ROB 537:
Learning-Based Control

Week 1, Lecture 2

Multi-Layered Feed Forward Neural Networks

Announcements:

HW 1 Due on 10/2

Oregon State University

Single Layer Feed Forward Networks

- Error on pattern n: $e^n = t^n - y^n$
- Mean Square Error: $E = \frac{1}{2}\sum_{n=1}^{N}(e^n)^2$
- Gradient descent: $\Delta w = -\eta \frac{\partial E}{\partial w}$

We'll drop the *n* superscript from *x* , *y*, *t*, *e* for simplicity

Oregon State University

Single Layer Feed Forward Networks

- Differentiate error with respect to weights

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial e_i} \frac{\partial e_i}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_{i,j}}$$

where $z_i = \sum_j w_{i,j} x_j$

Oregon State University

Single Layer Feed Forward Networks

- Compute each of the partials:
- First term: $\frac{\partial E}{\partial e_i} = \frac{\partial \left(\frac{1}{2} e_i^2 \right)}{\partial e_i} = e_i$
- Second term: $\frac{\partial e_i}{\partial y_i} = \frac{\partial (t_i - y_i)}{\partial y_i} = -1$
- Fourth term: $\frac{\partial x_j}{\partial w_{i,j}} = \frac{\partial \left(\sum_j w_{i,j} x_j \right)}{\partial w_{i,j}} = x_j$

Oregon State University

Single Layer Feed Forward Networks

- Third term: $$\frac{\partial y_i}{\partial z_i} = \frac{\partial\left(\frac{1}{1+e^{-z_i}}\right)}{\partial z_i}$$

Oregon State University

Single Layer Feed Forward Networks

- Third term: $$\frac{\partial y_i}{\partial z_i} = \frac{\partial\left(\frac{1}{1+e^{-z_i}}\right)}{\partial h_i}$$

$$= -\left(\frac{1}{1+e^{-z_i}}\right)^2 e^{-z_i}(-1)$$

$$= \left(\frac{1}{1+e^{-z_i}}\right)\left(\frac{e^{-z_i}}{1+e^{-z_i}}\right)$$

$$= \left(\frac{1}{1+e^{-z_i}}\right)\left(\frac{1+e^{-z_i}-1}{1+e^{-z_i}}\right)$$

$$= \left(\frac{1}{1+e^{-z_i}}\right)\left(\frac{1+e^{-z_i}}{1+e^{-z_i}} - \frac{1}{1+e^{-z_i}}\right)$$

$$= y_i(1-y_i)$$

Oregon State University

Single Layer Feed Forward Networks

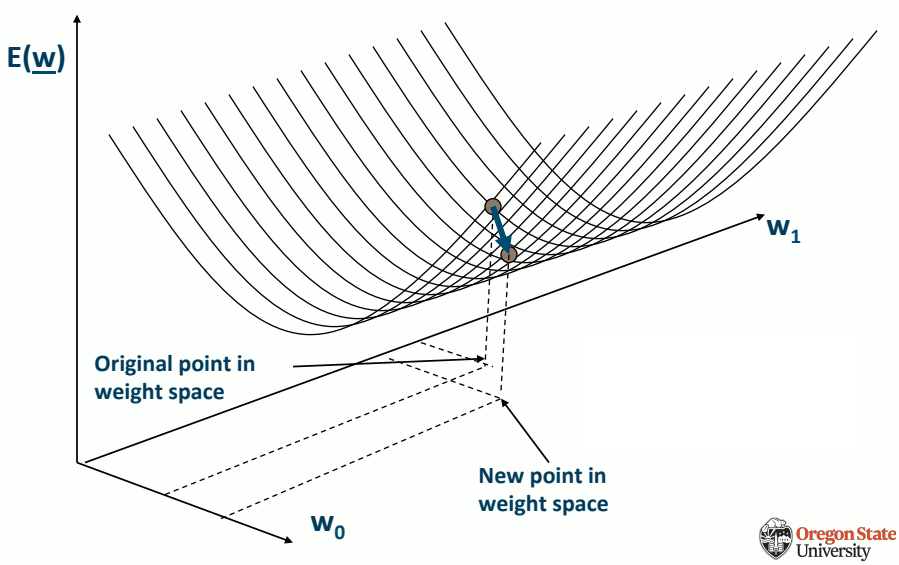
- Sigmoid activation function: $$f(a) = \frac{1}{1+e^{-a}}$$

