

CS545: Artificial Neural Networks

Chuck Anderson

Department of Computer Science
Colorado State University

Fall, 2009

Outline

Introduction

Structure

Training Algorithm

Gradient Descent

Derivation of Error Gradient

Introduction

Structure

Training Algorithm

Gradient Descent

Derivation of Error

Gradient

Linear Models as Neural Networks

Given X and T , find \mathbf{w}_k that minimizes squared error in output, then use it to make predictions. Collect all \mathbf{w}_k as columns in \mathbf{W} . $\tilde{\mathbf{X}}$ denotes \mathbf{X} with constant 1 column.

$$E(\mathbf{W}) = \sum_{n=1}^N \sum_{k=1}^K (t_{n,k} - \tilde{\mathbf{x}}_n^T \mathbf{w}_k)^2$$

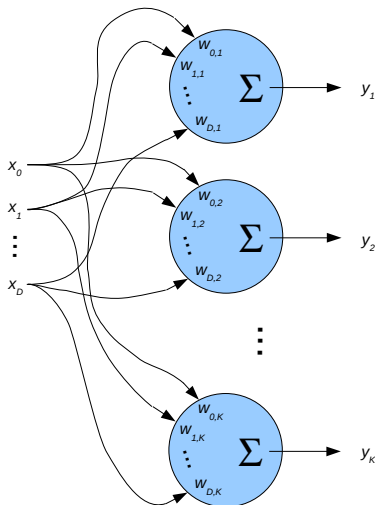
$$\mathbf{W} = (\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})^{-1} \tilde{\mathbf{X}}^T \mathbf{T}$$

$$\mathbf{Y} = \tilde{\mathbf{X}} \mathbf{W}$$

$\tilde{\mathbf{X}}$ is $N \times D + 1$

\mathbf{W} is $D + 1 \times K$

\mathbf{Y} is $N \times K$



$$\mathbf{W} = \begin{bmatrix} w_{0,1} & w_{0,2} & \cdots & w_{0,K} \\ w_{1,1} & w_{1,2} & \cdots & w_{1,K} \\ \vdots & \vdots & \ddots & \vdots \\ w_{D,1} & w_{D,2} & \cdots & w_{D,K} \end{bmatrix}$$

Can Make Nonlinear in Inputs

Transform X into $\Phi(X)$.

Minimize

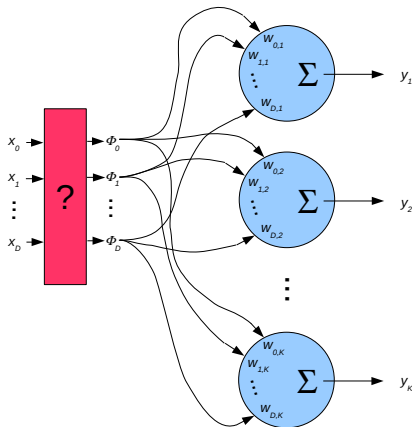
$$E_k = \sum_{n=1}^N (t_{n,k} - y(\Phi(\mathbf{x}_n), \mathbf{w}_k))^2$$

to get

$$\mathbf{W} = (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T \mathbf{T}$$

and use like

$$\mathbf{Y} = \tilde{\Phi} \mathbf{W}$$



Introduction

Structure

Training Algorithm

Gradient Descent
Derivation of Error
Gradient

Can Make Nonlinear in Inputs

Transform X into $\Phi(X)$.

Minimize

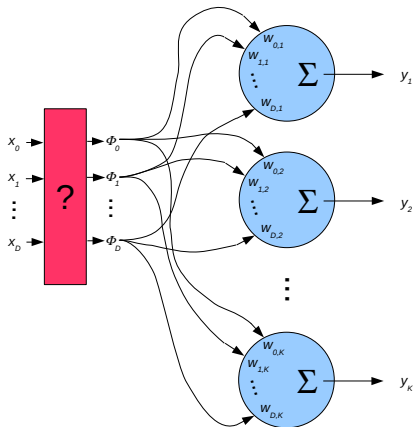
$$E_k = \sum_{n=1}^N (t_{n,k} - y(\Phi(\mathbf{x}_n), \mathbf{w}_k))^2$$

to get

$$\mathbf{W} = (\tilde{\Phi}^T \tilde{\Phi})^{-1} \tilde{\Phi}^T \mathbf{T}$$

and use like

$$\mathbf{Y} = \tilde{\Phi} \mathbf{W}$$



Now, what should we put in the red box?

Can we use the training data to figure this out?

