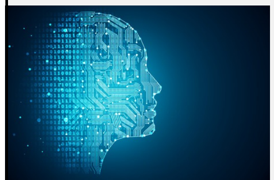
PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute



Data Science for Researchers and Scholars

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics, Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

1

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Regression and Multi-layer neural networks

- So far, we have focused on learning classifiers
- Now we turn to learning to approximate real-valued functions
  - Score on a test based on what we know about the student
  - Price of a stock based on its past performance and current market conditions
  - Price of a house given what we know about the characteristics of the neighborhood

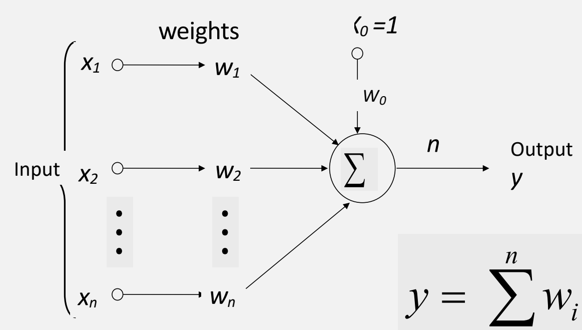
2

Simple Linear Regression

- In the simplest case, we have one (input), independent variable *x*, and one (output) dependent variable *y*
  - Multiple linear regression assumes an input vector **x**
  - Multivariate linear regression assumes an output vector **y**
- We will "fit" the points with a linear hyper-plane (line in the simplest case)
- Which line should we use?
  - Choose an objective function
  - For simple linear regression we choose sum squared error (SSE)
    - $\Sigma (d_i - y_i)^2 = \Sigma (e_i)^2$
  - Thus, find the linear surface which minimizes the sum of the squared residues (e.g. least squares)

3

Linear regression



$$y = \sum_{i=0}^{n} w_i x_i$$

4

Learning Task

$\mathbf{W} = [W_0 \ldots\ldots W_n]^T$ is the weight vector

$\mathbf{X}_p = \left[X_{0p} \ldots X_{np}\right]^T$ is the pth training sample

$y_p = \sum_i W_i X_{ip} = \mathbf{W} \bullet \mathbf{X}_p$ is the output of the neuron for input $\mathbf{X}_p$

$\mathbf{X}_p = f(\mathbf{X}_p)$ is the desired output for input $\mathbf{X}_p$

$e_p = (d_p - y_p)$ is the ***error*** of the neuron on input $\mathbf{X}_p$

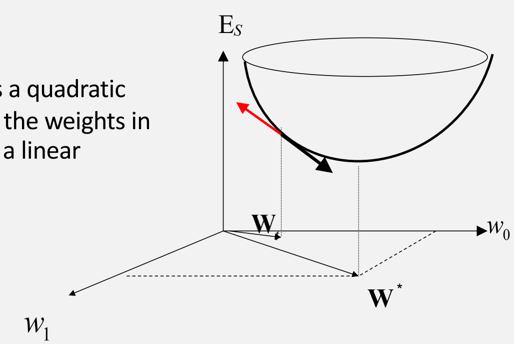
$S = \{(\mathbf{X}_p, d_p)\}$ is`a (multi) set of training examples

$E_S(\mathbf{W}) = E_S(W_0, W_1, \ldots\ldots W_n) = \frac{1}{2}\sum_p e_p^2$ is the estimated error of $\mathbf{W}$ on training set S

Goal: Find $\mathbf{W}^* = \arg\min_{\mathbf{W}} E_S(\mathbf{W})$

Learning linear functions

The error is a quadratic function of the weights in the case of a linear function



Learning linear functions

$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\frac{\partial}{\partial w_i}\left\{\sum_p e_p^2\right\} = \frac{1}{2}\left(\sum_p \frac{\partial}{\partial w_i}\left(e_p^2\right)\right)$$
$$= \frac{1}{2}\left(\sum_p (2e_p)\frac{\partial e_p}{\partial w_i}\right) = \sum_p e_p\left(\frac{\partial e_p}{\partial y_p}\right)\left(\frac{\partial y_p}{\partial w_i}\right) = \sum_p e_p(-1)\left(\frac{\partial}{\partial w_i}\left(\sum_{j=0}^{n} w_j x_{jp}\right)\right)$$
$$= -\sum_p \left(d_p - y_p\right)\left(\frac{\partial}{\partial w_i}\left(w_i x_{ip} + \sum_{j \neq i} w_j x_{jp}\right)\right)$$
$$= -\sum_p \left(d_p - y_p\right)\left(\frac{\partial}{\partial w_i}\left(w_i x_{ip}\right) + \frac{\partial}{\partial w_i}\left(\sum_{j \neq i} w_j x_{jp}\right)\right)$$
$$= -\sum_p \left(d_p - y_p\right) x_{ip}$$

$$w_i \leftarrow w_i + \eta \sum_p \left(d_p - y_p\right) x_{ip}$$

7

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Batch Update

$$w_i \leftarrow w_i + \eta \sum_p \left(d_p - y_p\right) x_{ip}$$

Per sample Update

$$w_i \leftarrow w_i + \eta \left(d_p - y_p\right) x_{ip}$$

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

8

Momentum update

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$\Delta w_i(t) = -\eta \left.\frac{\partial E}{\partial w_i}\right|_{w_i = w_i(t)} + \alpha \Delta w_i(t-1) \text{ where } 0 < \alpha < 1$$