- Gradient descent: $$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial e_i} \frac{\partial e_i}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_{i,j}}$$
$$= e_i \ (-1) \ y_i(1-y_i) \ (x_j)$$

- Weight update: $$\boxed{\Delta w_{i,j} = -\eta \frac{\partial E}{\partial w_{i,j}} = \eta \ e_i y_i (1-y_i) x_j}$$

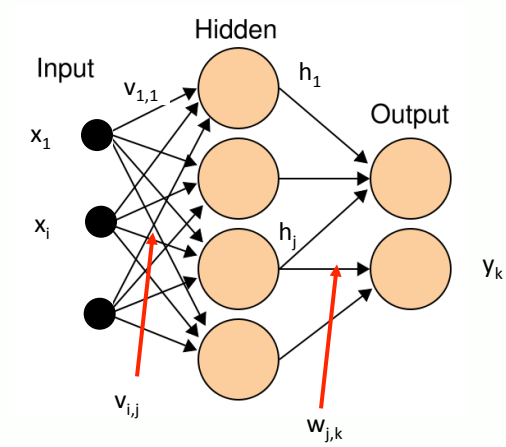
Oregon State University

Illustration of Gradient Descent



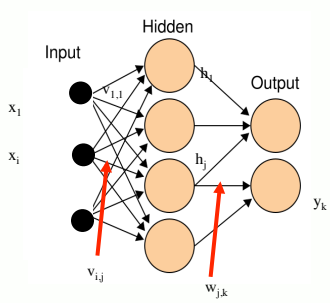
Oregon State University

Multi Layer Feed Forward Networks



Oregon State University

Multi Layer Feed Forward Networks



$$y_k = f\left(\sum_j w_{j,k} h_j + w_{0,k}\right)$$

$$= f\left(\sum_j w_{j,k} \cdot f\left(\sum_i v_{i,j} x_i + v_{0,j}\right) + w_{0,k}\right)$$

$$f(a) = \frac{1}{1+e^{-a}}$$

Oregon State University

Weight updates

- Derivative of error wrt weight j,k:

$$\begin{aligned}\frac{\partial E}{\partial w_{j,k}} &= e_k^n \frac{\partial e_k^n}{\partial w_{j,k}} \\ &= -e_k^n \frac{\partial f_k^n}{\partial w_{j,k}} \\ &= -e_k^n f\left(\sum_j w_{j,k} h_j^n\right)\left(1 - f\left(\sum_j w_{j,k} h_j^n\right)\right) h_j \\ &= -e_k^n y_k^n (1 - y_k^n) h_j\end{aligned}$$

Weight updates

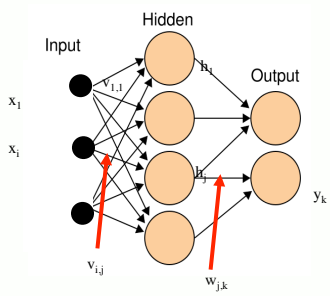
- Derivative of error wrt weight j,k:

$$\begin{aligned}\frac{\partial E}{\partial w_{j,k}} &= e_k^n \frac{\partial e_k^n}{\partial w_{j,k}} \\ &= -e_k^n \frac{\partial f_k^n}{\partial w_{j,k}} \\ &= -e_k^n f\left(\sum_j w_{j,k} h_j^n\right)\left(1 - f\left(\sum_j w_{j,k} h_j^n\right)\right) h_j \\ &= -e_k^n y_k^n (1 - y_k^n) h_j\end{aligned}$$

- Updating hidden-output layer weights: $\Delta w_{j,k} = -\eta \frac{\partial E}{\partial w_{j,k}} = \eta \delta_k h_j$

- Hidden-output layer deltas: $\delta_k = e_k^n y_k^n (1 - y_k^n)$

Multi Layer Feed Forward Networks



$$h_j = f\left(\sum_i v_{i,j} x_i + v_{0,j}\right)$$

What are the errors for the hidden layer?

