Linear and Logistic Regression as Artificial Neural Networks

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Outline

1. Introduction

NN Structure, Activation/Score Functions, Estimation

2. Linear Regression and Neural Nets

Simple & Multivariate with Gradient Descent

3. Non-Linear Regression and Neural Nets

Simple & Multivariate with Gradient Descent

4. Logistic Regression and Neural Nets

Simple & Multivariate with Gradient Descent

5. Multi-Layer Deep Neural Nets

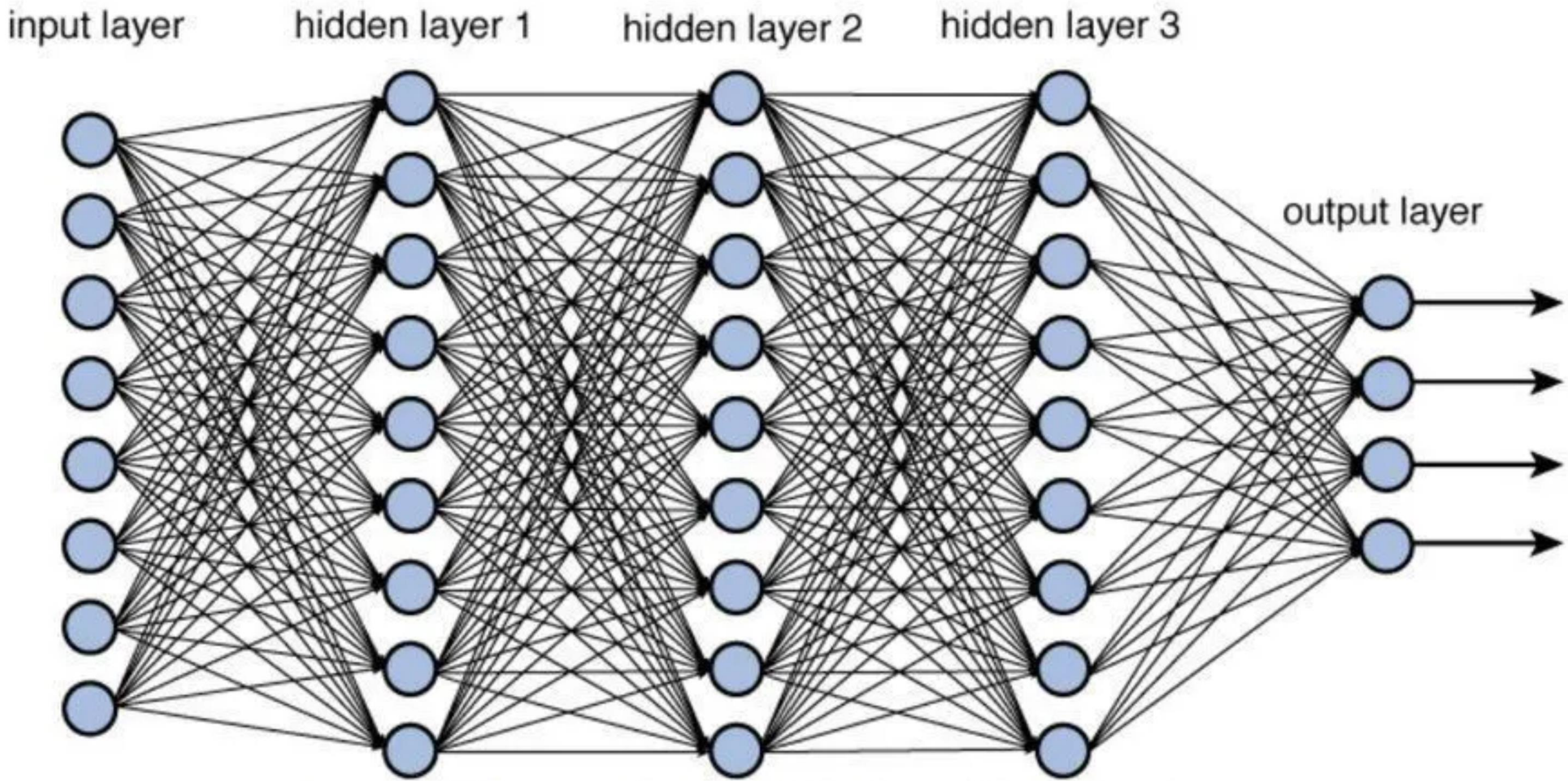
Two or More Layers

6. Discussion

More to learn hands on.

1. Introduction

There are often illustrations, but no details of mathematics.

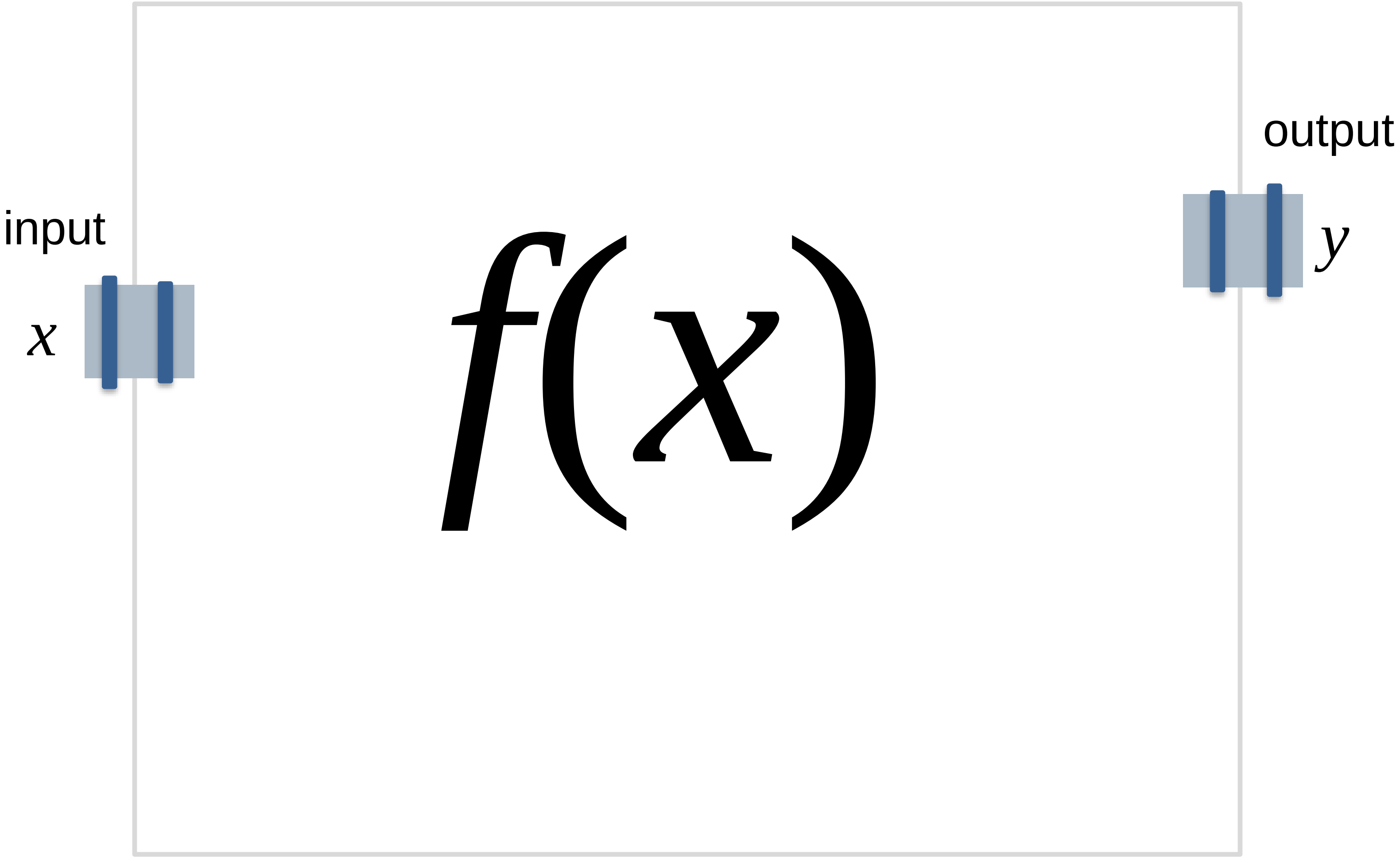


This leaves the science out of Data Science and results in a Data Artist.

https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

1. Introduction

Assume we want to learn the relationship between x and y.



Sample

outputs	inputs
y_1	x_1
y_2	x_2
y_3	x_3
y_4	x_4
y_5	x_5
y_6	x_6
y_7	x_7
y_8	x_8
y_9	x_9
y_{10}	x_{10}

1. Introduction

coefficient

$j = 0, \ldots, q$

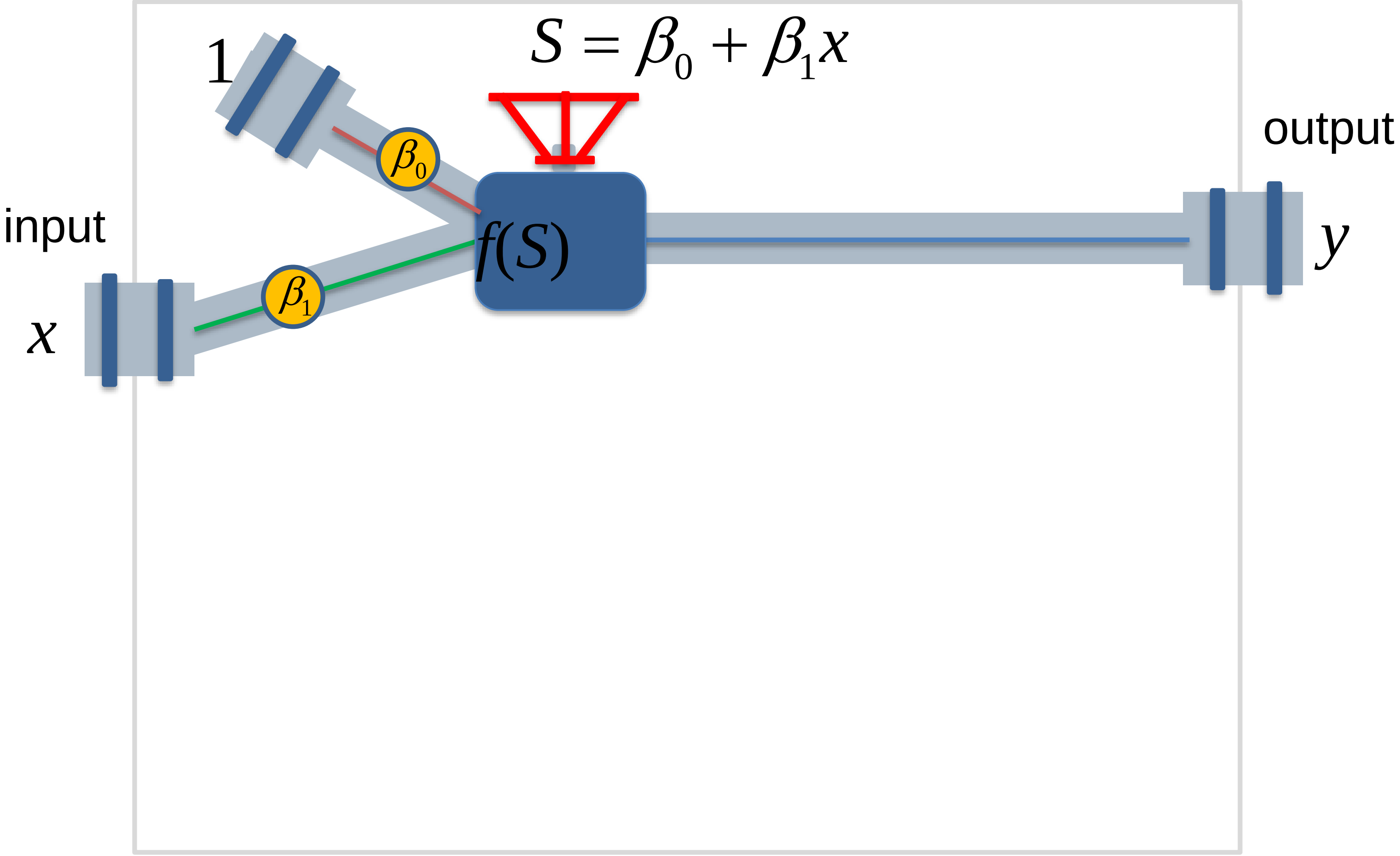
β_j $\quad f(S)$

$$p(S) = \frac{1}{1+e^{-S}}$$

Single input and single output.

functional flow valve

$$S = \beta_0 + \beta_1 x$$



Well known solutions and interpretations.