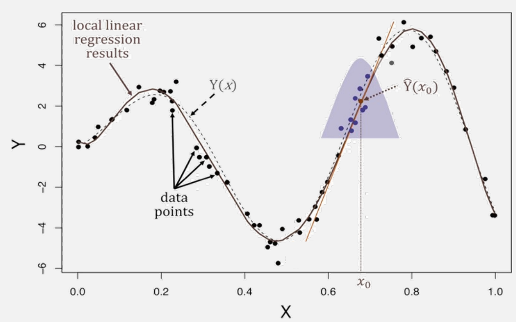
$$= -\eta \sum_{\tau=0}^{t} \alpha^{t-\tau} \left.\frac{\partial E}{\partial w_i}\right|_{w_i = w_i(\tau)}$$

The momentum update allows effective learning rate to increase when feasible and decrease when necessary.
Converges for $0 \leq \alpha < 1$

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Locally weighted regression

- What if the function is not linear?
- Perhaps we can approximate it by a collection of locally linear functions yielding a piecewise linear approximation?



Locally weighted regression

- Because local approximations are query dependent,
- We estimate a query dependent approximation of the function for a given query X_q based on the nearest neighbors of X_q
- Training the model involves simply storing the training data
- Locally weighted regression is performed when we have to make prediction for a given query X_q
- Let the approximation be of the form

$$g(X) = w_0 + \sum_{i=1}^{N} w_i x_i$$

in a small neighborhood around a query X_q

11

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Locally weighted regression

$$g(X) = w_0 + \sum_{i=1}^{N} w_i x_i$$

Minimize the error of the predicted value relative to the true value of the function over the K nearest neighbors of X_q

$$E(X_q) = \frac{1}{2} \sum_{X \in KNN(X_q)} (f(X) - g(X))^2$$

$$w_i \leftarrow w_i - \eta \frac{\partial E(X_q)}{\partial w_i}$$

$$w_i \leftarrow w_i + \eta \sum_{X \in KNN(X_q)} (f(X) - g(X)) \; x_i$$

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

12

Locally weighted regression

Minimize the error over all the neighbors of X_q in the training set weighted by an inverse function of distance to the neighbors

$$E_2(X_q) = \frac{1}{2} \sum_{X:(X,f(X))\in D} \left(f(X) - g(X)\right)^2 \phi\left(d(X_q, X)\right)$$

$$w_i \leftarrow w_i - \eta \frac{\partial E_2(X_q)}{\partial w_i}$$

$$w_i \leftarrow w_i + \eta \sum_{X:(X,f(X))\in D} \phi\left(d(X_q, X)\right)\left(f(X) - g(X)\right) x_i$$

13

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Locally weighted regression

Minimize the error over all the neighbors of X_q in the training set weighted by an inverse function of distance over the K nearest neighbors

$$E_2(X_q) = \frac{1}{2} \sum_{X:(X,f(X))\in D} \left(f(X) - g(X)\right)^2 \phi\left(d(X_q, X)\right)$$

$$w_i \leftarrow w_i - \eta \frac{\partial E_2(X_q)}{\partial w_i}$$

$$w_i \leftarrow w_i + \eta \sum_{X:(X,f(X))\in D} \phi\left(d(X_q, X)\right)\left(f(X) - g(X)\right) x_i$$

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

14

Neural Networks and Deep Learning

- Learning to approximate real-valued functions
- Bayesian recipe for learning real-valued functions
- Universal function approximation theorem
- Learning nonlinear functions using gradient descent
- Practical considerations and examples
- Deep Learning

Motivations

- Psychology – Empirical inadequacy of behaviorist theories of learning
  - simple reward-punishment based learning models are incapable of learning functions (e.g., exclusive OR) which are readily learned by animals (e.g., monkeys)
- Artificial Intelligence – the need for learning nonlinear functions where the form of the nonlinear relationship is unknown a-priori
- Statistics – Limitations of linear regression when the input-output relationship is nonlinear and is of unknown form
- Control – Need for nonlinear control methods

These considerations led multiple research communities to independently pursue generalizations of linear regression or the delta rule to the nonlinear setting

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Universal function approximation theorem* (UFAT)