We don't know the targets.

Now what?

Oregon State University

Error Backpropagation

- Updating input-hidden layer weights:

$$\Delta v_{i,j} = \eta \delta_j x_i$$

- Delta:

$$\delta_j = e_j^n f\left(\sum_i v_{i,j} x_i^n\right)\left(1 - f\left(\sum_i v_{i,j} x_i^n\right)\right)$$
$$= e_j^n h_j^n (1 - h_j^n)$$

Errors for the hidden layer:

Backpropagated deltas from output layer

Oregon State University

Backpropagation Summary:

- For sigmoidal activation functions, update any weight connecting "input" *i* to "output" *j*:

$$\Delta w_{i,j} = \eta \delta_j x_i$$

- Deltas given by:
  - For output layer $$\delta_j = e_j y_j (1 - y_j)$$
  - For hidden layer $$\delta_j = \sum_k w_{j,k} \delta_k \cdot h_j (1 - h_j)$$

Oregon State University

Backpropagation Algorithm

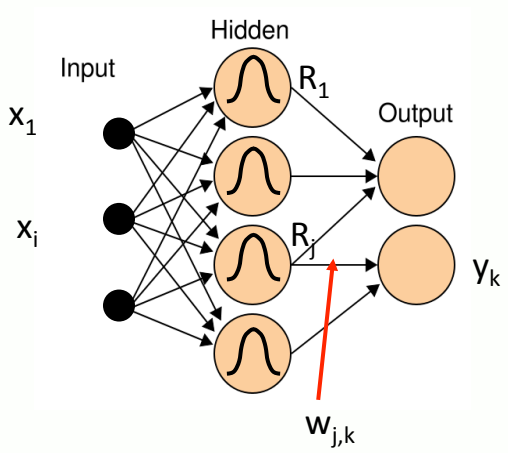
- For each epoch:
  - Present pattern x to network
    - Propagate signal forward:
      - Compute hidden unit values
      - Compute output value
    - Find error
      - Compute output layer deltas
      - Compute hidden layer deltas
    - Compute gradient for each weight
    - Update each weight
  - Present next pattern
- Repeat this process until MSE is satisfactory

Oregon State University

Feed Forward Neural Net Recap

- A multi-layered FFNN with sigmoidal activation functions is also called a multi-layered Perceptron.
- Key characteristic:
  - Global processing
  - All hidden units react to all inputs
- What if we want local processing?
  - Some hidden units react to some inputs?

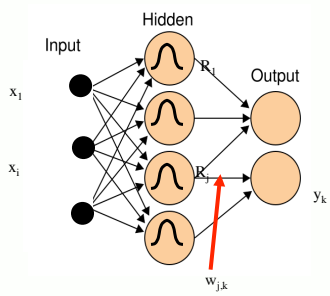
Radial Basis Function Networks

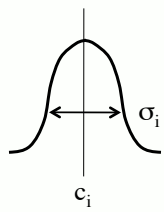


Key RBF differences:

- Local activation
- Linear output layer recommended
- Euclidean norm activation
- All hidden units are different functions
- One "hidden" layer

Radial Basis Function Networks





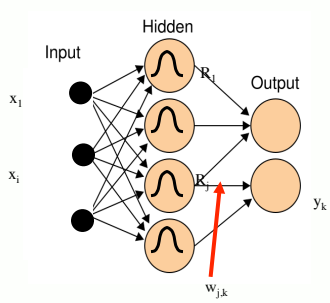
- c_j is the "center" of the jth radial basis function

  $c_j = \{c_{j,1}, \ldots, c_{j,I}, \ldots, c_{j,N}\}$

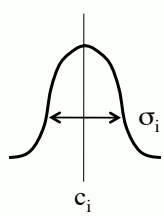
- σ_j is the "radius" of the jth radial basis function