Outline

Introduction

Structure

Training Algorithm

Gradient Descent

Derivation of Error Gradient

Introduction

Structure

Training Algorithm

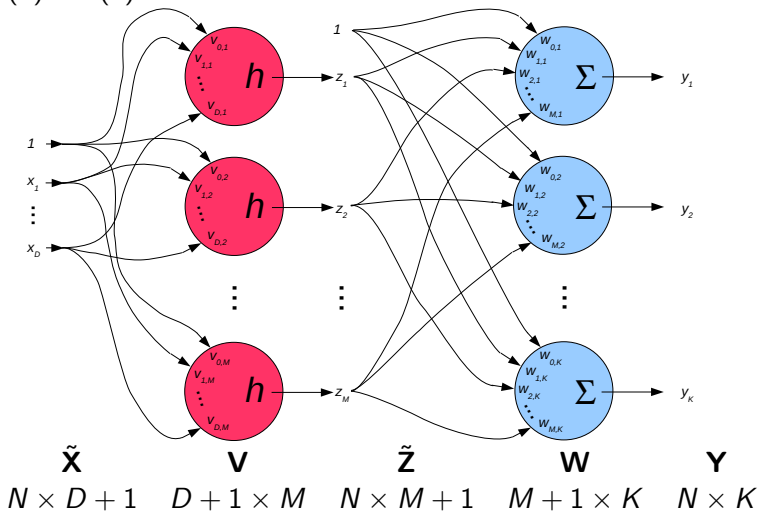
Gradient Descent

Derivation of Error

Gradient

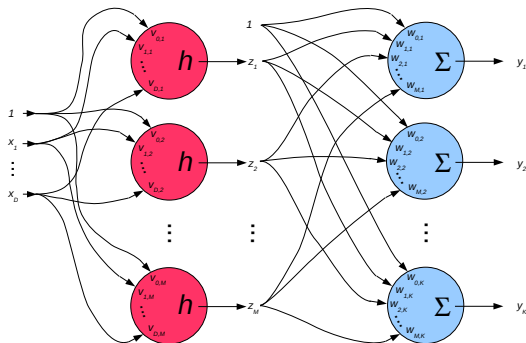
Two Layer Neural Network

We have now entered the world of neural networks, with $\phi(\mathbf{x}) = h(\mathbf{x})$.



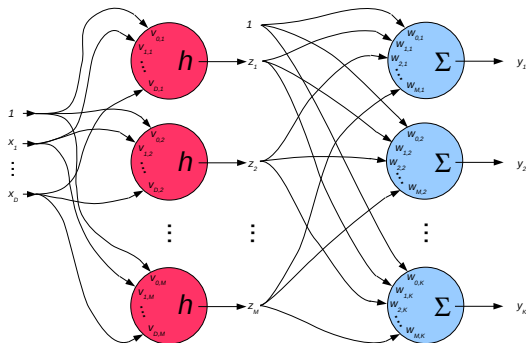
$$\mathbf{Z} = h(\tilde{\mathbf{X}}\mathbf{V}), \quad \mathbf{Y} = \tilde{h}(\mathbf{Z})\mathbf{W} \quad \text{or} \quad \mathbf{Z} = \tilde{h}(\tilde{\mathbf{X}}\mathbf{V})\mathbf{W}$$

What is h ?



- The two layers are called the “hidden” and “output” layer. h is the “activation function” for the units in the hidden layer.

What is h ?



- The two layers are called the “hidden” and “output” layer. h is the “activation function” for the units in the hidden layer.
- We will be doing gradient descent in the squared error, so want an h whose derivative doesn't grow as \mathbf{v} grows, and whose derivative is easy to calculate.

What is h ?

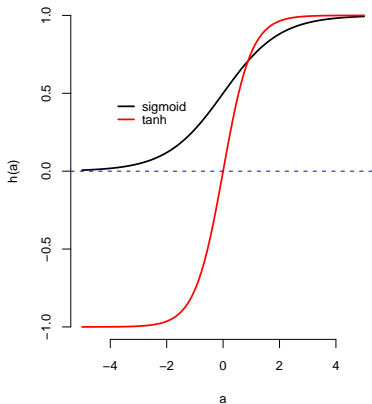
Two common choices,
where $a = \tilde{\mathbf{x}}^T \mathbf{v}$:

- sigmoid
(asymmetric)

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

- tanh (symmetric)

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



Outline

Introduction

Structure

Training Algorithm

Gradient Descent

Derivation of Error Gradient

Introduction

Structure

Training Algorithm

Gradient Descent

Derivation of Error
Gradient

Train by Gradient Descent in Mean Squared Error

- Mean Squared Error

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

Train by Gradient Descent in Mean Squared Error

- Mean Squared Error

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

- Gradient descent in mean squared error, using steepest descent

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \frac{\partial E}{\partial v_{j,m}}$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \frac{\partial E}{\partial w_{m,k}}$$

Train by Gradient Descent in Mean Squared Error

- Mean Squared Error

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

- Gradient descent in mean squared error, using steepest descent

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \frac{\partial E}{\partial v_{j,m}}$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \frac{\partial E}{\partial w_{m,k}}$$

- Often presented as $\rho_h = \rho_o$, but having different rates in the two layers often helps convergence rate.