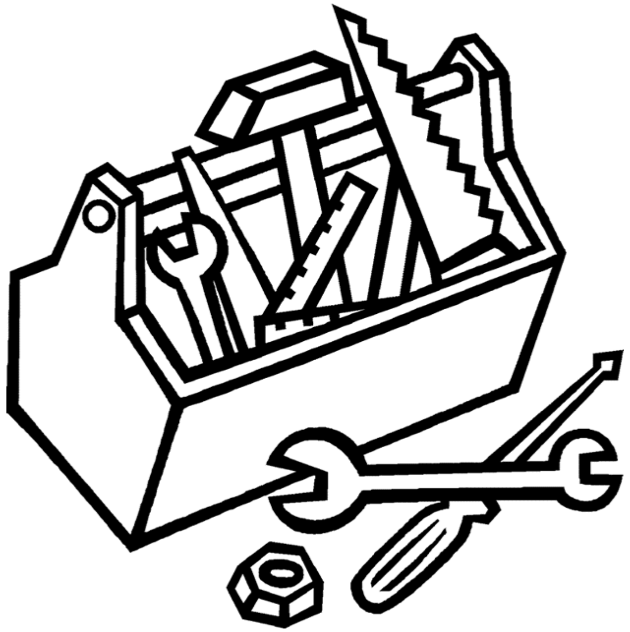
Functional Form

$$y = S$$

or

$$y = \begin{cases} 1 & \text{w/ probability } p(S) \\ 0 & \text{w/ probability } 1\text{-}p(S) \end{cases}$$

Many Tools

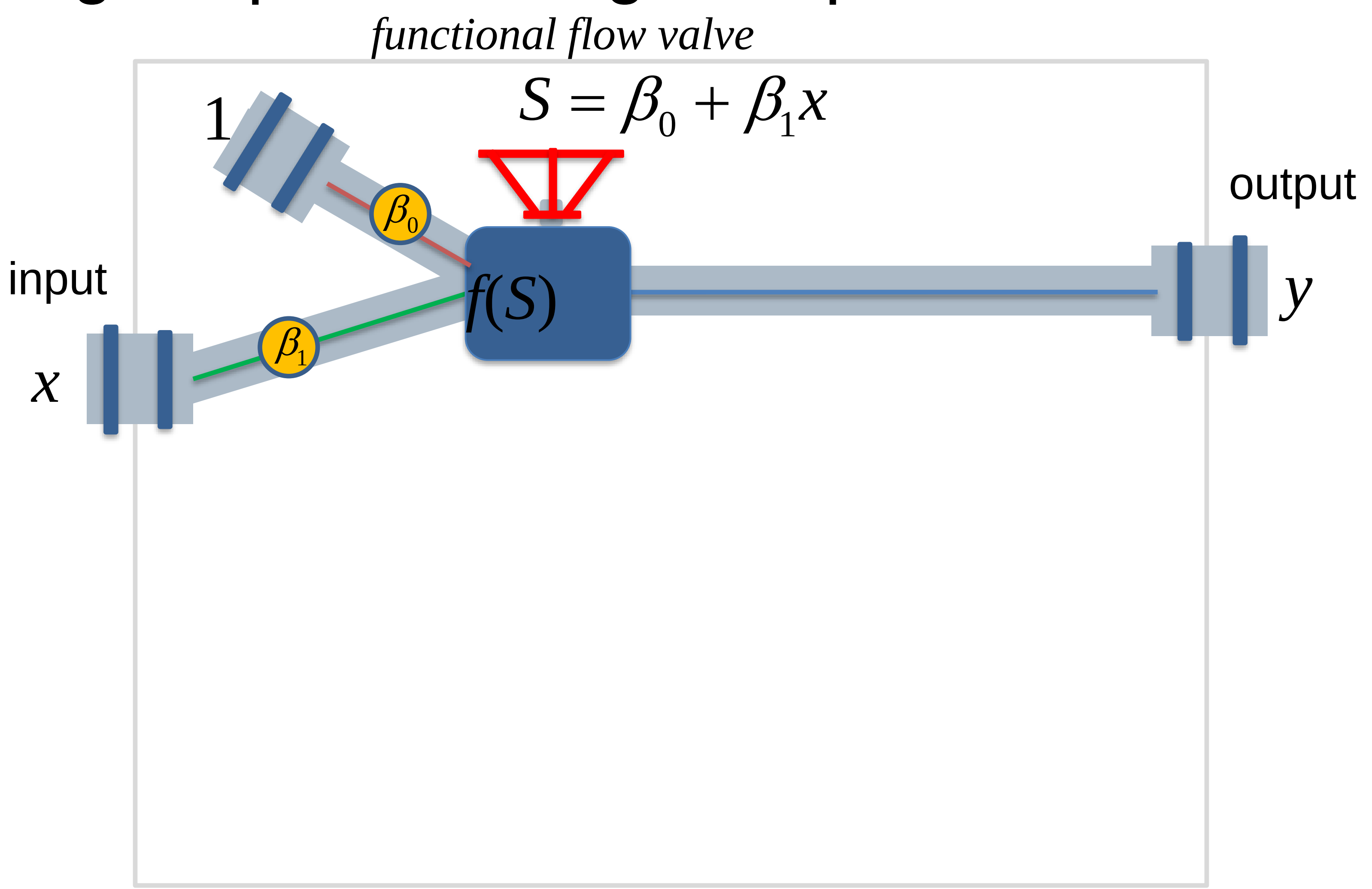


1. Introduction

coefficient
$j = 0, \ldots, q$

β_j $\quad f(S)$ $\quad p(S) = \dfrac{1}{1 + e^{-S}}$

Single input and single output.



Functional Form

$$y = S$$

or

$$y = \begin{cases} 1 & \text{w/ probability } p(S) \\ 0 & \text{w/ probability } 1\text{-}p(S) \end{cases}$$

Objective Function

$$Q(\beta_0, \beta_1)$$

Estimate Parameters

$$(\hat{\beta}_0, \hat{\beta}_1)$$

Well known solutions and interpretations.

1. Introduction

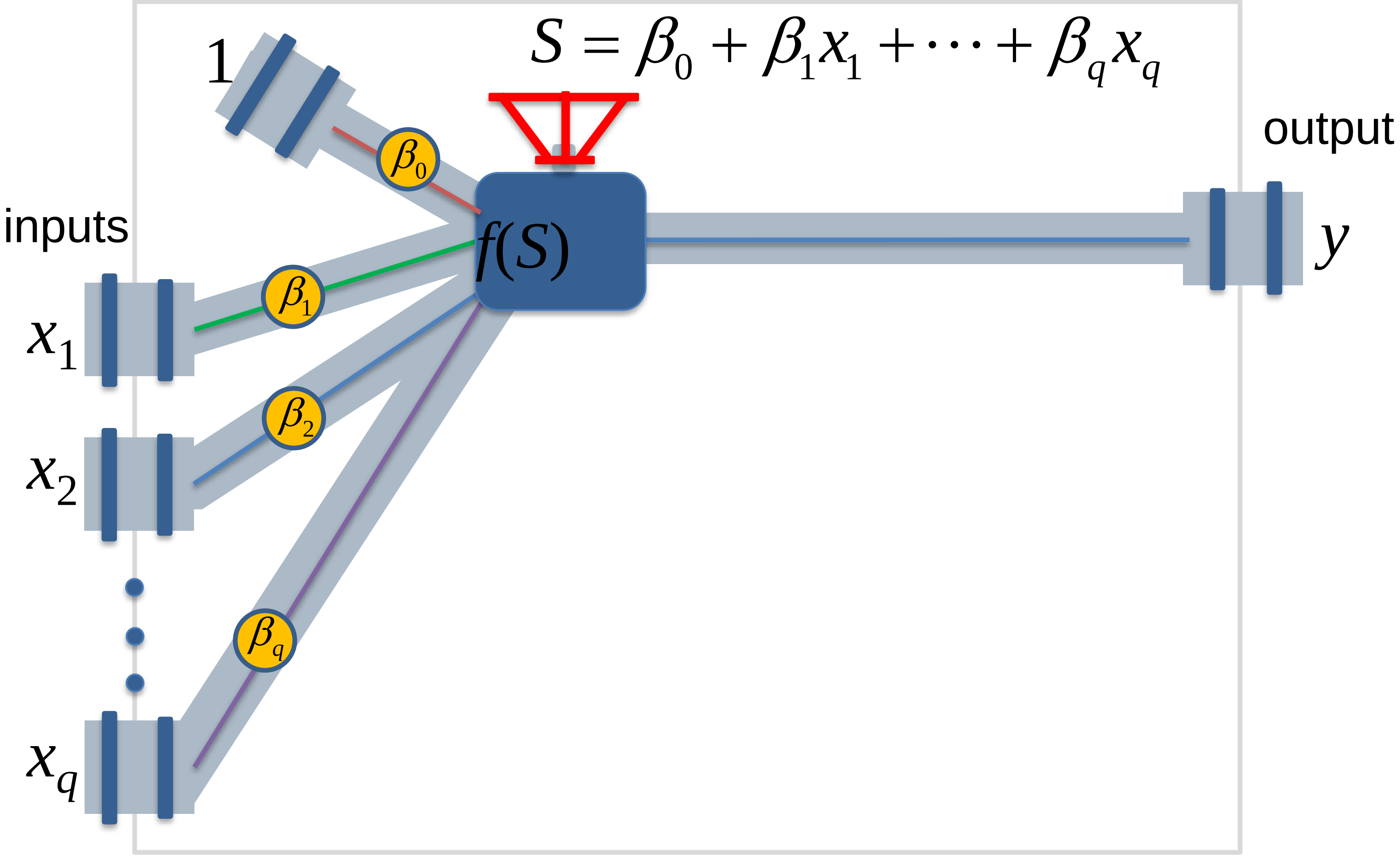
coefficient $j=0,\ldots,q$ $\quad \beta_j \quad f(S)$

$$p(S)=\frac{1}{1+e^{-S}}$$

Multiple inputs and single output.

functional flow valve

$$S=\beta_0+\beta_1 x_1+\cdots+\beta_q x_q$$



Well known solutions and interpretations.

Functional Form

$$y=S$$

or

$$y=\begin{cases}1 & \text{w/ probability } p(S)\\ 0 & \text{w/ probability 1-}p(S)\end{cases}$$

Objective Function

$$Q(\beta_0,\ldots,\beta_q)$$

Estimate Parameters

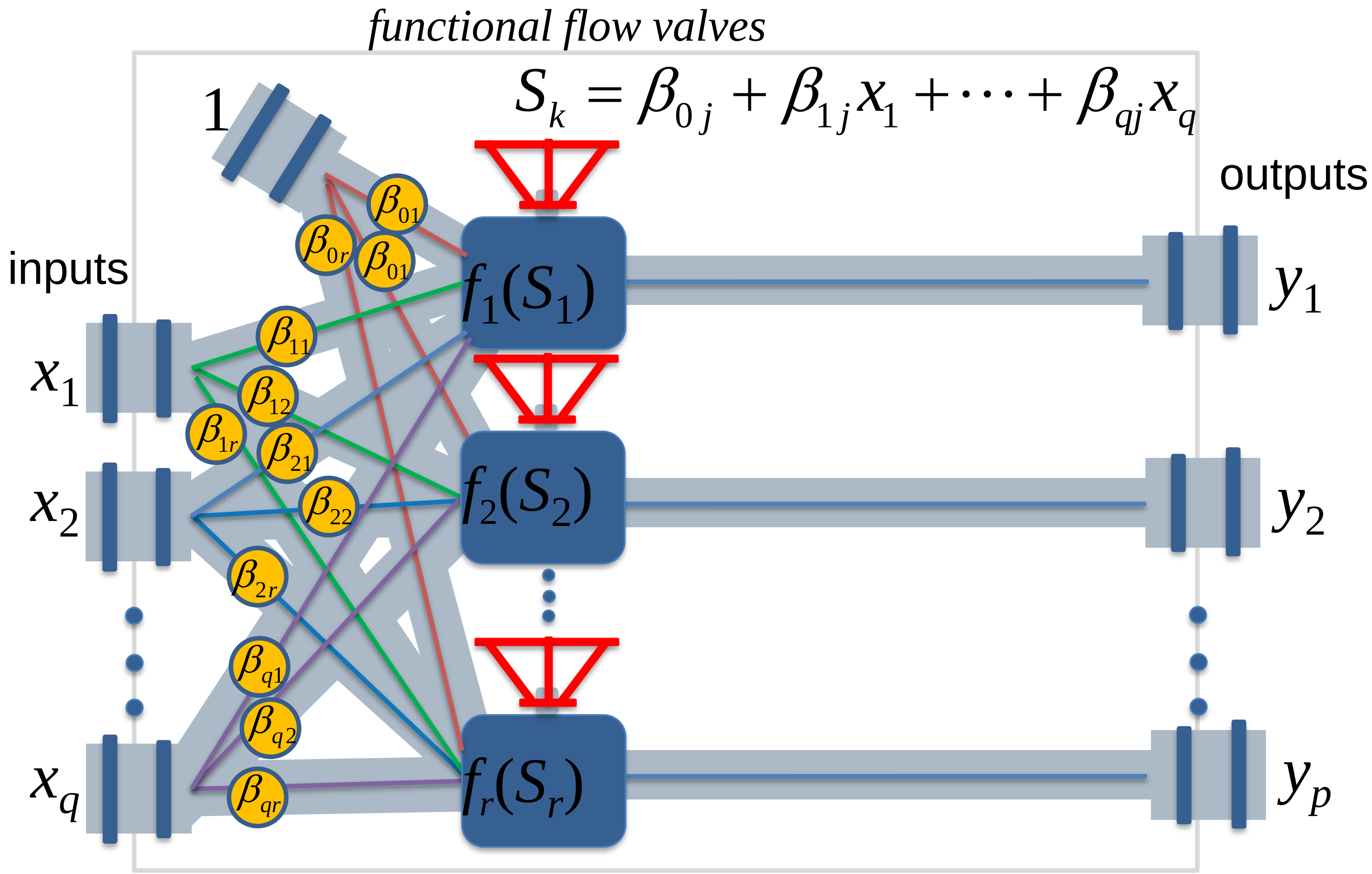
$$(\hat{\beta}_0,\ldots,\hat{\beta}_q)$$