Oregon State University

Radial Basis Function Networks

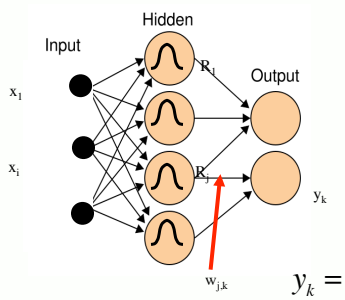


$$R_j(x) = \exp\left(-\frac{|x - c_j|^2}{2(\sigma_j)^2}\right)$$

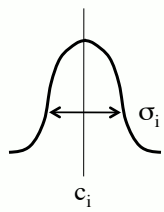


Oregon State University

Radial Basis Function Networks



$$R_j(x) = \exp\left(-\frac{|x - c_j|^2}{2(\sigma_j)^2}\right)$$



$$y_k = f\left(\sum_j w_{j,k} R_j + w_{0,k}\right)$$

$$= f\left(\sum_j w_{j,k} \cdot \exp\left(-\frac{|x - c_j|^2}{2(\sigma_j)^2}\right) + w_{0,k}\right)$$

Oregon State University

RBF Center Updates

- For single output (y) linear output layer:
- Write set of partials: $$\frac{\partial E}{\partial c_{j,k}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial R_j}\frac{\partial R_j}{\partial c_{j,k}}$$
- What is $\dfrac{\partial R_j}{\partial c_{j,k}}$?

Oregon State University

RBF Center Updates

- For quadratic distance:

$$\left|x - c_j\right|^2 = (x - c_{j,1})^2 + \cdots + (x - c_{j,i})^2 + \cdots + (x - c_{j,N})^2$$

- Center updates:

$$\frac{\partial R_j}{\partial c_{j,k}} = \frac{\partial \exp\left(-\frac{\left|x - c_j\right|^2}{2(\sigma_j)^2}\right)}{\partial c_{j,k}}$$

$$= \exp\left(-\frac{\left|x - c_j\right|^2}{2(\sigma_j)^2}\right)(-2)(-\frac{1}{2(\sigma_j)^2})(x_k - c_{j,k})$$

$$= R_j \frac{(x_k - c_{j,k})}{(\sigma_j)^2}$$

Oregon State University

RBF Center Updates

- For single output (y) linear output layer:

$$\frac{\partial E}{\partial c_{j,k}} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial y}\frac{\partial y}{\partial R_j}\frac{\partial R_j}{\partial c_{j,k}}$$

$$= e(-1)w_j \frac{\partial R_j}{\partial c_{j,k}}$$

$$= e(-1)w_j R_j \frac{(x_k - c_{j,k})}{(\sigma_j)^2}$$

- Updating Centers:

$$\boxed{\Delta c_{j,k} = -\eta \frac{\partial E}{\partial c_{j,k}} = \eta\ e\ w_j\ R_j\ \frac{(x_k - c_{j,k})}{(\sigma_j)^2}}$$

Oregon State University

Implementation issues

- Training, Testing and Validation
- Network architecture and Training
- Initial weights & Parameter selection
- Local minima
- Momentum term for weights
- Network complexity
- Convergence
- Generalization
- Model complexity
- Universal approximator theorem

Oregon State University

Training, Testing and Validation

- Training: using known samples to set parameters
- Testing: Verifying that learned mapping applies to unseen samples
- Validation: "testing" on the training samples to set parameters

- Generalization: Ability to extend learning to new samples

- Example: 1000 data points
  - Use 600 for training: set parameters
  - Use 200 for validation: check performance, adjust parameters
  - Use 200 for testing: Generalization performance

- Cross-validation: Train and validate on data partitions.
  - 4-fold cross validation means split data into four and train on three quarters and validate on one fourth for each combination
  - All training data used for training (200 data points not used above)

Oregon State University

Network Architecture and Training

- Architecture:
  - Feed forward network
  - 2 Layer FFN
- Neuron selection
  - Activation functions
- Learning Algorithm
  - Gradient descent
- How many hidden units?
- How long to train?