- Let $\varphi : \Re \rightarrow \Re$ be a non-constant, bounded (hence non-linear), monotone, continuous function.
- Let I_N be the N-dimensional unit hypercube in $\Re^N$.
- Let $C(I_N) = \{f: I_N \rightarrow \Re\}$ be the set of all continuous functions with domain I_N *and* range $\Re$.
- Then for any function $f \in C(I_N)$ and any $\varepsilon > 0$, $\exists$ an integer L and a sets of real values $\theta, \alpha_j, \theta_j, w_{ji}$ $(1 \leq j \leq L;\ 1 \leq I \leq N)$ such that

$$F(x_1, x_2 \ldots x_N) = \sum_{j=1}^{L} \alpha_j \phi\left(\sum_{i=1}^{N} w_{ji} x_i - \theta_j\right) - \theta$$

is a uniform approximation of f – that is,

$$\forall (x_1, \ldots x_N) \in I_N, \quad \left|F(x_1, \ldots x_N) - f(x_1, \ldots x_N)\right| < \varepsilon$$

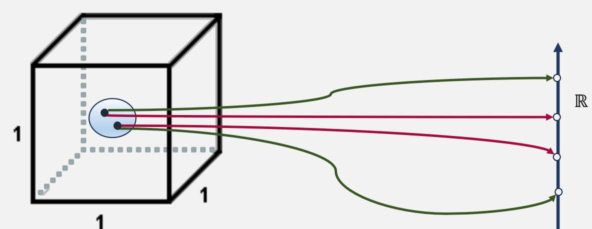
* Cybenko, 1989

PennState College of Information Sciences and Technology | Data Science for Researchers and Scholars | Vasant Honavar, Fall 2023

17

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Universal approximation theorem (UFAT) illustrated



- Two different functions (green, red) from $I_3 \rightarrow \mathbb{R}$
- UFAT asserts that any such continuous functions can be approximated arbitrarily well by a 3-layer feedforward network

PennState College of Information Sciences and Technology | Data Science for Researchers and Scholars | Vasant Honavar, Fall 2023

18

Universal function approximation theorem (UFAT)

$$F(x_1, x_2 \ldots x_n) = \sum_{j=1}^{L} \alpha_j \varphi\left(\sum_{i=1}^{N} w_{ji} x_i - \theta_j\right) - \theta$$

- The sigmoid function satisfies the UFAT requirements

$$\varphi(z) = \frac{1}{1+e^{-az}}; a > 0 \qquad \lim_{z \to -\infty} \varphi(z) = 0; \ \lim_{z \to +\infty} \varphi(z) = 1$$

- Later it was shown that similar universal approximation properties can be guaranteed for a variety of other choices for $\varphi(z)$ – e.g., radial basis functions, ReLU functions, etc.

Implications of Universal function approximation theorem

- UFAT guarantees the existence of arbitrarily accurate approximations of continuous functions defined over bounded subsets of $\Re^N$
- UFAT characterizes the representational power a certain class of multi-layer networks relative to the set of continuous functions defined on bounded subsets of $\Re^N$
- UFAT is not constructive – it does not tell us how to choose the parameters to construct a desired function
- To learn an unknown nonlinear continuous function from data, we need an algorithm to search the space of multilayer networks
- By interpreting the outputs of the network as posterior probabilities of classes, we can use such networks for classification

Feed-forward neural networks

- A feed-forward n-layer network consists of n layers of nodes
- 1 layer of Input nodes
- *n*-2 layers of Hidden nodes
- 1 layer of Output nodes
- interconnected by modifiable weights from input nodes to the hidden nodes and the hidden nodes to the output nodes
- More general topologies (e.g., with connections that skip layers, e.g., direct connections between input and output nodes) are possible

Three-layer feed-forward neural network

- A single bias unit (set to 1) is connected to each unit other than the input units
- Net input

$$n_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} \equiv \mathbf{W}_j . \bullet \mathbf{X},$$

- where the subscript i indexes units in the input layer, j in the hidden; w_{ji} denotes the input-to-hidden layer weights at the hidden unit j.
- The output of a hidden unit is a nonlinear function of its net input. That is, $y_j = f(n_j)$ e.g.,

$$y_j = \frac{1}{1+e^{-n_j}}$$

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Three-layer feed-forward neural network

- Each output unit similarly computes its net activation based on the hidden unit signals as:

$$n_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = \mathbf{W}_k \bullet \mathbf{Y},$$

- where the subscript k indexes units in the output layer and n_H denotes the number of hidden units
- The output can be a linear or nonlinear function of the net input e.g., $z_k = n_k$

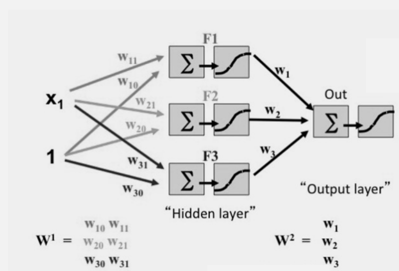
PennState | Data Science for Researchers and Scholars | Vasant Honavar, Fall 2023

23

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Computing nonlinear functions