1. Introduction

coefficient $j = 0,\ldots,q$ node $k = 1,\ldots,r$

β_{jk} $f_k(S_k)$

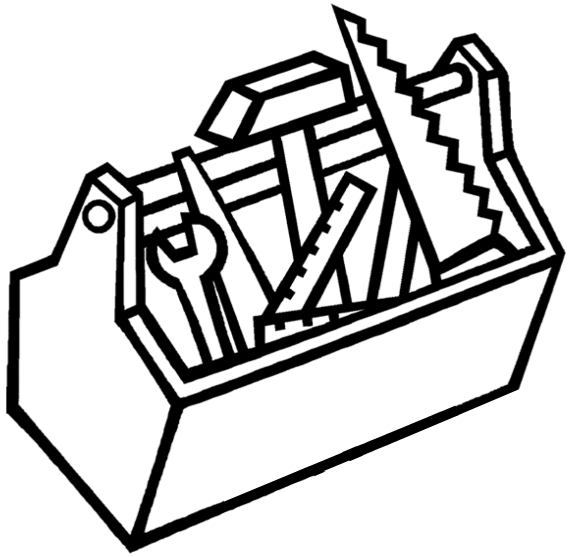
Multiple inputs and multiple outputs.



Well known solutions and interpretations.

Functional Form
Multivariate Linear Regression
Other nonlinear functions
Multinomial Logistic…

Fewer Tools

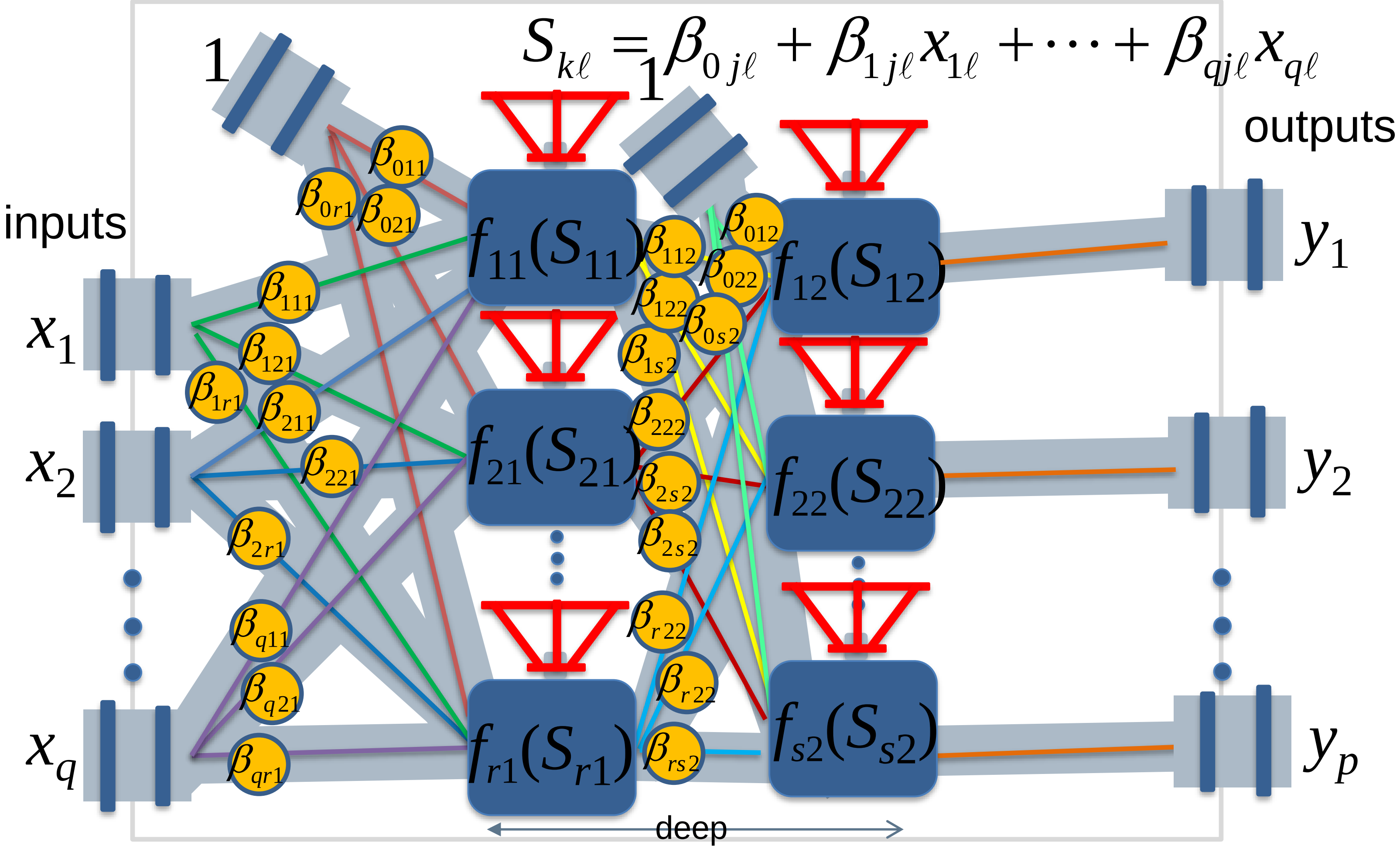


1. Introduction

coefficient $j=0,\ldots,q$ node $k=1,\ldots,r$ layer $\ell=1,2$ $\beta_{jk\ell}$ $f_{k\ell}(S_{k\ell})$

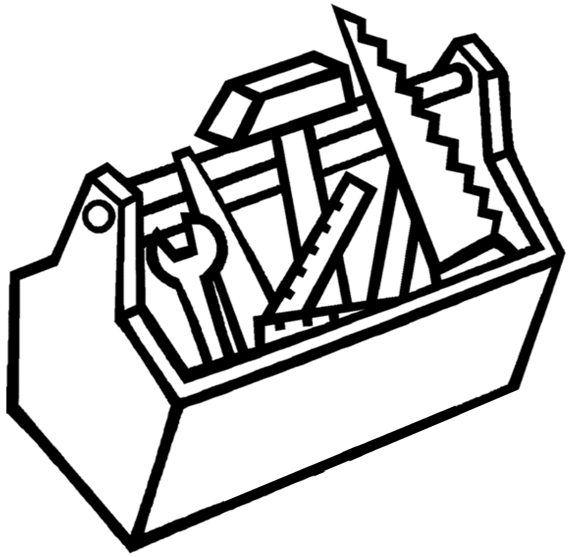
Multiple inputs and multiple outputs.

functional flow valves



Functional Form
Multivariate Linear Regression
Other nonlinear functions
Multinomial Logistic…

Fewer Tools

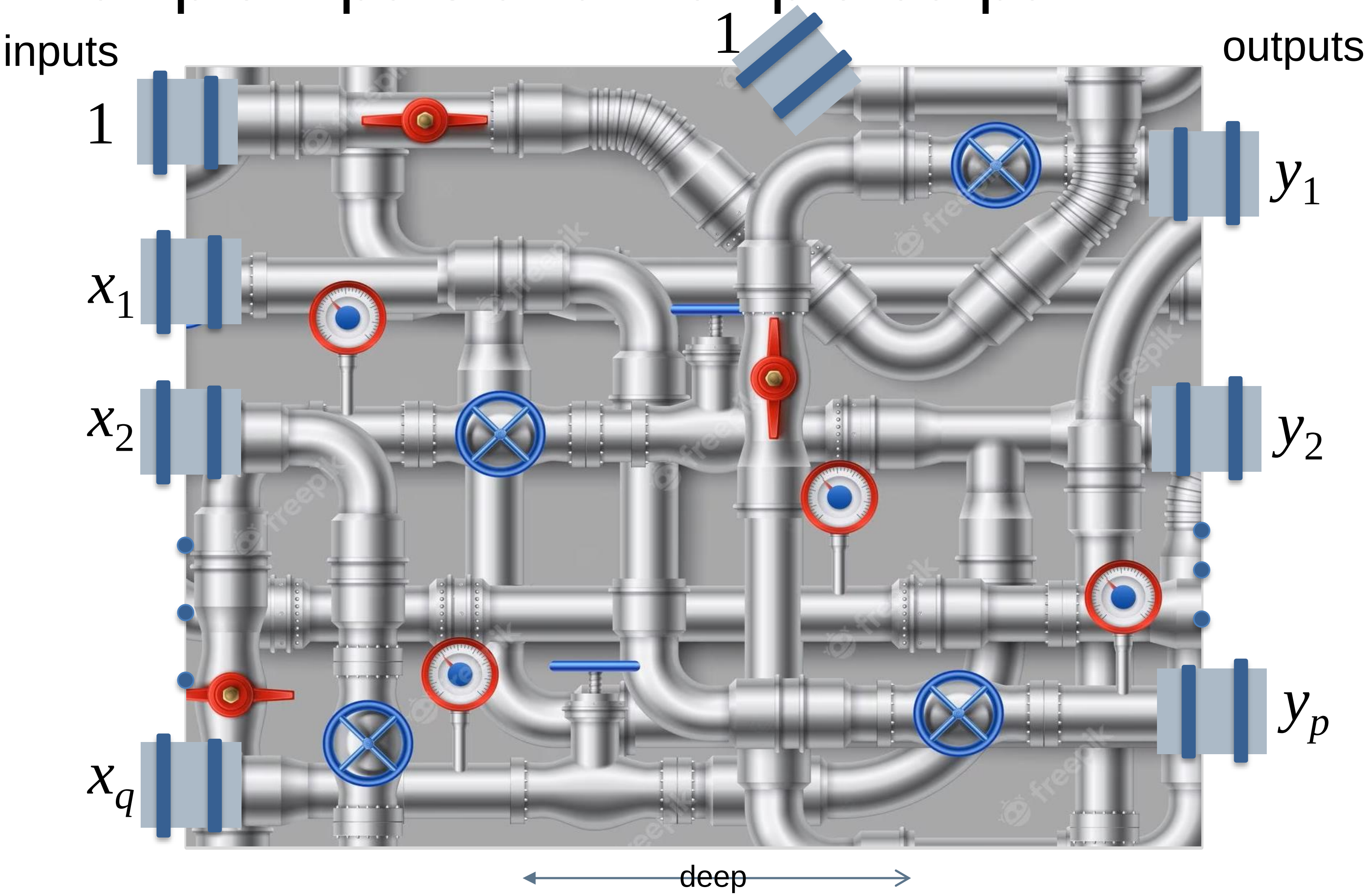


Well known solutions and interpretations.

Many interconnected valves. Functions of functions of … of inputs.

