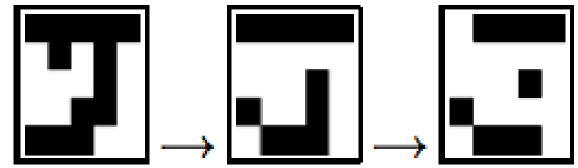


.	-1	1	-1	1	x	x	-3	3	x	x	-1	1	-1	x	-1	1	-3	x	1	3	-1	1	x	-1
-1	.	3	5	-1	-1	-3	-1	-3	-1	-3	1	x	1	-3	1	-1	-1	-1	-1	-3	5	3	3	-3
1	3	.	3	1	-3	-1	x	-1	-3	-1	-1	x	-1	-1	-1	1	-3	1	-3	-1	3	5	1	-1
-1	5	3	.	-1	-1	-3	-1	-3	-1	-3	1	-5	1	-3	1	-1	-1	-1	-1	-3	5	x	3	-3
1	-1	1	-1	.	1	-1	-3	x	x	3	-5	1	-1	-1	3	x	-3	1	-3	3	-1	1	-3	3
x	-1	-3	-1	1	.	-1	1	-1	1	3	-1	1	-1	-1	3	-3	1	x	1	x	-1	-3	1	3
x	-3	-1	-3	-1	-1	.	-1	1	3	1	1	3	-3	5	-3	3	-1	-1	x	1	-3	-1	-1	1
-3	-1	x	-1	-3	1	-1	.	-1	1	-1	3	1	x	-1	-1	1	5	1	1	-1	x	-3	1	-1
3	-3	-1	-3	x	-1	1	-1	.	-1	1	-3	3	1	1	1	-1	-1	3	-1	5	-3	-1	x	1
x	-1	-3	-1	x	1	3	1	-1	.	-1	3	1	-1	3	-1	x	1	-3	5	-1	-1	-3	1	-1
x	-3	-1	-3	3	3	1	-1	1	-1	.	-3	3	-3	1	1	-1	-1	-1	-1	1	-3	-1	-1	5
-1	1	-1	1	-5	-1	1	3	-3	3	-3	.	-1	1	1	-3	3	x	-1	3	-3	1	-1	3	-3
1	x	x	-5	1	1	3	1	3	1	3	-1	.	-1	3	-1	1	1	1	1	3	-5	-3	-3	3
-1	1	-1	1	-1	-1	-3	x	1	-1	-3	1	-1	.	x	1	-1	3	3	-1	1	1	-1	-1	-3
x	-3	-1	-3	-1	-1	5	-1	1	3	1	1	3	x	.	x	3	-1	-1	3	1	-3	-1	-1	1
-1	1	-1	1	3	3	-3	-1	1	-1	1	-3	-1	1	x	.	-5	-1	-1	-1	1	1	-1	-1	1
1	-1	1	-1	x	-3	3	1	-1	x	-1	3	1	-1	3	-5	.	1	1	1	-1	-1	1	1	-1
-3	-1	-3	-1	-3	1	-1	5	-1	1	-1	x	1	3	-1	-1	1	.	1	1	-1	-1	-3	1	-1
x	-1	1	-1	1	x	-1	1	3	-3	-1	-1	1	3	-1	-1	1	1	.	-3	3	-1	1	-3	-1
1	-1	-3	-1	-3	1	x	1	-1	5	-1	3	1	-1	3	-1	1	1	-3	.	x	-1	-3	1	-1
3	-3	-1	-3	3	x	1	-1	5	-1	1	-3	3	1	1	1	-1	-1	3	x	.	-3	-1	-5	1
-1	5	3	5	-1	-1	-3	x	-3	-1	-3	1	-5	1	-3	1	-1	-1	-1	-1	-3	.	3	x	-3
1	3	5	x	1	-3	-1	-3	-1	-3	-1	-1	-3	-1	-1	-1	1	-3	1	-3	-1	3	.	1	-1
x	3	1	3	-3	1	-1	1	x	1	-1	3	-3	-1	-1	-1	1	1	-3	1	-5	x	1	.	-1
-1	-3	-1	-3	3	3	1	-1	1	-1	5	-3	3	-3	1	1	-1	-1	-1	-1	1	-3	-1	-1	.

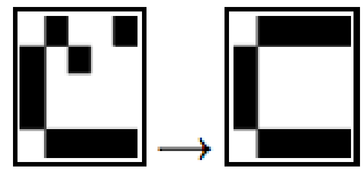
(b)



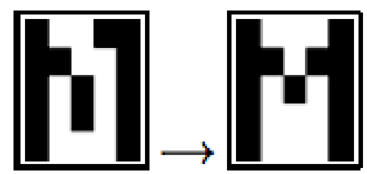
(c)



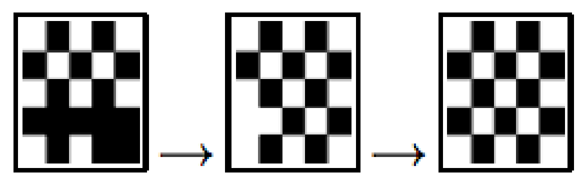
(d)



(e)



(f)



Failures of ANN's

- Stability of memories is an issue to be considered
- For failure mode analysis (where hopfield networks fail to correctly restore memories), see MacKay Chapter 42