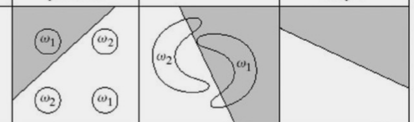
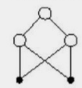
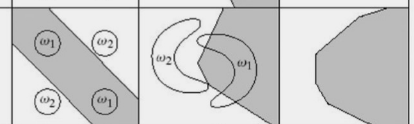
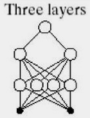
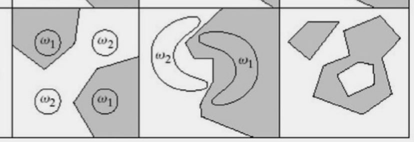
- A 2-layer network with 2 inputs and 1 output
- The hidden layer and output layer nodes are sigmoid neurons



PennState | Data Science for Researchers and Scholars | Vasant Honavar, Fall 2023

24

Decision boundaries realizable by multi-layer neural networks

Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)			

25

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Learning nonlinear functions

- Given a training set determine
  - Network structure – number of hidden nodes or more generally, network topology
    - Architecture search
      - Start small and grow the network
      - Start large and prune the network
  - For a given structure, determine the parameters (weights) that minimize the error (loss) on the training data
    - Mean squared error for function approximation
    - Classification error (e.g., smooth loss) for classification
- For now, we focus on the latter

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

26

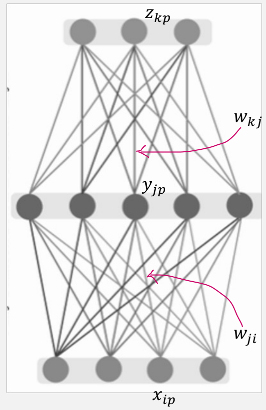
Generalized delta rule – error back-propagation

- Challenge – we know the desired outputs for nodes in the output layer, but not the hidden layer
- Presents the credit assignment problem – dividing the credit or blame for the performance of the output nodes among hidden nodes
- Generalized delta rule offers an elegant solution to the credit assignment problem
  - in feed-forward neural networks in which each neuron computes a differentiable function of its inputs
  - Solution generalizes to other kinds of networks with differentiable error functions, including recurrent networks (with feedback loops), modern deep neural networks, etc.

Feed-forward networks

- Forward operation (computing output for a given input based on the current weights)
- Learning – modification of the network parameters (weights) to minimize an appropriate error measure
- Because each neuron computes a differentiable function of its inputs
  - If error is a differentiable function of the network outputs, it is a differentiable function of the weights
  - We can learn the weights by performing gradient descent!

A fully connected 2-layer network



Given the the pth sample $\mathbf{x}_p$

- Let x_{ip} be the ith input
- Let $n_{jp} = \sum_i \; w_{ji}\, x_{ip}$ be the net input of the jth hidden neuron
- Let $y_{jp} = \frac{1}{1+e^{-n_{jp}}}$ be the output of the jth hidden neuron
- Let $n_{kp} = \sum_j \; w_{kj}\, z_{jp}$ be the net input of the kth output neuron
- Let $z_{kp} = n_{kp}$ be the output of the kth output neuron

29

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Generalized delta rule

- Let t_{kp} be the k-th target (or desired) output for input pattern $\mathbf{X}_p$ and z_{kp} be the output produced by k-th output node and let $\mathbf{W}$ represent all the weights in the network
- Training error: $E_S(\mathbf{W}) = \frac{1}{2}\sum_p \sum_{k=1}^{M} (t_{kp} - z_{kp})^2 = \sum_p E_p(\mathbf{W})$
- The weights are initi ues and are changed in a direction that will reduce the error:

Batch Update $\quad \Delta w_{ji} = -\eta \frac{\partial E_S}{\partial w_{ji}} \quad \Delta w_{kj} = -\eta \frac{\partial E_S}{\partial w_{kj}}$

Per sample update $\quad \Delta w_{ji} = -\eta \frac{\partial E_p}{\partial w_{ji}} \quad \Delta w_{kj} = -\eta \frac{\partial E_p}{\partial w_{kj}}$

PennState College of Information Sciences And Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

30

Generalized delta rule

Change in hidden to output weights $\Delta w_{kj} = -\eta \dfrac{\partial E_p}{\partial w_{kj}}$

$$E_p = \frac{1}{2}\sum_k \left(t_{kp} - z_{kp}\right)^2$$

$$\frac{\partial E_p}{\partial w_{kj}} = \frac{\partial E_p}{\partial z_{kp}}\frac{\partial z_{kp}}{\partial w_{kj}} = \frac{\partial E_p}{\partial z_{kp}}\frac{\partial z_{kp}}{\partial n_{kp}}\frac{\partial n_{kp}}{\partial w_{kj}} = -\left(t_{kp} - z_{kp}\right)(1)y_{jp}$$