1. Introduction

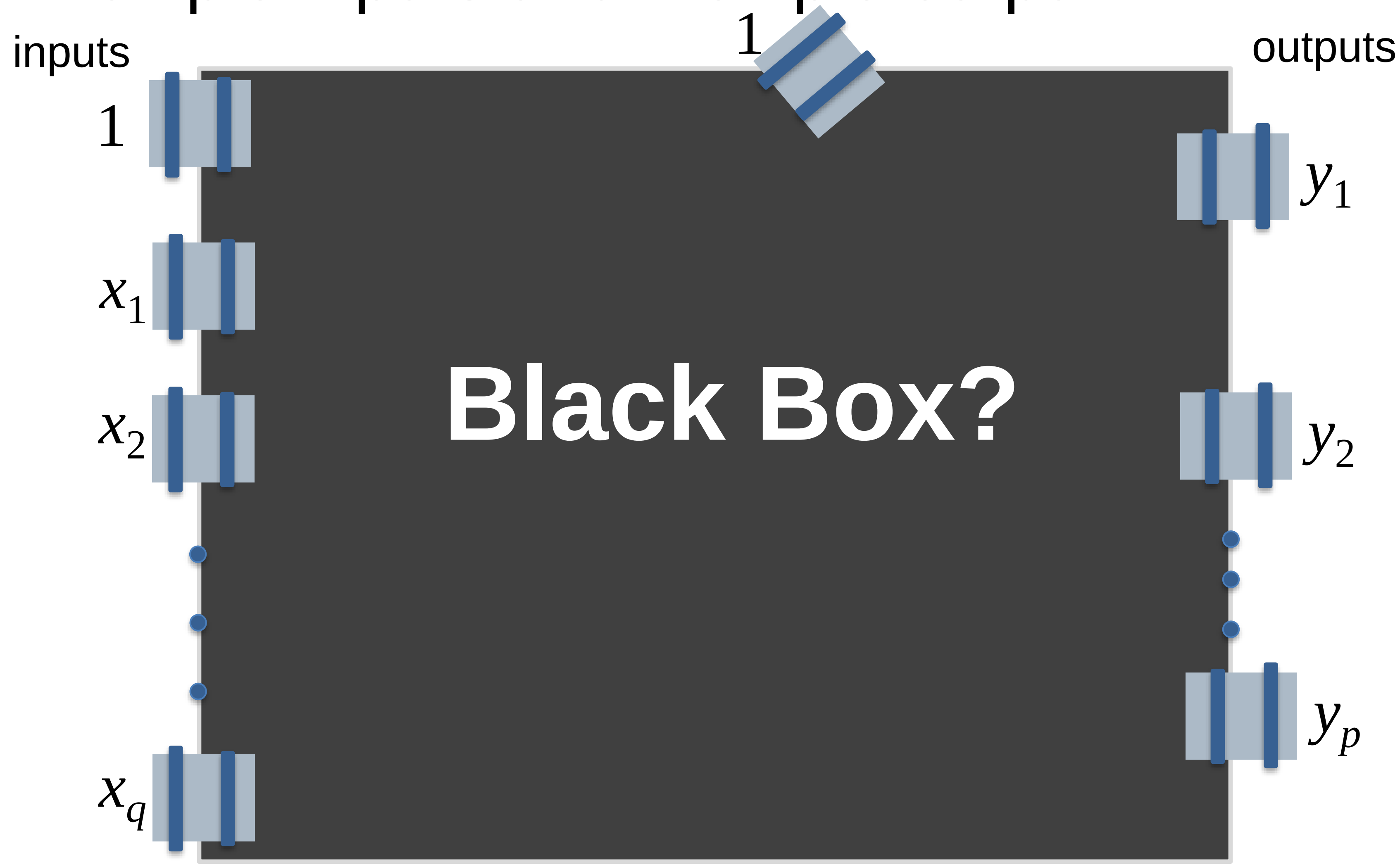
Multiple inputs and multiple output.



Many interconnected valves. Functions of functions of … of inputs.

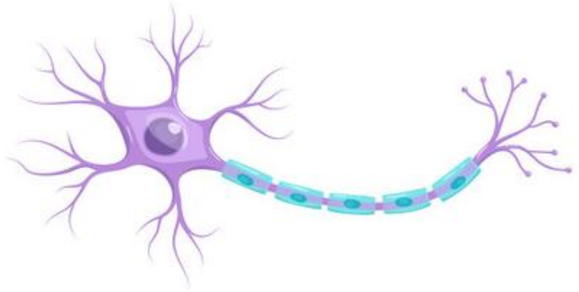
1. Introduction

Multiple inputs and multiple output.

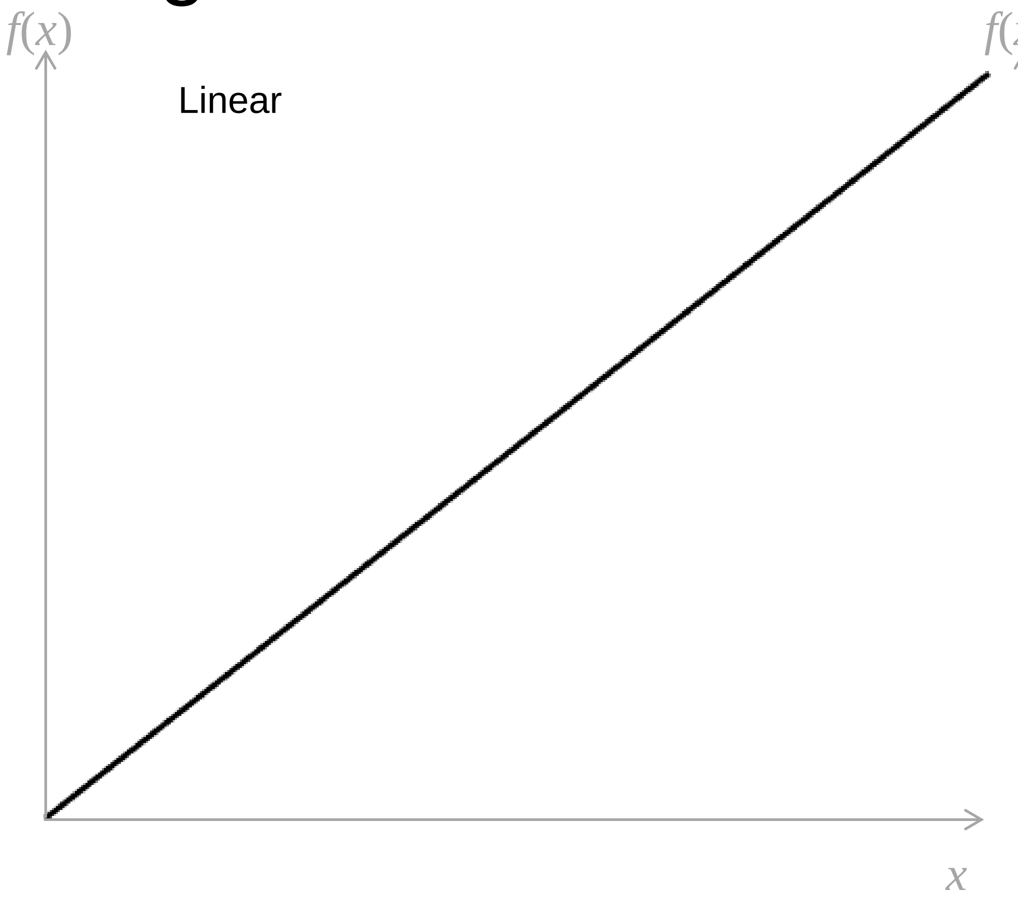


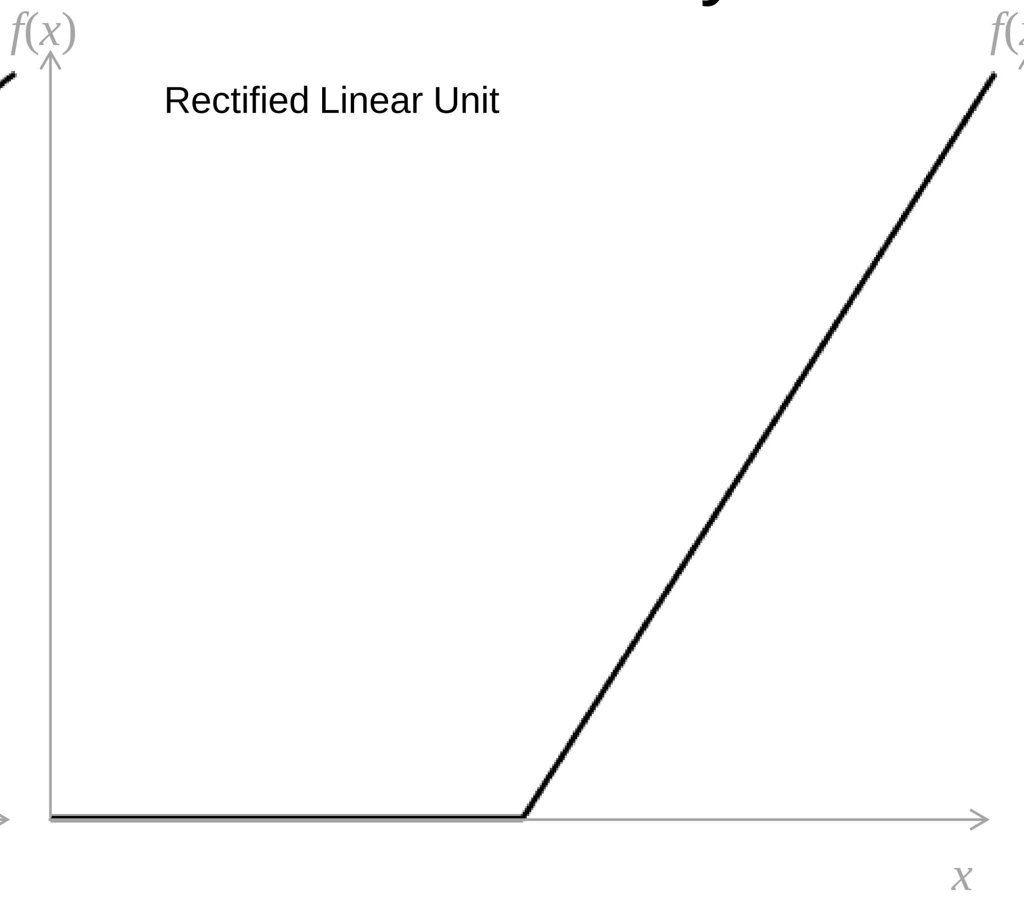
Although you put all the pieces in the box, it is extremely complicated to understand and interpret. We will examine the basic structure.

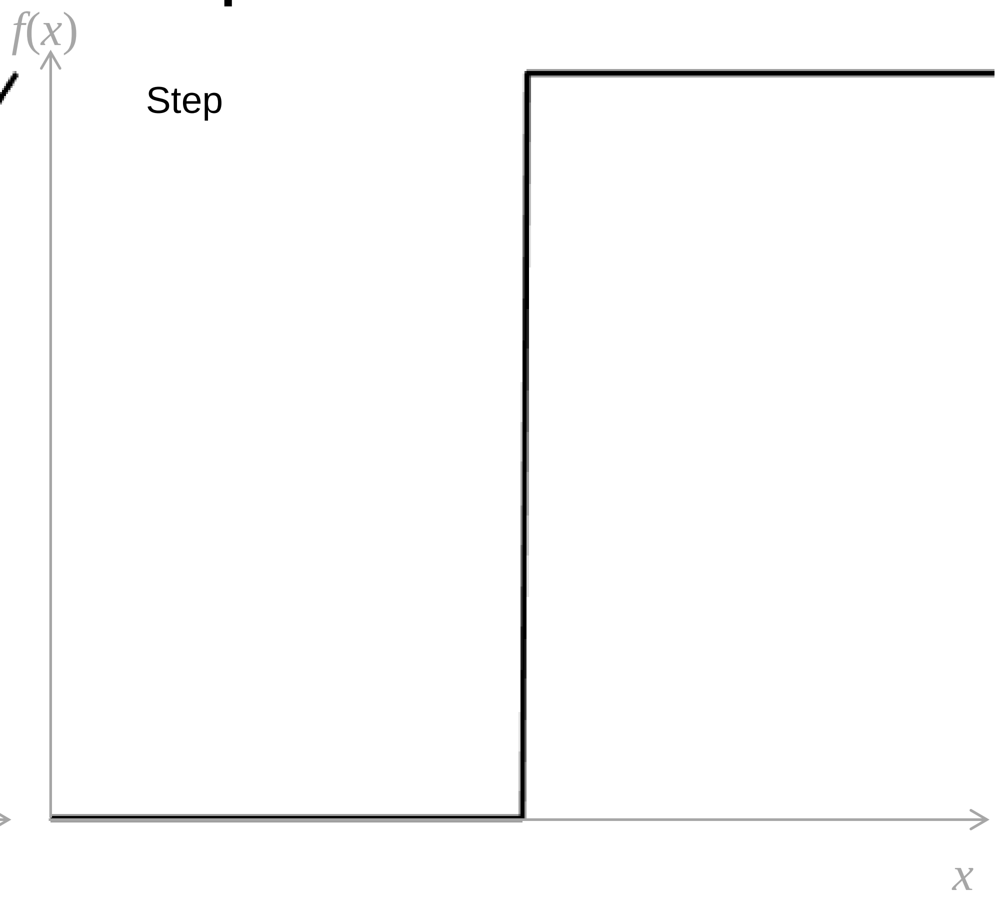
1. Introduction



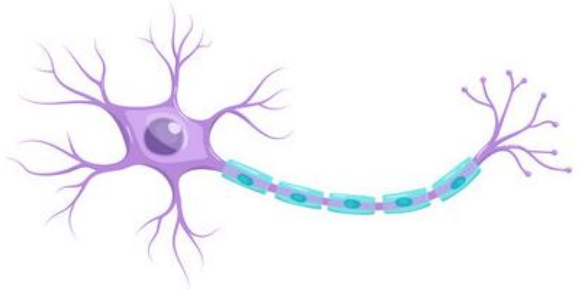
There are many activation functions $f(\cdot)$ from (non)linear regression, but most used are motivated by neuronal representations.



