

# CS545: Artificial Neural Networks

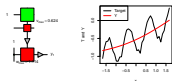
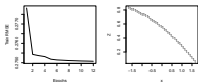
Chuck Anderson

Department of Computer Science  
 Colorado State University

Fall, 2009

## One Hidden Unit Can Form Linear Model

- Recall that output of neural network is  $\mathbf{Y} = \tilde{h}(\tilde{\mathbf{X}}\mathbf{V})\mathbf{W}$
- If  $h = \tanh$ , then when  $\mathbf{x}^T\mathbf{V}$  is close to zero for all  $\mathbf{x}$  the value of  $h$  varies mostly linearly with  $\mathbf{x}$ .
- Example, using Scaled Conjugate Gradient to train:



- Potential complexity of learned model increases with number of hidden units.

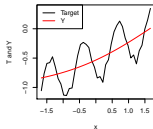
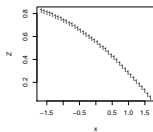
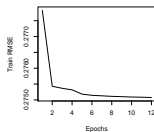
## Outline

Number of Hidden Units

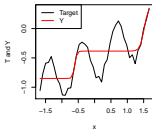
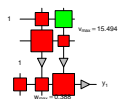
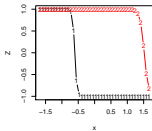
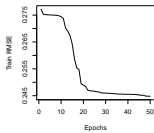
Limiting Overfitting

Number of Hidden Units  
 Weight Magnitude Penalty  
 Early Stopping

## One Hidden Unit, Again



## Two Hidden Units



6 / 18

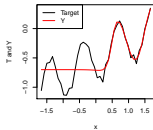
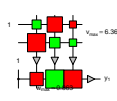
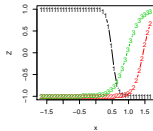
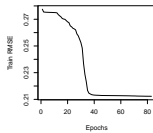
CS545: Artificial Neural Networks  
Chuck Anderson

Number of Hidden Units

Limiting Overfitting

Number of Hidden Units  
Weight Magnitude  
Penalty  
Early Stopping

## Three Hidden Units



6 / 18

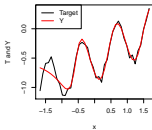
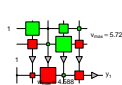
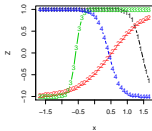
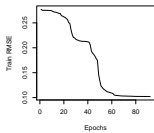
CS545: Artificial Neural Networks  
Chuck Anderson

Number of Hidden Units

Limiting Overfitting

Number of Hidden Units  
Weight Magnitude  
Penalty  
Early Stopping

## Four Hidden Units



7 / 18

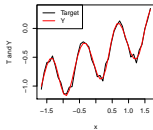
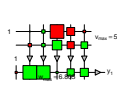
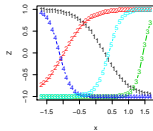
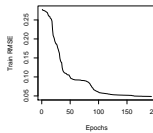
CS545: Artificial Neural Networks  
Chuck Anderson

Number of Hidden Units

Limiting Overfitting

Number of Hidden Units  
Weight Magnitude  
Penalty  
Early Stopping

## Five Hidden Units



6 / 18

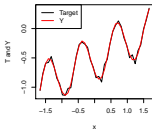
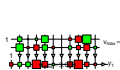
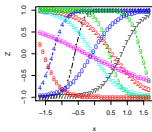
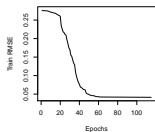
CS545: Artificial Neural Networks  
Chuck Anderson

Number of Hidden Units

Limiting Overfitting

Number of Hidden Units  
Weight Magnitude  
Penalty  
Early Stopping

## Ten Hidden Units



## Limiting the Possible Amount of Overfitting

- Nonlinear function approximators can overfit training data. What are some ways to limit the possibility of overfitting?
  - 1 Limit the number of hidden units. How many is too many?
  - 2 Limit the magnitude of the weights. Like ridge regression. Which weights and by how much?
  - 3 Limit the number of epochs. When should we stop?

### Number of Hidden Units

#### Limiting Overfitting

Number of Hidden Units  
Weight Magnitude Penalty  
Early Stopping

### Number of Hidden Units

#### Limiting Overfitting

Number of Hidden Units  
Weight Magnitude Penalty  
Early Stopping

## Outline

### Number of Hidden Units

#### Limiting Overfitting

Number of Hidden Units  
Weight Magnitude Penalty  
Early Stopping

### Number of Hidden Units

#### Limiting Overfitting

Number of Hidden Units  
Weight Magnitude Penalty  
Early Stopping

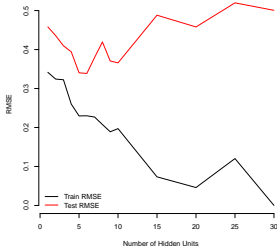
### Number of Hidden Units

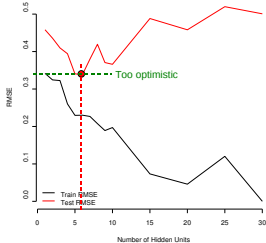
#### Limiting Overfitting

Number of Hidden Units  
Weight Magnitude Penalty  
Early Stopping

## Limit the Number of Hidden Units

- How many is too many?
- When RMS error on untrained data is lowest.





- Do not report RMS error of 0.33 as expected error on new data.
- No guarantee that new data will be like test data. What can we do?

## Limit the Weight Magnitude

- In ridge regression, we added penalty of  $\lambda \mathbf{w}^T \mathbf{w}$ , the sum of squared weight magnitudes. Can we do this for neural networks?
- Sure.

$$\mathbf{w} = (w_{0,1}, \dots, w_{D,M}, w_{0,1}, \dots, w_{M,K})$$

$$E = \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k=1}^K (y_{n,k} - t_{n,k})^2 + \lambda (\mathbf{w}^T \mathbf{w})$$

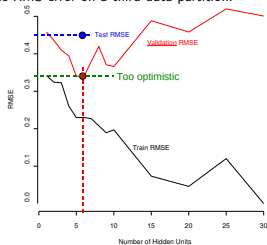
- Update equations become

$$\mathbf{V} \leftarrow \mathbf{V} - \rho_o \left( \frac{1}{N} \frac{1}{K} \tilde{\mathbf{X}}^T (\mathbf{Y} - \mathbf{T}) \tilde{\mathbf{W}}^T \cdot (1 - \mathbf{Z}^2) \right) + \lambda \mathbf{V}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \rho_h \left( \frac{1}{N} \frac{1}{K} \tilde{\mathbf{Z}}^T (\mathbf{Y} - \mathbf{T}) + \lambda \mathbf{W} \right)$$

## Three Data Partitions

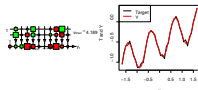
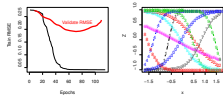
- Calculate RMS error on a third data partition.



- Train: optimize parameters. (60%)  
Validation: select best model. (20%)  
Test: predict future error (20%)

## Early "Stopping"

- During training, weight magnitudes increase, making model more complex. Can we use this to find best model?
- Monitor RMS error on validation set while training.



- Use weights when validation RMS error is lowest.