Train by Gradient Descent in Mean Squared Error

- Mean Squared Error

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

- Gradient descent in mean squared error, using steepest descent

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \frac{\partial E}{\partial v_{j,m}}$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \frac{\partial E}{\partial w_{m,k}}$$

- Often presented as $\rho_h = \rho_o$, but having different rates in the two layers often helps convergence rate.
- Will this find the global optimum (the values of v and w that minimize the mean squared error)?

Train by Gradient Descent in Mean Squared Error

- Mean Squared Error

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

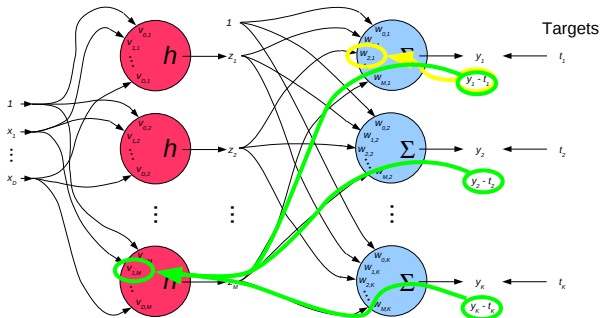
- Gradient descent in mean squared error, using steepest descent

$$v_{j,m} \leftarrow v_{j,m} - \rho_h \frac{\partial E}{\partial v_{j,m}}$$

$$w_{m,k} \leftarrow w_{m,k} - \rho_o \frac{\partial E}{\partial w_{m,k}}$$

- Often presented as $\rho_h = \rho_o$, but having different rates in the two layers often helps convergence rate.
- Will this find the global optimum (the values of v and w that minimize the mean squared error)?
- No. Just a local minimum.

Back-Propagation of Error Information



$$v_{j,m} \leftarrow v_{j,m} - \rho h \frac{\partial E}{\partial v_{j,m}} \quad w_{m,k} \leftarrow w_{m,k} - \rho o \frac{\partial E}{\partial w_{m,k}}$$

With a bit of calculus and algebra we arrive at

$$v_{j,m} \leftarrow v_{j,m} - \rho h \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k=1}^K (y_{n,k} - t_{n,k}) w_{m,k} (1 - \tilde{z}_m^2) \tilde{x}_{n,j}$$

$$w_{m,k} \leftarrow w_{m,k} - \rho o \frac{1}{N} \sum_{n=1}^N (y_{n,k} - t_{n,k}) \tilde{z}_m$$

First work on $\frac{\partial E}{\partial w_{m,k}}$

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

$$\frac{\partial E}{\partial w_{m,k}} = 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \frac{\partial y_{n,k'}}{\partial w_{m,k}}$$

Since $y_{n,k'} = \sum_{m'=0}^M w_{m',k'} \tilde{z}_{n,m'}$

$$\frac{\partial E}{\partial w_{m,k}} = 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \frac{\partial \left(\sum_{m'=0}^M w_{m',k'} \tilde{z}_{n,m'} \right)}{\partial w_{m,k}}$$

$$= 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N (y_{n,k} - t_{n,k}) \tilde{z}_{n,m}$$

Now the hard one, $\frac{\partial E}{\partial v_{j,m}}$

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

$$\frac{\partial E}{\partial v_{j,m}} = 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \frac{\partial y_{n,k'}}{\partial v_{j,m}}$$

Since $y_{n,k'} = \sum_{m'=0}^M w_{m',k'} \tilde{z}_{n,m'}$

$$= \sum_{m'=0}^M w_{m',k'} \tilde{h} \left(\sum_{j'=0}^D v_{j',m'} \tilde{x}_{n,j'} \right)$$

$$\frac{\partial E}{\partial v_{j,m}} = 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \frac{\partial \left(\sum_{m'=0}^M w_{m',k'} \tilde{h} \left(\sum_{j'=0}^D v_{j',m'} \tilde{x}_{n,j'} \right) \right)}{\partial v_{j,m}}$$

Let $a_{n,m'} = \tilde{h} \left(\sum_{j'=0}^D v_{j',m'} \tilde{x}_{n,j'} \right)$

$$\frac{\partial E}{\partial v_{j,m}} = 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \sum_{m'=0}^M w_{m',k'} \frac{\partial \tilde{h}(a_{n,m'})}{\partial a_{n,m'}} \frac{\partial a_{n,m'}}{\partial v_{j,m}}$$

$$= 2 \frac{1}{N} \frac{1}{K} \sum_{n=1}^N \sum_{k'=1}^K (y_{n,k'} - t_{n,k'}) \sum_{m'=0}^M w_{m',k'} \frac{\partial \tilde{h}(a_{n,m'})}{\partial a_{n,m'}} \tilde{x}_{n,j}$$