Feed-forward Network Functions

Sargur Srihari

Topics

- 1. Extension of linear models
- 2. Feed-forward Network Functions
- 3. Weight-space symmetries

Recap of Linear Models

Linear Models for Regression, Classification have form

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{j} \boldsymbol{\phi}_{j}(\boldsymbol{x})\right)$$

where x is a D-dimensional vector $\phi_j(x)$ are fixed nonlinear basis functions e.g., Gaussian, sigmoid or powers of x

- For Regression f is identity function
- For Classification f is a nonlinear *activation* function
 - If f is sigmoid it is called logistic regression

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

Limitation of Simple Linear Models

Linear models with fixed basis functions have limited applicability

$$\left| y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{j} \phi_{j}(\boldsymbol{x})\right) \right|$$

- This is due to curse of dimensionality
 - E.g., no of coefficients needed, means of Gaussian, grow with no. of varaibles
- To extend such models to large-scale problems it is necessary to adapt the basis functions ϕ_i to data
 - Become useful in large scale problems
- Both SVMs and Neural Networks address this limitation

Extending Linear Models

• SVMs and Neural Networks address dimensionality limitation of Generalized linear model $y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{i=1}^{M} w_i \phi_i(\mathbf{x})\right)$

1. The SVM solution

- By varying number of basis functions M centered on training data points
 - Select subset of these during training
 - Advantage: although training involves nonlinear optimization, the objective function is convex and solution is straightforward
 - No of basis function is much smaller than no of training points, although still large and grows with size of training se

2. The Neural Network solution

- No of basis functions M is fixed in advance, but allow them to be adaptive
 - But the ϕ_j have their own parameters $\{w_{ji}\}$
 - Most successful model of this type is a feed-forward neural network
 - Adapt all parameter values during training

• Instead of
$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right)$$
 we have $y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$

SVM versus Neural Networks

SVM

- Involves non-linear optimization
- Objective function is convex
 - Leads to straightforward optimization, e.g., single minimum
- Number of basis functions is much smaller than number of training points
 - Often large and increases with size of training set
- RVM results in sparser models
 - Produces probabilistic outputs at expense of non-convex optimization

Neural Network

- Fixed number of basis functions, parametric forms
 - Multilayer perceptron uses layers of logistic regression models
 - Also involves non-convex optimization during training (many minima)
 - At expense of training, get a more compact and faster model

Origin of Neural Networks

- To find information processing models of biological systems
- Term covers wide range of models
 - Exaggerated claims of biological plausibility
- Biological realism imposes unnecessary constraints
- Neural networks are efficient models for machine learning
 - Particularly multilayer perceptrons
- Network parameters are obtained in maximum likelihood framework
 - A nonlinear optimization problem
 - Requires evaluatiing derivative of log-likelihood function wrt network parameters
 - Done efficiently using error back propagation

Feed Forward Network Functions

A neural network can also be represented similar to linear models but basis functions are generalized

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_{j} \phi_{j}(\boldsymbol{x})\right)$$

activation function

For regression: identity

function

For classification: a non-

linear function

Coefficients w_j adjusted during training

There can be several activation functions

Basis functions

 $\phi_j(x)$ a nonlinear function of a linear combination of D inputs

its parameters are adjusted during training

8

Basic Neural Network Model

- Can be described by a series of functional transformations
- D input variables $x_1, ..., x_D$
- M linear combinations in the form

$$a_{j} = \sum_{i=1}^{D} w_{ji}^{(1)} x_{i} + w_{j0}^{(1)}$$
 where $j = 1,..,M$

- Superscript (1) indicates parameters are in first layer of network
- Parameters $w_{ii}^{(1)}$ are referred to as weights
- Parameters $w_0^{(1)}$ are biases, with $x_0=1$
- Quantities a_j are known as *activations* (which are input to activation functions)
- No of hidden units M can be regarded as no. of basis functions
 - Where each basis function has parameters which can be adjusted⁹

Activation Functions

Each activation a_i is transformed using differentiable nonlinear activation functions

$$z_j = h(a_j)$$

- The z_i correspond to *outputs of basis functions* $\phi_i(x)$
 - or first layer of network or hidden units
- Nonlinear functions h
- Three examples of activation functions