Let $t_{kp} - z_{kp} = \delta_{kp}$

$$E_p \rightarrow z_{kp} \rightarrow n_{kp} \rightarrow w_{kj}$$

$$w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E_p}{\partial w_{kj}} = w_{kj} + \left(t_{kp} - z_{kp}\right) y_{jp} = w_{kj} + \delta_{kp} y_{jp}$$

31

Generalized delta rule

Change in input to hidden weights $\Delta w_{ji} = -\eta \dfrac{\partial E_p}{\partial w_{ji}}$

$$\frac{\partial E_p}{\partial w_{ji}} = \sum_{k=1}^{M} \frac{\partial E_p}{\partial z_{kp}}\frac{\partial z_{kp}}{\partial w_{ji}} = \sum_{k=1}^{M} \frac{\partial E_p}{\partial z_{kp}}\frac{\partial z_{kp}}{\partial y_{jp}}\cdot\frac{\partial y_{jp}}{\partial n_{jp}}\cdot\frac{\partial n_{jp}}{\partial w_{ji}}$$

$$= \sum_{k=1}^{M} \frac{\partial}{\partial z_{kp}}\left[\frac{1}{2}\sum_{l=1}^{M}\left(t_{lp} - z_{lp}\right)^2\right]\left(w_{kj}\right)\left(y_{jp}\right)\left(1 - y_{jp}\right)\left(x_{ip}\right)$$

$$= -\sum_{k=1}^{M}\left(t_{kp} - z_{kp}\right)\left(w_{kj}\right)\left(y_{jp}\right)\left(1 - y_{jp}\right)\left(x_{ip}\right)$$

$$= -\underbrace{\left(\sum_{k=1}^{M}\delta_{kp}\left(w_{kj}\right)\left(y_{jp}\right)\left(1 - y_{jp}\right)\right)}_{\delta_{jp}}\left(x_{ip}\right)$$

$$= -\delta_{jp} x_{ip}$$

Chain Rule

$$E_p \rightarrow z_{1p} \cdots z_{kp} \cdots z_{Mp} \rightarrow y_{jp} \rightarrow n_{jp} \rightarrow w_{ji}$$

$$w_{ji} \leftarrow w_{ji} + \eta \delta_{jp} x_{ip}$$

32

In the preceding slide, we have made use of the fact that

$$\frac{\partial y_{jp}}{\partial n_{jp}} = \frac{\partial}{\partial n_{jp}}\left(\frac{1}{1+e^{-n_{jp}}}\right)$$

$$= \frac{(-e^{-n_{jp}})}{(1+e^{-n_{jp}})^2}$$

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

$$= y_{jp}(1-y_{jp})$$

33

Backpropagation algorithm

- Start with small random initial weights
- Until desired stopping criterion is satisfied do
- Select a training sample from S
  - Compute the outputs of all nodes based on current weights and the input sample
  - Compute the weight updates for output nodes
  - Compute the weight updates for hidden nodes
  - Update the weights

34

Using neural networks for classification

- Network outputs are real valued.
- How can we use the networks for classification?

$$F(\mathbf{X}_p) = \arg\max_k z_{kp}$$

Classify a pattern by assigning it to the class that corresponds to the index of the output node with the largest output for the pattern

Training multi-layer networks – Some Useful Tricks

- Initializing weights to small random values that place the neurons in the linear portion of their operating range for most of the patterns in the training set improves speed of convergence e.g.,

$$w_{ji} = \pm \frac{1}{2N} \sum_{i=1,\ldots,N} \frac{1}{|x_i|}$$

For input to hidden layer weights with the sign of the weight chosen at random

$$w_{kj} = \pm \frac{1}{2N} \sum_{i=1,\ldots,N} \left(\frac{1}{\varphi\left(\sum w_{ji} x_i\right)} \right)$$

For hidden to output layer weights with the sign of the weight chosen at random

Some Useful Tricks

- Use of momentum term allows the effective learning rate for each weight to adapt as needed and helps speed up convergence – in a network with 2 layers of weights,

$$\left.\begin{aligned}
w_{ji}(t+1) &= w_{ji}(t)+\Delta w_{ji}(t)\\
\Delta w_{ji}(t) &= -\eta \left.\frac{\partial E_S}{\partial w_{ji}}\right|_{w_{ji}=w_{ji}(t)} + \alpha \Delta w_{ji}(t-1)\\
w_{kj}(t+1) &= w_{kj}(t)+\Delta w_{kj}(t)\\
\Delta w_{kj}(t) &= -\eta \left.\frac{\partial E_S}{\partial w_{ji}}\right|_{w_{kj}=w_{kj}(t)} + \alpha \Delta w_{kj}(t-1)
\end{aligned}\right\}$$

where $0 < \alpha, \eta < 1$ with typical values of $\eta = 0.5$ to 0.6, $\alpha = 0.8$ to 0.9