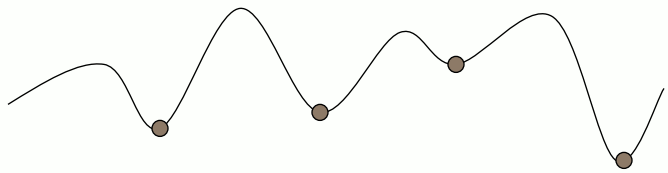
Oregon State University

Initial Weights

- Initial Weights
  - Random
  - Seed special concept
- Clustering (for RBF networks)

Oregon State University

Local Minima



- Can have multiple local minima
- Gradient descent goes to the closest local minimum:
  - solution: random restarts from multiple places in weight space

Momentum Term

- Weight update changes too fast with:

$$\Delta w_{i,j} = \eta \delta_j x_i$$

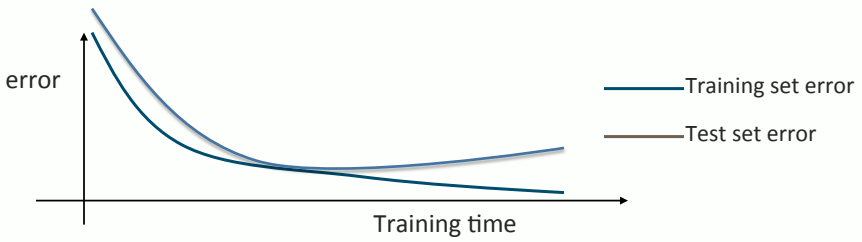
- Let each update be closer to last update. Give the gradient "momentum":

$$\boxed{\Delta w^n{}_{i,j} = \alpha \Delta w_{i,j}^{n-1} + \eta \delta_j{}^n \; x_i{}^n}$$

Convergence

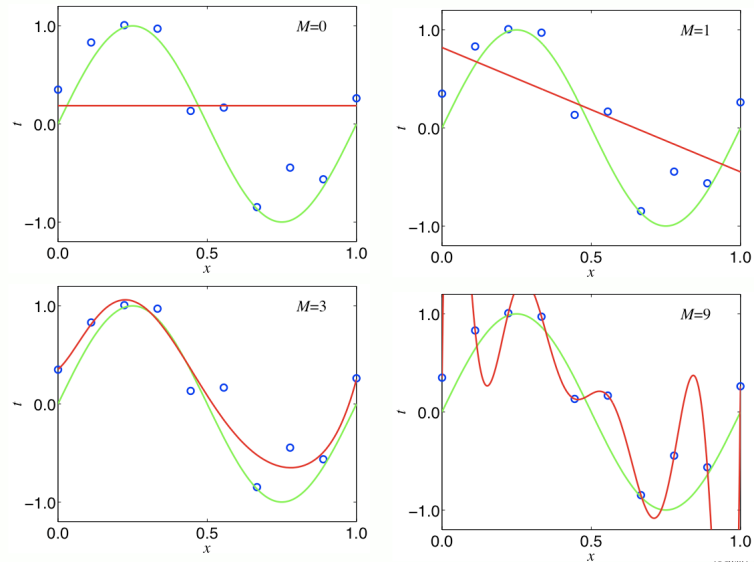
- Preset time:
  - Train for 2000 epochs
- Preset error criteria:
  - Train until MSE reaches
- Relative error criteria:
  - Train till MSE changes by less than .1% per epoch
- Use some "left out" patterns to validate training. When validation error bottoms out, stop training.

Generalization



- Training set error reduced continuously
- Test set error (generalization error) increases after a point
  - Network starts to learn the "noise" in the training data

Model Complexity



Oregon State University

Universal Function Approximation

- How good an approximator is a multi layer feed forward neural network?

Universal Approximation Theorem:

Under mild assumptions, for any given constant ε and continuous function $f(x_1,\ldots,x_m)$, there exists a one hidden-layer MLP (F):

$$F(x_1, \ldots, x_m) = \Sigma^i v_i \, h(\Sigma^j w_{ij} x_j + b_i)$$

h(.) is nonlinear activation function

with the property that :

$$| f(x_1,\ldots,x_m) - F(x_1,\ldots,x_m) | < \varepsilon$$

Cybenko 1989 ; Hornik 1991

Oregon State University

Video: Character Recognition

http://www.youtube.com/watch?v=3WR5pAa2m2E

Oregon State University

Video: Speed Sign Recognition

http://www.youtube.com/watch?v=kkha3sPoU70

Oregon State University