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

1. Logistic sigmoid 2. Hyperbolic tangent

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

3. Rectified Linear unit

$$f(x) = \max(0, x)$$

Second Layer: Activations

• Values z_j are again linearly combined to give output unit activations

$$a_k = \sum_{i=1}^{M} w_{ki}^{(2)} x_i + w_{k0}^{(2)}$$
 where $k = 1,..., K$

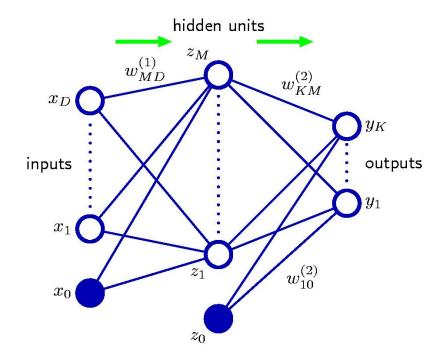
- where K is the total number of outputs
- This corresponds to the second layer of the network, and again $w_{k0}^{(2)}$ are bias parameters
- Output unit activations are transformed by using appropriate activation function to give network outputs y_k

Choice of Activation Function

- Determined by the nature of the data and the assumed distribution of the target variables
- For standard regression problems the activation function is the identity function so that $y_k \!\!=\! a_k$
- For multiple binary classification problems, each output unit activation is transformed using a logistic sigmoid function so that $y_k = \sigma\left(a_k\right)$
- For multiclass problems, a softmax activation function of the form $\frac{\exp(a_k)}{\sum \exp(a_j)}$ is used.

Network Diagram for two-layer neural network

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$



Sigmoid and Softmax output activation

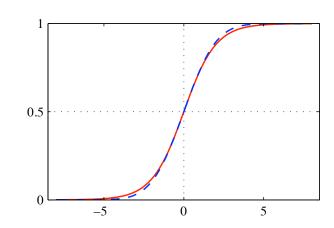
- Standard regression problems
 - identity function

$$y_k = a_k$$

- For binary classification
 - logistic sigmoid function

$$y_k = \sigma(a_k)$$

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



- For multiclass problems
 - a softmax activation function

$$\frac{\exp(a_{_k})}{\sum_{_j} \exp(a_{_j})}$$

Note that logistic sigmoid also involves an exponential making it a special case of softmax

 Choice of output Activation Functions is further discussed in Network Training

tanh is a rescaled sigmoid function

The logistic sigmoid function is

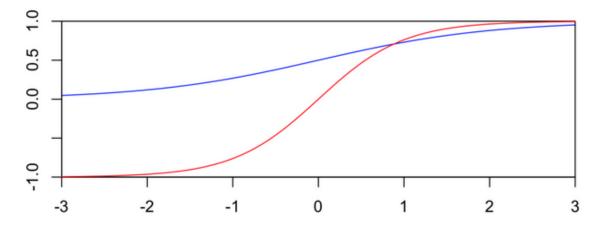
$$\sigma(a) = \frac{1}{1 + e^{-a}}$$

- The outputs range from 0 to 1 and are often interpreted as probabilities
- Tanh is a rescaling of logistic sigmoid, such that outputs range from -1 to 1. There is horizontal stretching as well.
 - It leads to the standard definition $|\tanh(a) = 2\sigma(2a) 1|$

$$\tanh(a) = 2\sigma(2a) - 1$$

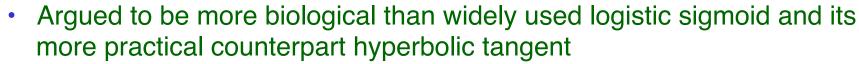
$$\tanh(a) = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

- The (-1,+1) output range is more convenient for neural networks.
- The two functions are plotted below: blue is the logistic and red is tanh

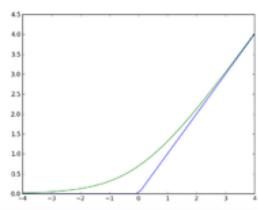


Rectifier Linear Unit (ReLU) Activation

- $f(x) = \max(0,x)$
- Where x is input to activation function
 - This is a ramp function

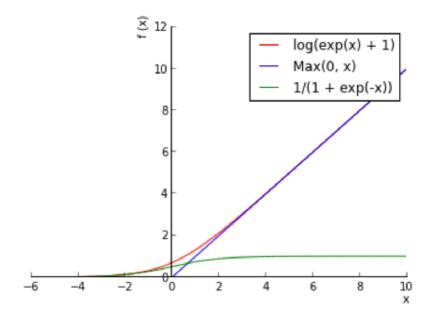


- Popular activation function for deep neural networks since function does not quickly saturate (leading to vanishing gradients)
- Smooth approximation is $f(x) = \ln (1 + e^x)$ called the *softplus*
- Derivative of softplus is $f'(x)=e^x/(e^x+1)=1/(1+e^{-x})$ i.e., logistic sigmoid
- Can be extended to include Gaussian noise