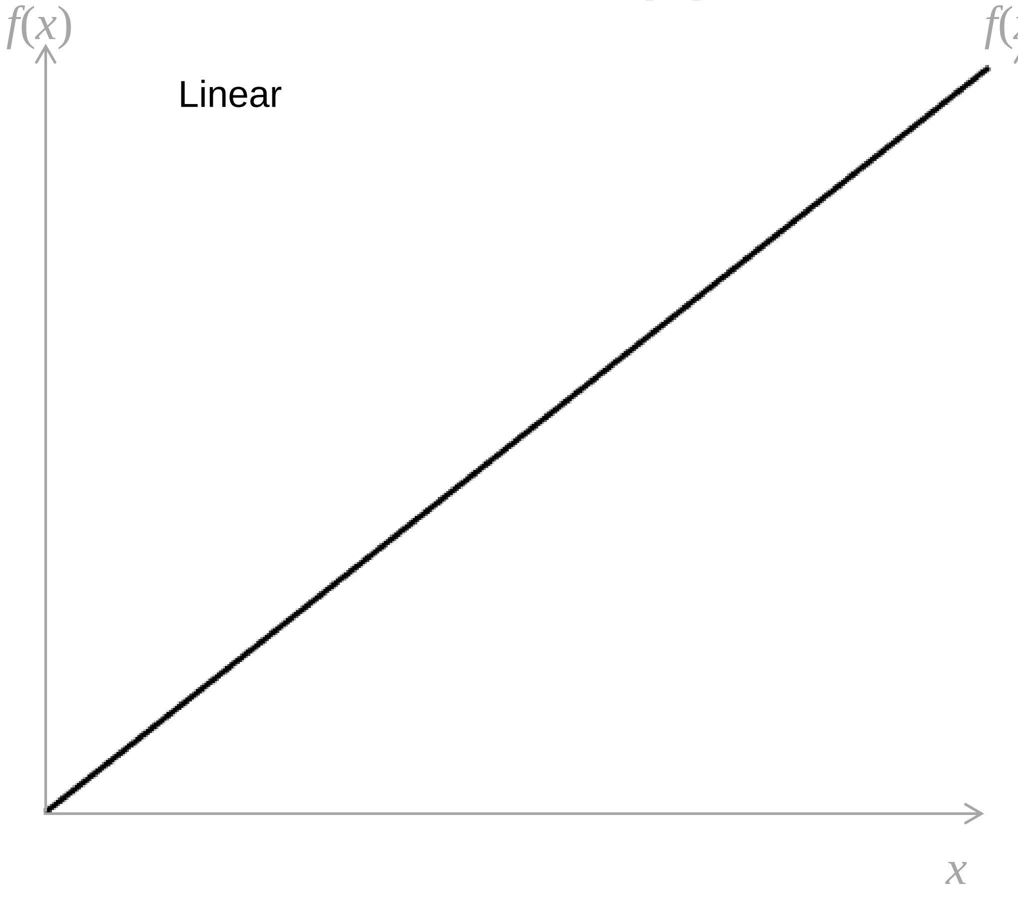




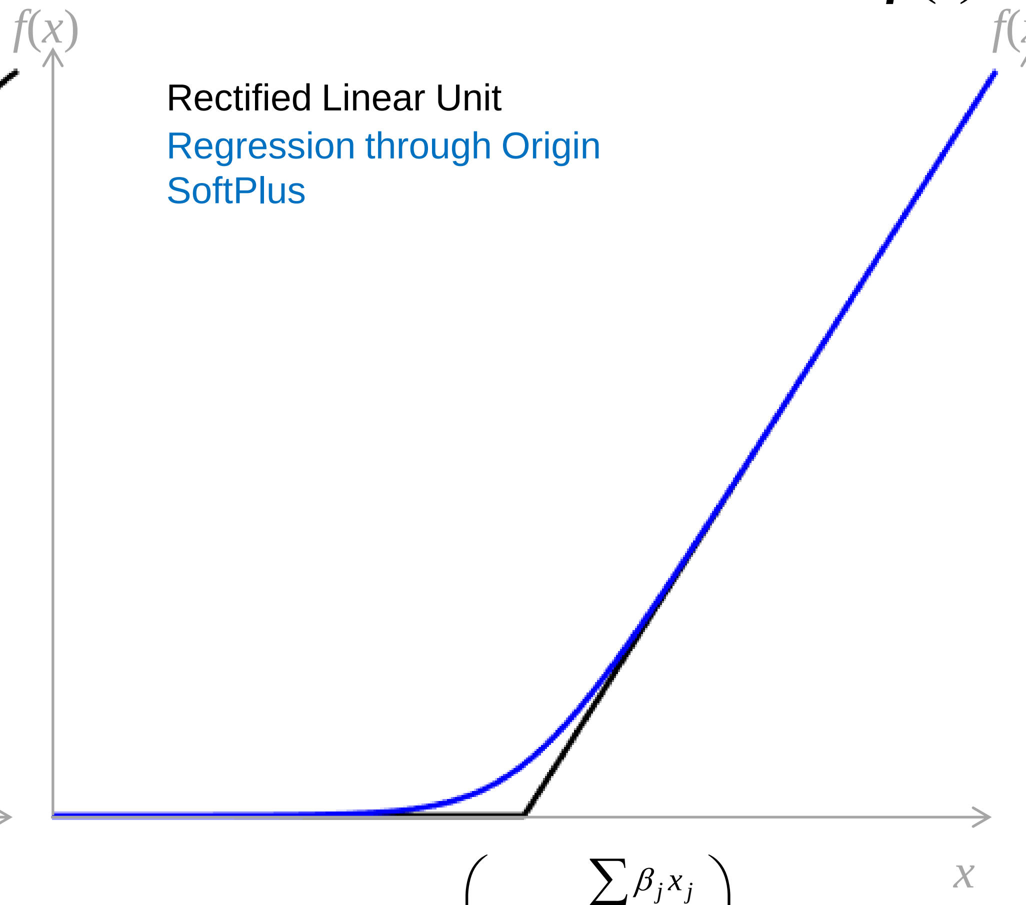
1. Introduction



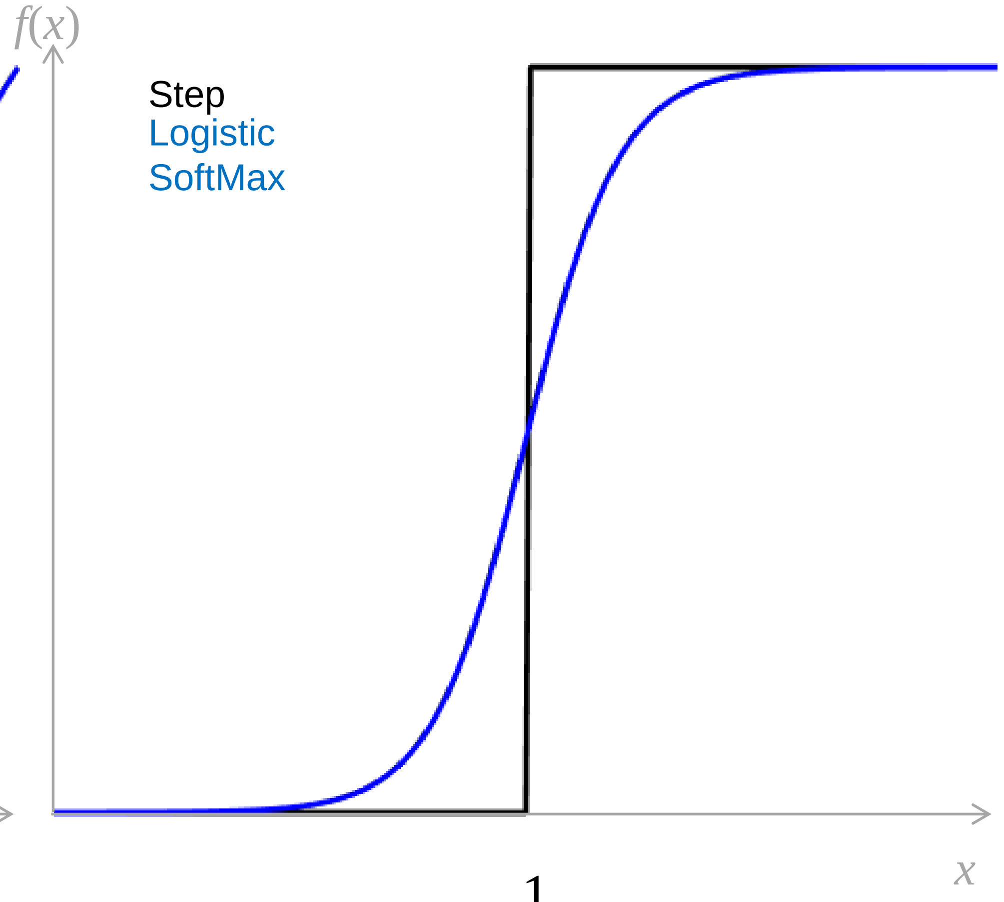
Step and ReLU not differentiable for optimization so smooth differentiable approximate activation functions, $f(\cdot)$ are used.



$$f(\cdot)=\sum_j \beta_j x_j$$



$$f(\cdot)=\ln\left(1+e^{\sum_j \beta_j x_j}\right)$$



$$f(\cdot)=\frac{1}{1+e^{-\sum_j \beta_j x_j}}$$

1. Introduction

$$S_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_q x_{qi}$$

The form of the function depends on the type of input/output variable.

Distribution	Activation	Likelihood
Continuous Real Valued $y_i \sim Normal(f(S_i),\sigma^2)$	Linear $f(S_i) = S_i$	Normal $L(\beta_1,\ldots,\beta_q) = (2\pi)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}\sum_i (y_i - f(S_i))^2\right]$
Continuous Real Valued (Not focusing on.) $y_i \sim Normal(f(S_i),\sigma^2)$	SoftPlus $f(S_i) = \ln\left(1+e^{S_i}\right)$	Normal $L(\beta_1,\ldots,\beta_q) = (2\pi)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}\sum_i (y_i - f(S_i))^2\right]$
Discrete 0/1 Valued $y_i \sim Bernoulli(p(S_i))$	Logistic $p(S_i) = \frac{1}{1+e^{-S_i}}$	Bernoulli $L(\beta_0,\ldots,\beta_q) = \prod_i [p(S_i)]^{y_i}[1-p(S_i)]^{1-y_i}$

These are motivated from statistics!

1. Introduction

$$S_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_q x_{qi} = \sum_j \beta_j x_{ji}$$

We generally take the natural log of the likelihood for the score function.

Distribution	Activation	Score/Objective Function
Continuous Real Valued $y_i \sim Normal(f(S_i), \sigma^2)$	Linear $f(S_i) = S_i$	Least Squares $Q = \frac{1}{n}\sum_i [y_i - \sum_j \beta_j x_{ji}]^2$
Continuous Real Valued (Not focusing on.) $y_i \sim Normal(f(S_i), \sigma^2)$	SoftPlus $f(S_i) = \ln\left(1 + e^{S_i}\right)$	Least Squares $Q = \frac{1}{n}\sum_i [y_i - \ln(1 + \exp(\sum_j \beta_j x_{ji}))]^2$
Discrete 0/1 Valued $y_i \sim Bernoulli(p(S_i))$	Logistic $p(S_i) = \frac{1}{1 + e^{-S_i}}$	Bernoulli $Q = \sum_i y_i (\sum_j \beta_j x_{ji}) - \sum_i \ln[1 + \exp(\sum_j \beta_j x_{ji})]$

And optimize the score function using “training” data $x_1, \ldots, x_n$!

1. Introduction

$$S_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_q x_{qi}$$
$$= \sum_j \beta_j x_{ji}$$

We can differentiate the score function for optimization.

Distribution | **Activation** | **Score/Objective Function**

Least Squares-Linear

$$Q = \frac{1}{n}\sum_i [y_i - \sum_j \beta_j x_{ji}]^2$$

Derivative

$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n}\sum_i [y_i - \sum_j \beta_j x_{ij}](-x_{ij})$$

Least Squares-SoftPlus
(Not focusing on.)

$$Q = \frac{1}{n}\sum_i [y_i - \ln(1+\exp(\sum_j \beta_j x_{ji}))]^2$$

Derivative

$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n}\sum_i \left[\frac{x_{ij}[y_i - \ln(1+\exp(\sum_j \beta_j x_{ij}))]\exp(\sum_j \beta_j x_{ij})}{1+\exp(\sum_j \beta_j x_{ij})}\right]$$

Bernoulli-Logistic

$$Q = \sum_i y_i(\sum_j \beta_j x_{ji}) - \sum_i \ln[1+\exp(\sum_j \beta_j x_{ji})]$$

Derivative

$$\frac{\partial Q}{\partial \beta_j} = \sum_i x_{ij} y_i - \sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_j x_{ij})}$$

And optimize the score function using “training” data $x_1,\ldots,x_n$!