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some Useful Tricks

- Use sigmoid function which satisfies $\varphi(-z) = -\varphi(z)$ helps speed up convergence

$$\varphi(z) = a\left(\frac{1-e^{-bz}}{1+e^{-bz}}\right)$$

$$a = 1.716,\ \mathrm{b} = \frac{2}{3} \Rightarrow \left.\frac{\partial \varphi}{\partial z}\right|_{z=0} \approx 1$$

$$\text{and } \varphi(z) \text{ is linear in the range } -1 < z < 1$$

Some Useful Tricks

- Randomize the order of presentation of training examples from one pass to the next helps avoid local minima
- Introduce small amounts of noise in the weight updates (or into examples) during training helps improve generalization
  - minimizes over fitting
  - makes the learned approximation more robust to noise
  - helps avoid local minima
- If using the suggested sigmoid nodes in the output layer, set target output for output nodes to be 1 for target class and -1 for all others

39

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some useful tricks

- Regularization helps avoid over fitting and improves generalization

$$R(\mathbf{W}) = \lambda E(\mathbf{W}) + (1-\lambda)C(\mathbf{W});\ 0 \le \lambda \le 1$$

$$C(\mathbf{W}) = \frac{1}{2}\left(\sum_{ji} w_{ji}^2 + \sum_{kj} w_{kj}^2\right)$$

$$-\frac{\partial C}{\partial w_{ji}} = -w_{ji} \text{ and } -\frac{\partial C}{\partial w_{kj}} = -w_{kj}$$

Start with λ close to 1 and gradually lower it during training. When λ <1, it tends to drive weights toward zero setting up a tension between error reduction and complexity minimization

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

40

Some Useful Tricks

Input and output encodings

- Do not eliminate *natural* proximity in the input or output space
  - Do not normalize input patterns to be of unit length if the length is likely to be relevant for distinguishing between classes
- Do not introduce *unwarranted* proximity as an artifact
  - Do not use $\log_2 M$ outputs to encode M classes, use M outputs instead to avoid spurious proximity in the output space

41

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some Useful Tricks

Examples of a good code

- Binary thermometer codes for encoding real values
  - Suppose we can use 10 bits to represent a value between -1.0 and +1.0
  - We can quantize the interval [-1, 1] into 10 equal parts
  - 0.38 in thermometer code is 1111000000
  - 0.60 in thermometer code is 1111110000
  - Note values that are close along the real number line have thermometer codes that are close in Hamming distance

Example of a bad code

- Ordinary binary representations of integers

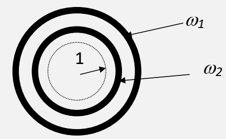
PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

42

Some Useful Tricks

- Normalizing inputs – know when and when not to normalize
- Normalizing each input pattern so that it is of unit length is commonplace, especially among engineers, but often a bad idea

$$\mathbf{X}_p \leftarrow \frac{\mathbf{X}_p}{\left\|\mathbf{X}_p\right\|}$$



  - Two classes ω_1 and ω_2 that were separable become not separable after normalization because they both map to the unit circle

Some Useful Tricks

- Better: Scale each component of the input separately to lie between -1 and 1 with mean of 0 and standard deviation of 1

$$\mu_i = \frac{1}{P}\sum_{q=1}^{P} x_{iq}$$

$$\sigma_i^{\,2} = \frac{1}{P}\sum_{q=1}^{P} x_{iq}^2 - \mu_i^2$$

$$x_{ip} \leftarrow \frac{\left(x_{ip} - \mu_i\right)}{\sigma_i}$$

Some Useful Tricks

Initializing weights (revisited)

Suppose weights are uniformly distributed between $-w$ and $+w$

Standardized input to a hidden neuron is distributed between $-w\sqrt{N}$ and $+w\sqrt{N}$

We want this to fall between -1 and +1 $\Rightarrow \left(w = \frac{1}{\sqrt{N}} \right)$

$$\Rightarrow -\frac{1}{\sqrt{N}} < w_{ji} < \frac{1}{\sqrt{N}}$$

$$-\frac{1}{\sqrt{n_H}} < w_{kj} < \frac{1}{\sqrt{n_H}}$$

45

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some Useful Tricks

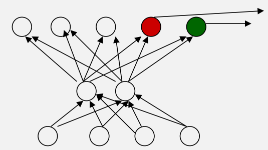
- Use of problem specific information (if known) speeds up convergence and improves generalization
- In networks designed for translation-invariant visual image classification, building in translation invariance as a constraint on the weights helps
- If we know the function to be approximated is smooth, we can build that in as part of the criterion to be minimized
  - minimize in addition to the error, the gradient of the error with respect to the inputs

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

46

Some Useful Tricks

- Manufacture training data – training networks with translated and rotated patterns if translation and rotation invariant recognition is desired
- Incorporate hints during training
- Hints are used as additional outputs during training to help shape the hidden layer representation



Hint nodes (e.g., vowels versus consonants in training a phoneme recognizer)