Comparison of ReLU and Sigmoid

- Sigmoid has gradient vanishing problem as we increase or decrease x
- Plots of softplus, hardmax (Max) and sigmoid:



Overall Network Function

Combining stages of the overall function with sigmoidal output

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

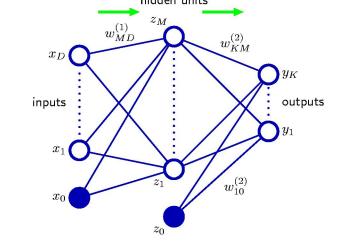
- Where w is the set of all weights and bias parameters
- Thus a neural network is simply
 - a set of nonlinear functions from input variables $\{x_{\rm i}\}$ to output variables $\{y_{\rm k}\}$
 - controlled by vector w of adjustable parameters
- Note presence of both σ and h functions

Forward Propagation

Bias parameters can be absorbed into weight parameters by

defining a new input variable x_0

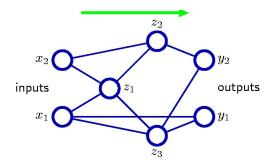
$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma \left(\sum_{j=1}^{M} w_{kj}^{(2)} h \left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i \right) \right)$$



- Process of evaluation is forward propagation through network
- Multilayer perceptron is a misnomer
 - since only continuous sigmoidal functions are used

Feed Forward Topology

Network can be sparse with not all connections being present



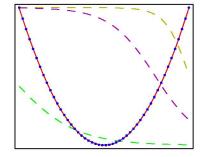
- More complex network diagrams
- But restricted to feed forward architecture
 - No closed cycles ensures outputs are deterministic functions of inputs
- Each hidden or output unit computes a function given by

$$z_{_k} = h \left(\sum_{j} w_{_{kj}} z_{_j} \right)$$

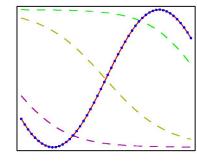
Examples of Neural Network Models of Regression

- Two-layer network with three hidden units
- Three hidden units collaborate to approximate the final function
- Dashed lines show outputs of hidden units
- 50 data points (blue dots) from each of four functions (over [-1,1])

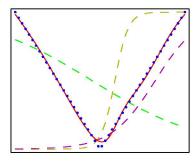
$$f(x) = x^2$$



$$f(x) = \sin(x)$$

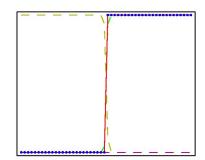


$$f(x) = |x|$$



$$f(x) = H(x)$$

Heaviside Step Function



21

Example of Two-class Classification

Neural Network:

Two inputs, two hidden units with tan h activation functions And single output with sigmoid activation

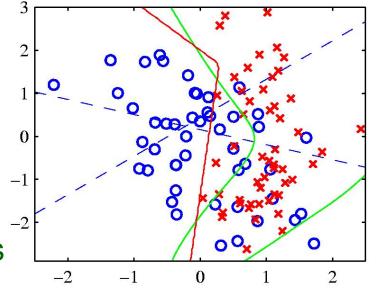
Synthetic dataset

Red line:

decision boundary y=0.5 for network

Dashed blue lines:

contours for z=0.5 of two hidden units

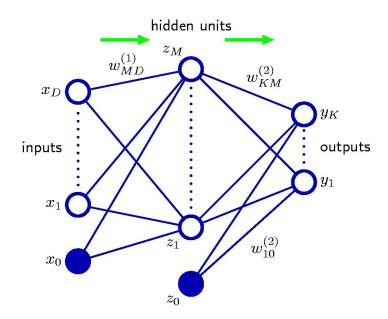


Green line:

optimal decision boundary from distributions used to generate the data

Weight-Space Symmetries

- A property of feedforward networks that plays a role in Bayesian model comparison is that:
 - There are multiple distinct choices for the weight vector w can all give rise to the same mapping function from inputs to outputs
- Consider a two-layer network of the form below with M hidden units having \tanh activation function and full connectivity in both layers



Fully connected network with Weight-Space Symmetries

 M hidden units change sign of all weights and bias feeding to particular hidden node

• Since $\tanh(-a) = -\tanh(a)$ this change can be compensated by changing sign of all weights leading out of that hidden unit

For M hidden units M such sign-flip symmetries

Thus any given weight vector will be one of 2^M equivalent weight vectors Since the values of all weights and bias of a node can be interchanged for M hidden units there are M! equivalent weight vectors

Network has overall weight space symmetry factor of $M!2^M$

 For networks with more than two layers of weights, the total level of symmetry will be given by the product of such factors, one for each layer of hidden units

24