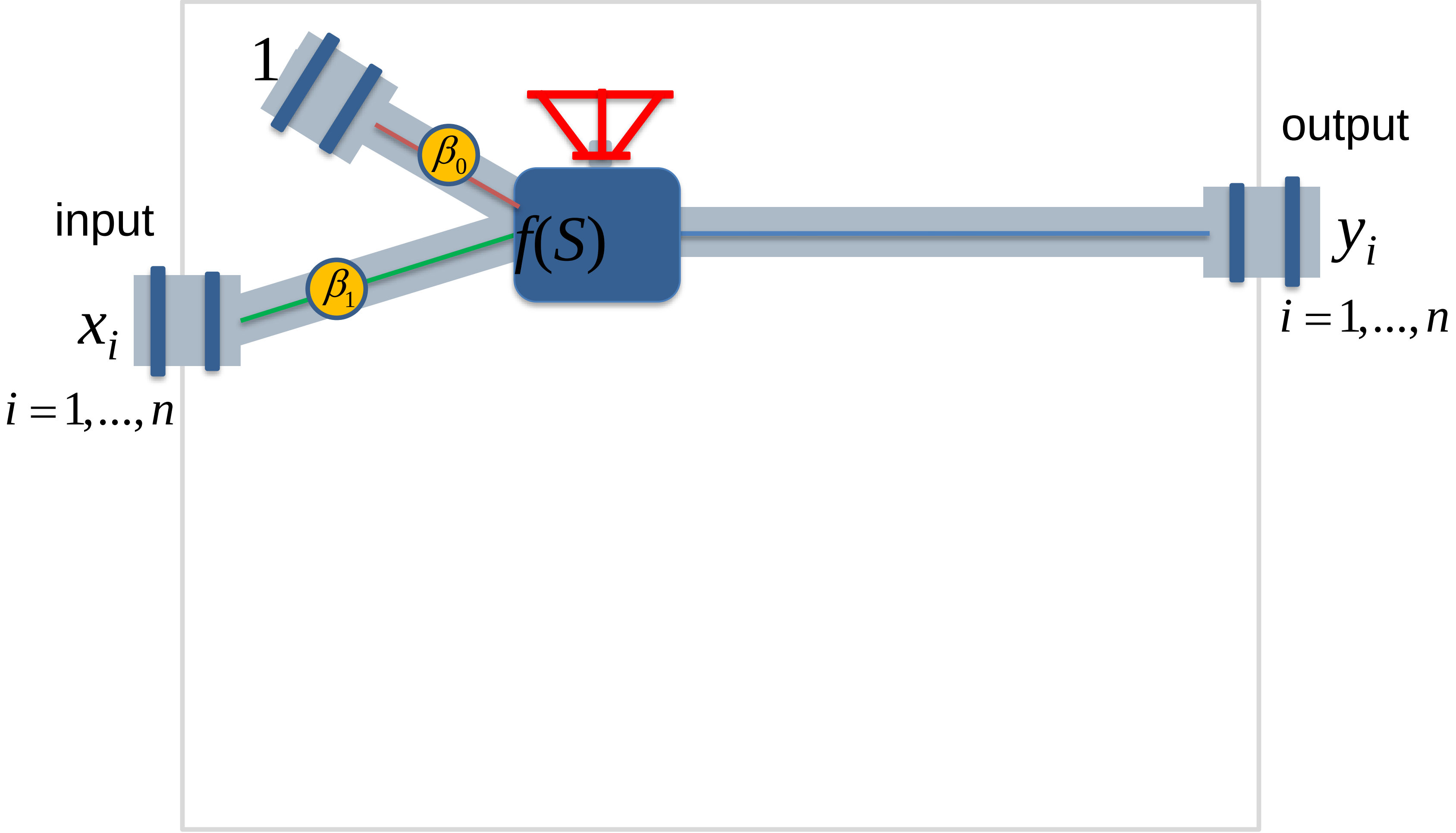
1. Introduction

coefficient
$j=0,\ldots,q$

β_j $f(S)$ $S=\sum_j \beta_j x_{ji}$

Single input and single output.

functional flow valve



1) Start with initial t=0 values $(\hat{\beta}_0^{(t)},\hat{\beta}_1^{(t)})$
2) Run n data through

$$Q_i^{(t)}=[y_i-f(\sum_j \hat{\beta}_j^{(t)}x_{ij})]^2$$
$i=1,\ldots,n$

3) Calculate score function

$$Q^{(t)}=\frac{1}{n}\sum_i [y_i-f(\sum_j \hat{\beta}_j^{(t)}x_{ij})]^2$$

4) Update coefficients GD

$(\hat{\beta}_0^{(t+1)},\hat{\beta}_1^{(t+1)})$ ∇Q

5) Return to Step 2.

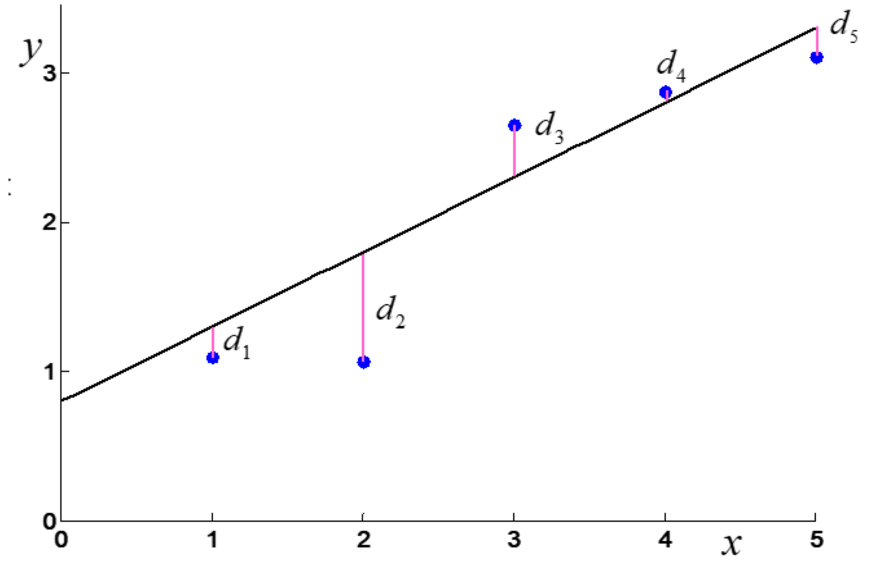
A well known general numerical solution.

2. Linear Regression and Neural Nets

Often we believe that there is a linear relationship between an independent variable x and a dependent variable y with measurement error.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

$$i = 1, \ldots, n$$



Could assume normal error or use least squares.

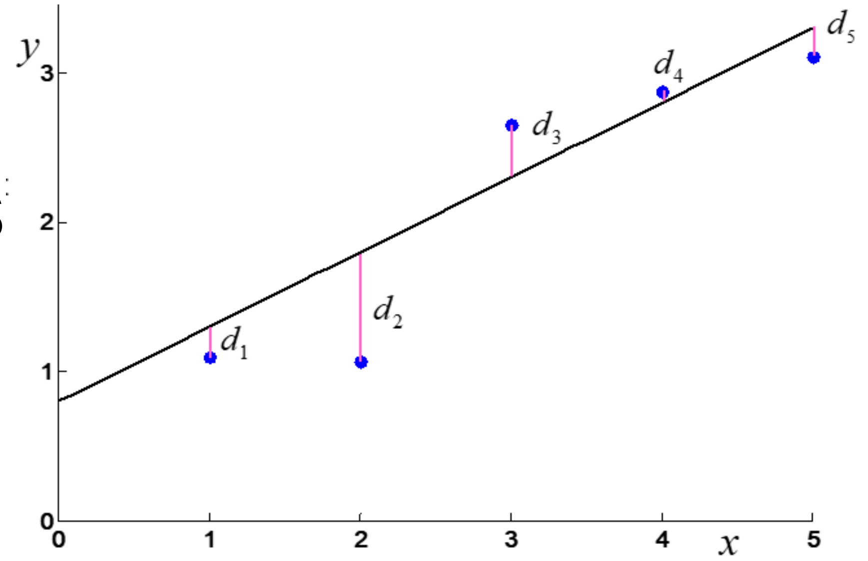
$$Q = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

2. Linear Regression and Neural Nets

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$
$$i = 1, \ldots, n$$

We can estimate the "best" linear relationship between an independent variable x and a dependent variable y using the score function

$$Q = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$



by taking derivatives with respect to the β coefficients

$$\frac{\partial Q}{\partial \beta_j} = -\frac{2}{n}\sum_{i} x_{ij}(y_i - \sum_{j} \beta_j x_{ij}) \qquad x_{i0} = 1 \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, q \end{matrix}$$

written in vector form as

$$\nabla Q = -\frac{2}{n}(X'y - X'X\beta) \qquad y = (y_1, \ldots, y_n)'$$

setting equal to zero and solving to get

$$\hat{\beta} = (X'X)^{-1}X'y$$

$$\beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_q \end{pmatrix} \qquad \nabla Q = \left(\frac{\partial Q}{\partial \beta_j}\right)$$

2. Linear Regression and Neural Nets

$$S_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_q x_{qi}$$

A *Neural Net* is a way to perform *Multiple Linear Regression* by using a linear activation function with the corresponding normal likelihood and least squares score function.

Linear Activation

$$f(S_i) = \sum_j \beta_j x_{ji}$$

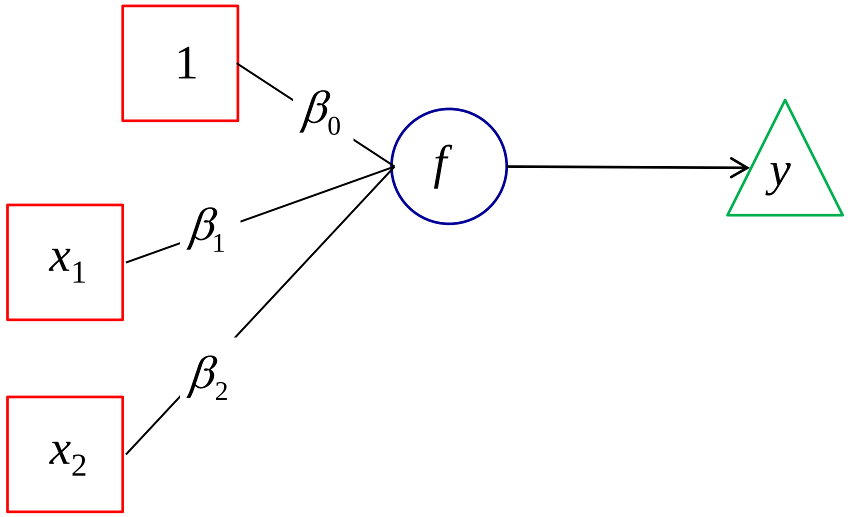
Normal Likelihood Score

$$Q = \frac{1}{n}\sum_i [y_i - \sum_j \beta_j x_{ji}]^2$$

Derivatives

$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n}\sum_i [y_i - \sum_j \beta_j x_{ij}](-x_{ij})$$

$$j = 0,1,2$$



$$\nabla Q = \begin{bmatrix} \dfrac{\partial Q}{\partial \beta_0} \\ \dfrac{\partial Q}{\partial \beta_1} \\ \dfrac{\partial Q}{\partial \beta_2} \end{bmatrix}$$

can set to 0 and get

$$\hat{\beta} = (X'X)^{-1}X'y$$

$$y = (y_1, \ldots, y_n)'$$

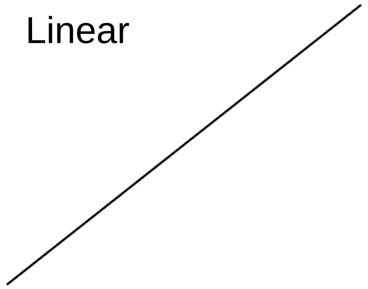
Gradient

$$\nabla Q = -\frac{2}{n}(X'y - X'X\beta)$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

Linear



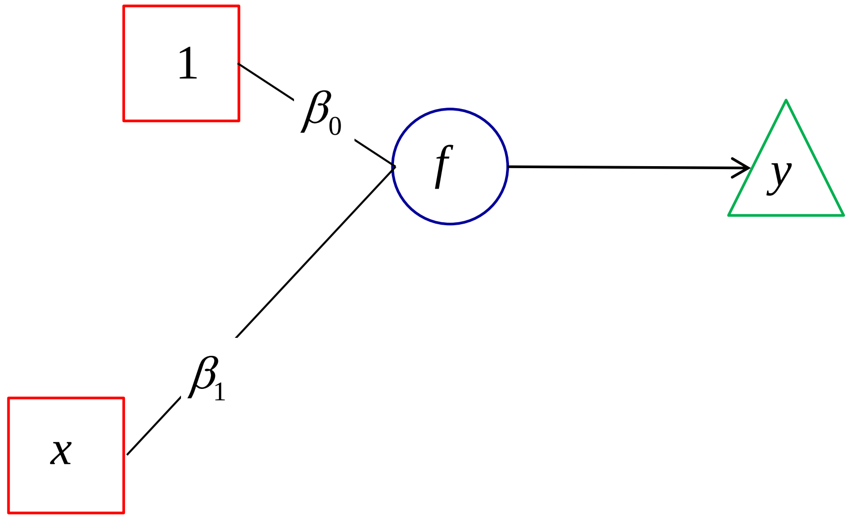
2. Linear Regression and Neural Nets

Example: Given observed data:

Data (x,y)
(1,1.4)
(2,2.3)
(3,1.7)
(4,3.0)
(5,3.4)

Linear Activation

$$f(x) = \beta_0 + \beta_1 x$$



use the *Neural Net* structure

and *Gradient Descent* to iteratively estimate the parameters.