47

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some Useful Tricks

- Reducing the effective number of free parameters (degrees of freedom) helps improve generalization
- Regularization
- Preprocess the data to reduce the dimensionality of the input
  - Train an “auto encoder” neural network with output same as input, but with fewer hidden neurons than the number of inputs
  - Use the hidden layer outputs as inputs to a second network to do function approximation

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

48

Some Useful Tricks

- Choice of appropriate error function is critical
  - Do not blindly minimize sum squared error
  - There are many cases where other criteria are appropriate
- Example

$$E_S(\mathbf{W}) = \sum_{p=1}^{P} \sum_{k=1}^{M} t_{kp} \ln\left(\frac{t_{kp}}{z_{kp}}\right)$$

Cross-entropy error function is appropriate for minimizing the distance between the target probability distribution over the *M* output variables and the probability distribution represented by the network

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Some Useful Tricks

- Interpreting the outputs as class conditional probabilities
- Use exponential output nodes

$$n_{kp} = \sum_{j=0}^{n_H} w_{kj} y_{jp}$$

$$\text{linear output} \quad z_{kp} = \left(\frac{n_{kp}}{\sum_{l=1}^{M} n_{lp}}\right)$$

$$\text{exponential output} \; z_{kp} = \left(\frac{e^{n_{kp}}}{\sum_{l=1}^{M} n_{kp}}\right)$$

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

Bayes classification and Neural Networks

$$P(\omega_k \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \omega_k) P(\omega_k)}{\sum_{l=1}^{M} P(\mathbf{X} \mid \omega_l) P(\omega_l)}$$

$$\left.\begin{aligned} t_k(\mathbf{X}_p) &= t_{kp} = 1 \text{ if } \mathbf{X}_p \in \omega_k \\ t_k(\mathbf{X}_p) &= t_{kp} = 0 \text{ if } \mathbf{X}_p \notin \omega_k \end{aligned}\right\} k\text{th target output}$$

$$g_k(\mathbf{X}_p; \mathbf{W}) = k\text{th output for input } \mathbf{X}_p$$

$$E_S(\mathbf{W}) = \sum_{p=1}^{P} \left(g_k(\mathbf{X}_p; \mathbf{W}) - t_{kp} \right)^2$$

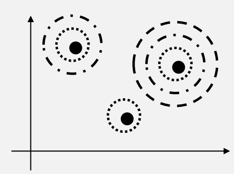
We can show that the error is minimized when

$$g_k(\mathbf{X}; \mathbf{W}) \approx P(\omega_k \mid \mathbf{X})$$

51

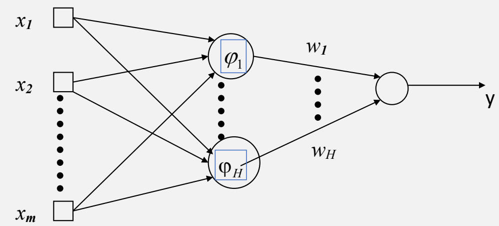
Radial-Basis Function Networks

- A function is approximated as a linear combination of radial basis functions (RBF). RBFs capture local behaviors of functions.
- RBFs represent local receptive fields



52

Radial Basis Function Networks



- Hidden layer applies a non-linear transformation from the input space to the hidden space.
- Output layer applies a linear transformation from the hidden space to the output space.

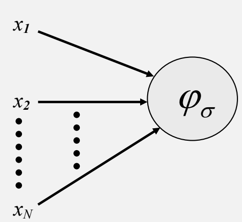
53

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Example of a radial basis function

- Hidden units: use a radial basis function

$\phi\sigma(\,||\mathbf{X}-\mathbf{W}||^2)$ the output depends on the distance of the input x from the center t



$\phi_\sigma(\,||\mathbf{X}-\mathbf{W}||^2)$

W is called center
σ is called spread
center and spread are parameters

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

54

Radial basis function

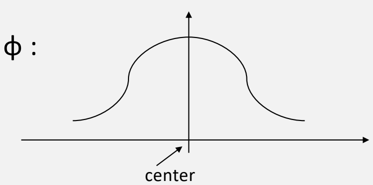
- A hidden neuron is more sensitive to data points near its center. This sensitivity may be tuned by adjusting the spread σ.
- Larger spread $\Rightarrow$ less sensitivity
- Neurons in the visual cortex have locally tuned frequency responses.