Normal Likelihood Score

$$Q = \frac{1}{n}\sum_i [y_i - \sum_j \beta_j x_{ji}]^2$$

Gradient

$$\nabla Q = -\frac{2}{n}(X'y - X'X\beta)$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

$$\gamma = .0001$$

2. Linear Regression and Neural Nets

Example: Given observed data:

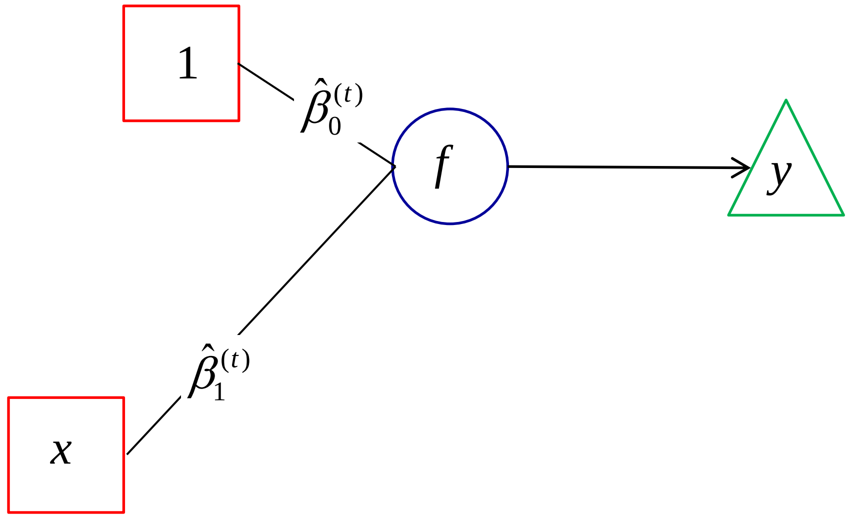
Data (x,y)
(1,1.4)
(2,2.3)
(3,1.7)
(4,3.0)
(5,3.4)

True Values

$(\beta_0, \beta_1) = (0.80, 0.50)$

Linear Activation

$$f(x) = \beta_0 + \beta_1 x$$



t=0

$$(\hat{\beta}_0^{(0)}, \hat{\beta}_1^{(0)}) = (1.50, 1.00)$$

Run data through with $\hat{\beta}^{(t)} = (\hat{\beta}_0^{(t)}, \hat{\beta}_1^{(t)})'$

Calculate $\nabla Q(\hat{\beta}^{(t)}) = -\frac{2}{n}(X'y - X'X\hat{\beta}^{(t)})$, $\gamma = .0001$

Calculate new $\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$, t=t+1

2. Linear Regression and Neural Nets

Example: Given observed data

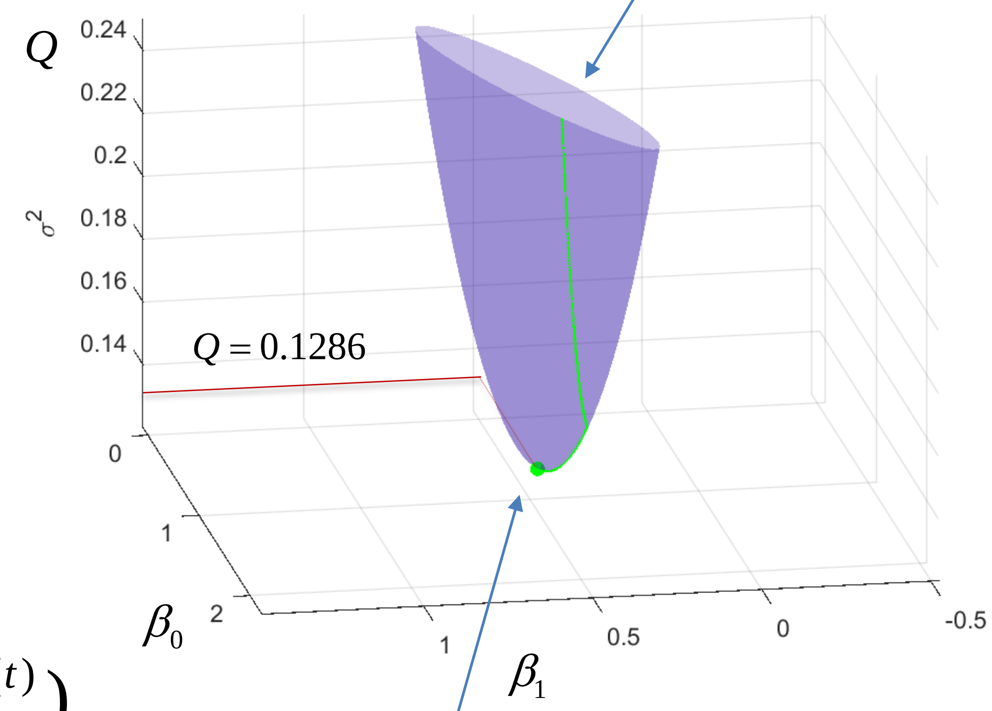
Data (x,y)
(1,1.4)
(2,2.3)
(3,1.7)
(4,3.0)
(5,3.4)

Linear Activation

$$f(x) = \beta_0 + \beta_1 x$$

$$\hat{\beta}^{(0)} = \begin{bmatrix} 1.50 \\ 1.00 \end{bmatrix}$$

Start



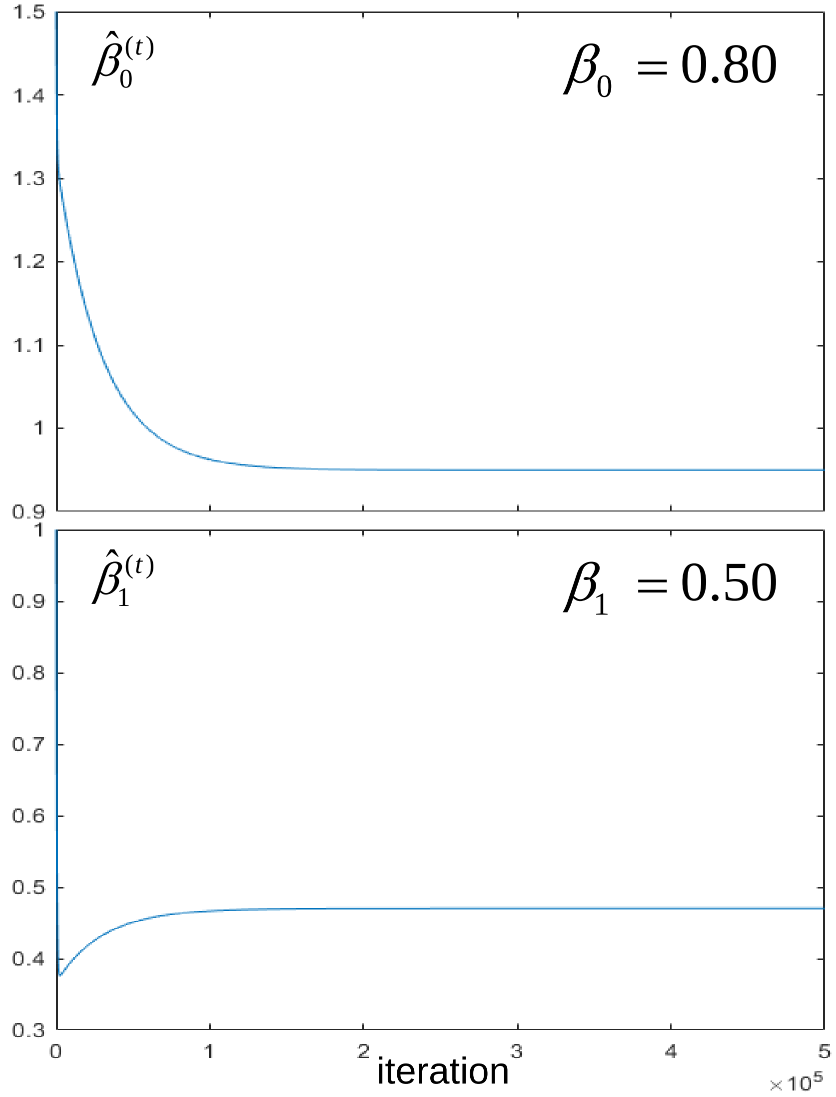
Finish

$$\hat{\beta} = \begin{bmatrix} 0.95 \\ 0.47 \end{bmatrix}$$

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

Normal Likelihood Score

$$Q = \frac{1}{n}\sum_i [y_i - \sum_j \beta_j x_{ji}]^2$$



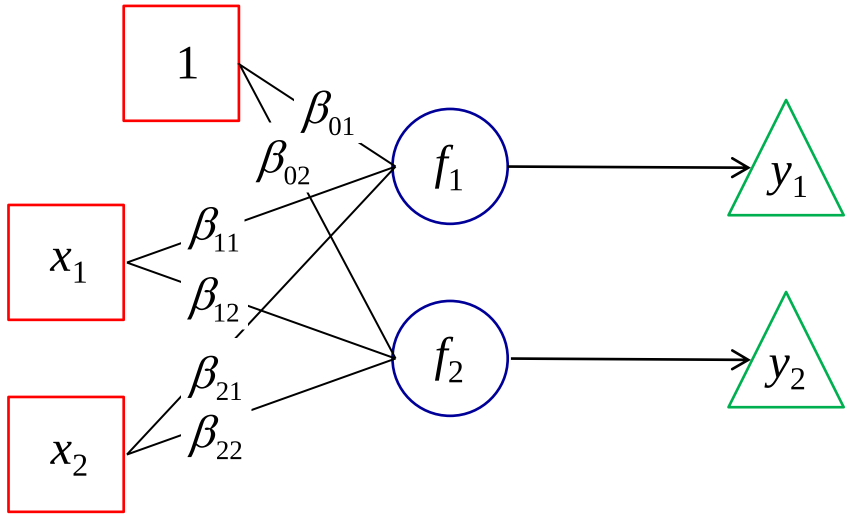
2. Linear Regression and Neural Nets

$$S_{ik} = \beta_{0k} + \beta_{1k}x_{1i} + \cdots + \beta_{qk}x_{qi}$$

A Neural Net is a way to do *Multivariate Linear Regression* with linear activation function and normal likelihood – least squares score function.

Linear Activation

$$f_k(S_{ik}) = \sum_j \beta_{jk} x_{ji}$$



$$\nabla Q_k = \begin{bmatrix} \frac{\partial Q_k}{\partial \beta_{0k}} \\ \frac{\partial Q_k}{\partial \beta_{1k}} \\ \frac{\partial Q_k}{\partial \beta_{2k}} \end{bmatrix}$$

Normal Likelihood Score

$$Q_k = \frac{1}{n}\sum_i [y_i - \sum_j \beta_{jk} x_{ji}]^2$$

Derivatives

$$\frac{\partial Q_k}{\partial \beta_{jk}} = \frac{2}{n}\sum_i [y_i - \sum_j \beta_{jk} x_{ij}](-x_{ij})$$

$j = 0,1,2 \quad k = 1,2$

Gradient

$$\nabla Q_k = -\frac{2}{n}(X'y - X'X\beta_k)$$

can set to 0 and get

$\hat{\beta}_k = (X'X)^{-1}X'y_k$

$y_k = (y_{1k},\ldots,y_{nk})'$

Gradient Descent

$$\hat{\beta}_k^{(t+1)} = \hat{\beta}_k^{(t)} - \gamma \nabla Q_k(\hat{\beta}_k^{(t)})$$

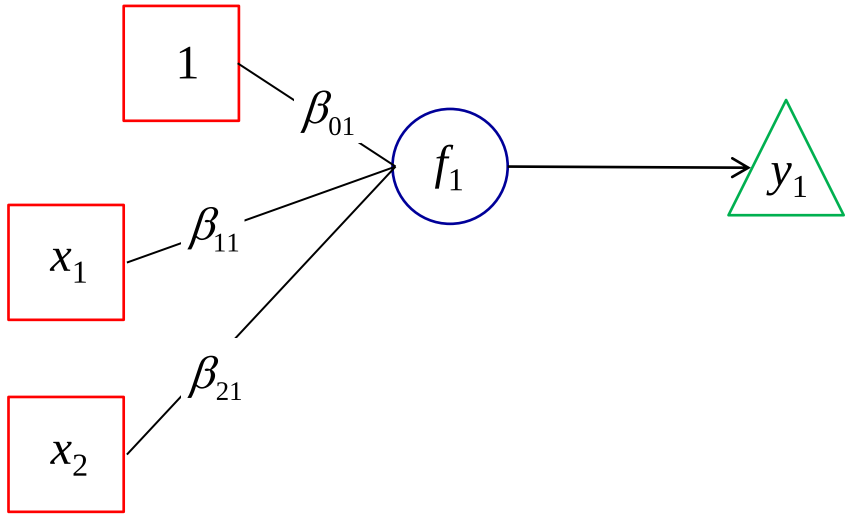
2. Linear Regression and Neural Nets

$$S_{i1} = \beta_{01} + \beta_{11} x_{1i} + \cdots + \beta_{q1} x_{qi}$$

A Neural Net is a way to do *Multivariate Linear Regression* with linear activation function and normal likelihood – least squares score function.