55

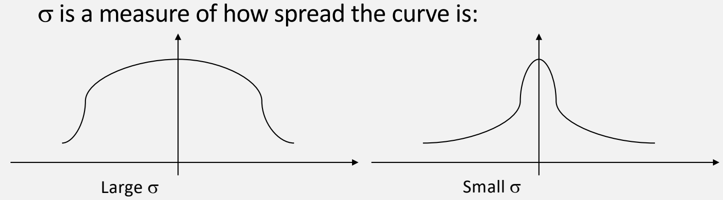
PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Gaussian Radial Basis Function ϕ

ϕ :



σ is a measure of how spread the curve is:



PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

56

Types of radial basis functions

- Multiquadrics

$$\varphi(r) = (r^2 + c^2)^{1/2}$$

$c > 0$

$r = \| \mathbf{X} - \mathbf{W} \|$

- Inverse multiquadrics

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \qquad c > 0$$

- Gaussian functions:

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \qquad \sigma > 0$$

57

RBF Learning Algorithm

$$\Delta\alpha_j = -\eta_j \frac{\partial E_S}{\partial \alpha_j}$$

$$\boxed{\Delta\sigma_j = -\eta_{\sigma_j} \frac{\partial E_p}{\partial \sigma_j}}$$

$$\boxed{\Delta w_{ji} = -\eta_{ji} \frac{\partial E_p}{\partial w_{ji}}}$$

The necessary gradients can be calculated using chain rule

58

RBF Learning Algorithm

$$z_{jp} = e^{-\frac{-\|\mathbf{x}_p - \mathbf{w}_j\|^2}{2\sigma_j^2}}$$

$$y_p = \sum_{j=0}^{L} \alpha_j z_{jp}$$

$$E_p = \frac{1}{2}\left(t_p - y_p\right)^2$$

$$\mathbf{X}_p = \left[x_{1p}.........x_{Np}\right]^T$$

$$\mathbf{W}_j = \left[w_{j1}......w_{jN}\right]^T$$

$$\Delta\alpha_j = -\eta_j \frac{\partial E_p}{\partial \alpha_j}$$

$$\boxed{\Delta\sigma_j = -\eta_{\sigma_j} \frac{\partial E_p}{\partial \sigma_j}}$$

$$\boxed{\Delta w_{ji} = -\eta_{ji} \frac{\partial E_p}{\partial w_{ji}}}$$

59

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

RBF Learning Algorithm

$$\Delta\alpha_j = -\eta_j \frac{\partial E_p}{\partial \alpha_j} = \eta_j\left(t_p - y_p\right)z_{jp}$$

$$\alpha_j \leftarrow \alpha_j + \eta_j\left(t_p - y_p\right)z_{jp}$$

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial z_{jp}}\frac{\partial z_{jp}}{\partial w_{ji}}$$

$$= -\left(t_p - y_p\right)\alpha_j\left(\frac{z_{jp}}{\sigma_j^2}\right)\left(x_{ip} - w_{ji}\right)$$

$$w_{ji} = w_{ji} + \eta_{ji}\left(t_p - y_p\right)\alpha_j\left(\frac{z_{jp}}{\sigma_j^2}\right)\left(x_{ip} - w_{ji}\right)$$

PennState College of Information Sciences and Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

60

RBF Learning Algorithm

$$\frac{\partial E_p}{\partial \sigma_j} = \frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial z_{jp}}\frac{\partial z_{jp}}{\partial \sigma_j}$$

$$= -(t_p - y_p)\alpha_j(-z_{jp})\left(\left(\frac{2}{\sigma_j}\right)(\ln z_{jp})\right)$$

$$\sigma_j \leftarrow \sigma_j - \eta_j(t_p - y_p)\alpha_j(z_{jp})\left(\left(\frac{2}{\sigma_j}\right)(\ln z_{jp})\right)$$

61

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

Generalized RBF Learning Algorithm

Some useful facts

$$\|V\|^2 = V^T V \;\; \text{(norm)}$$

$$\|V\|_C^2 = (CV)^T(CV) = V^T C^T C V \text{ (weighted norm)}$$

$$\|V\|_C^2 = \|V\|^2 \text{ if } C^T C = \text{identity matrix}$$

$$\frac{d}{d\mathbf{X}}(A\mathbf{X}) = A$$

$$\frac{d}{d\mathbf{X}}(\mathbf{X}^T A\mathbf{X}) = 2A\mathbf{X} \text{ (when A is a symmetric matrix)}$$

$$\frac{d}{dA}(\mathbf{X}^T A\mathbf{X}) = \mathbf{X}^T\mathbf{X}$$

PennState College of Information Sciences And Technology
Data Science for Researchers and Scholars
Vasant Honavar, Fall 2023

62

Generalizd RBF Learning Algorithm

$$z_{jp} = e^{-\frac{1}{2}(\mathbf{x}_p - \mathbf{w}_j)^T \Sigma_j (\mathbf{x}_p - \mathbf{w}_j)}$$

$$y_p = \sum_{j=0}^{L} \alpha_j z_{jp}$$

$$E_p = \frac{1}{2}\left(t_p - y_p\right)^2$$

$$\mathbf{X}_p = \left[x_{1p}.........x_{Np}\right]^T$$

$$\mathbf{W}_j = \left[w_{j1}......w_{jN}\right]^T$$

Exercise: Derive the update equations for $\mathbf{w}_j$, Σ_j and α_j

RBF Learning Algorithm

- Initialize the parameters -- centers of the hidden neurons are typically initialized to coincide with a subset of the training set
- Use gradient descent to adjust the parameters using the training data until the desired performance criterion is satisfied