Linear Activation

$$f_1(S_{i1}) = \sum_j \beta_{j1} x_{ji}$$



$$\nabla Q_1 = \begin{bmatrix} \frac{\partial Q_1}{\partial \beta_{01}} \\ \frac{\partial Q_1}{\partial \beta_{11}} \\ \frac{\partial Q_1}{\partial \beta_{21}} \end{bmatrix}$$

Normal Likelihood Score

$$Q_1 = \frac{1}{n} \sum_i [y_i - \sum_j \beta_{j1} x_{ji}]^2$$

Derivatives

$$\frac{\partial Q_1}{\partial \beta_{j1}} = \frac{2}{n} \sum_i [y_i - \sum_j \beta_{j1} x_{ij}](-x_{ij})$$

$j = 0,1,2$

Gradient

$$\nabla Q_1 = -\frac{2}{n}(X'y - X'X\beta_1)$$

can set to 0 and get

$\hat{\beta}_1 = (X'X)^{-1}X'y_1$

$y_1 = (y_{11}, \ldots, y_{n1})'$

Gradient Descent

$$\hat{\beta}_1^{(t+1)} = \hat{\beta}_1^{(t)} - \gamma \nabla Q_1(\hat{\beta}_1^{(t)})$$

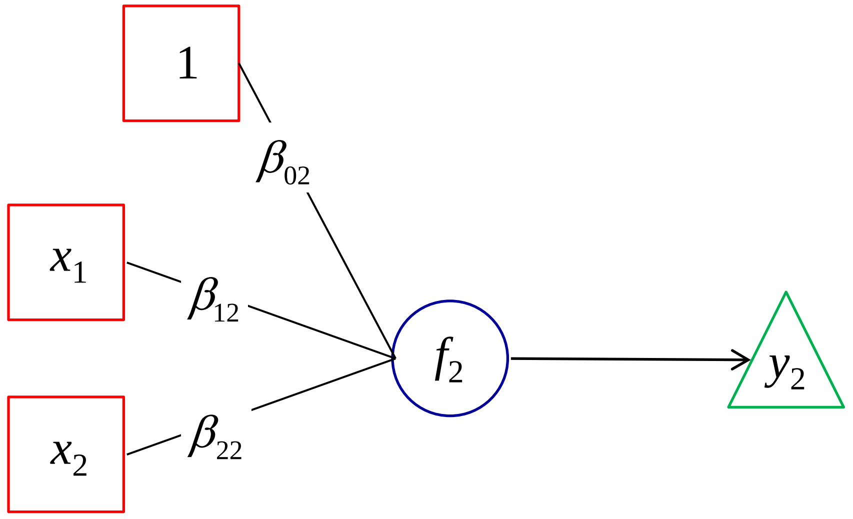
2. Linear Regression and Neural Nets

$$S_{i2} = \beta_{02} + \beta_{12}x_{1i} + \cdots + \beta_{q2}x_{qi}$$

A Neural Net is a way to do *Multivariate Linear Regression* with linear activation function and normal likelihood – least squares score function.

Linear Activation

$$f_2(S_{i2}) = \sum_j \beta_{j2}x_{ji}$$



$$\nabla Q_2 = \begin{bmatrix} \frac{\partial Q_2}{\partial \beta_{02}} \\ \frac{\partial Q_2}{\partial \beta_{12}} \\ \frac{\partial Q_2}{\partial \beta_{22}} \end{bmatrix}$$

Normal Likelihood Score

$$Q_2 = \frac{1}{n}\sum_i [y_i - \sum_j \beta_{j2}x_{ji}]^2$$

Derivatives

$$\frac{\partial Q_2}{\partial \beta_{j2}} = \frac{2}{n}\sum_i [y_i - \sum_j \beta_{j2}x_{ij}](-x_{ij})$$

$j = 0,1,2$

Gradient

$$\nabla Q_2 = -\frac{2}{n}(X'y - X'X\beta_2)$$

can set to 0 and get

$\hat{\beta}_2 = (X'X)^{-1}X'y_2$

$y_2 = (y_{12},\ldots,y_{n2})'$

Gradient Descent

$$\hat{\beta}_2^{(t+1)} = \hat{\beta}_2^{(t)} - \gamma\nabla Q_2(\hat{\beta}_2^{(t)})$$

2. Linear Regression and Neural Nets

$$S_{ik} = \beta_{0k} + \beta_{1k} x_{1i} + \cdots + \beta_{qk} x_{qi}$$

Two independent parallel *Multiple Regressions* yield the same coefficient estimate as simultaneous multivariate regression (different covariance).

$$\hat{B} = (X'X)^{-1} X'Y$$

$$\hat{B} = (\hat{\beta}_1, \hat{\beta}_2)$$

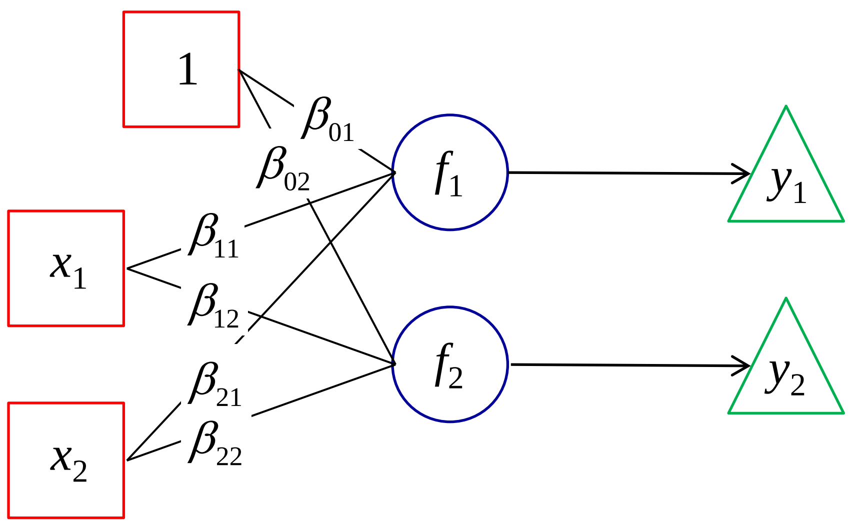
$$Y = (y_1, y_2)$$

$$\hat{\beta}_1 = (X'X)^{-1} X' y_1$$

$$\hat{\beta}_2 = (X'X)^{-1} X' y_2$$

$$y_1 = (y_{11}, \ldots, y_{n1})'$$

$$y_2 = (y_{12}, \ldots, y_{n2})'$$



$$\hat{\beta}_1^{(t+1)} = \hat{\beta}_1^{(t)} - \gamma \nabla Q_1(\hat{\beta}_1^{(t)})$$

$$\hat{\beta}_2^{(t+1)} = \hat{\beta}_2^{(t)} - \gamma \nabla Q_2(\hat{\beta}_2^{(t)})$$

Now the trained *Neural Net* can be applied to new data.

3. Non-Linear Regression and Neural Nets

We might believe that there is a non-linear relationship between an independent variable x, and a dependent variable y with measurement error.

$$y_i = f(x_i) + \varepsilon_i \qquad\qquad i = 1,\ldots,n$$

We can assume normal errors or use least squares.

$$Q = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

As an example consider the SoftPlus function $f(\cdot) = \ln(1 + e^{\sum_j \beta_j x_j})$.

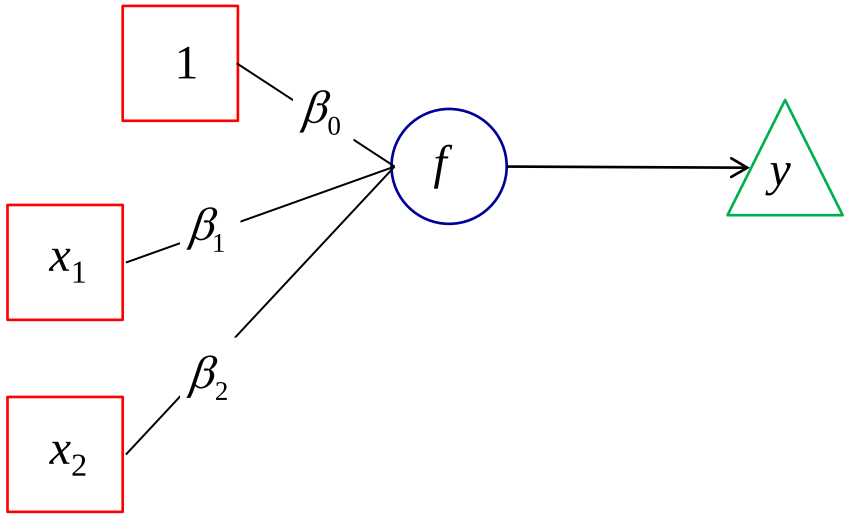
3. Non-Linear Regression and Neural Nets

$$S_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_q x_{qi}$$

A Neural Net is a way to do *Non-Linear Regression* with SoftPlus activation function and normal likelihood – least squares score function.

SoftPlus Activation

$$f(S_i) = \ln(1 + e^{S_i})$$



$$\nabla Q = \begin{bmatrix} \dfrac{\partial Q}{\partial \beta_0} \\ \dfrac{\partial Q}{\partial \beta_1} \\ \dfrac{\partial Q}{\partial \beta_2} \end{bmatrix}$$

Normal Likelihood Score

$$Q = \frac{1}{n} \sum_i [y_i - \ln(1 + e^{S_i})]^2$$

Derivatives

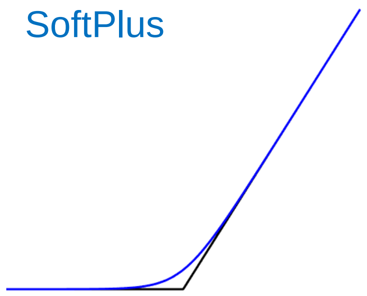
$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n} \sum_i \left[\frac{x_{ij}[y_i - \ln(1 + \exp(\sum_j \beta_j x_{ij}))]\exp(\sum_j \beta_j x_{ij})}{1 + \exp(\sum_j \beta_j x_{ij})} \right] \quad j = 0,1,2$$

Gradient

$$\nabla Q = \left(\frac{\partial Q}{\partial \beta_j} \right)$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$



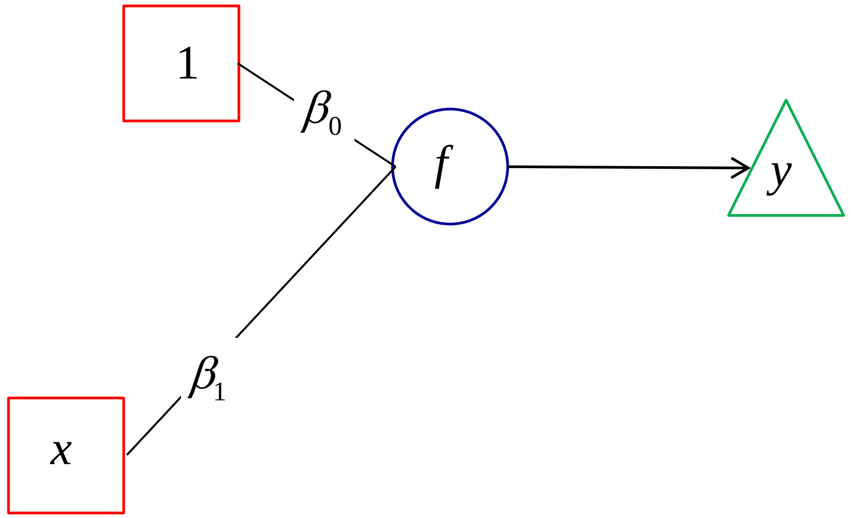
3. Non-Linear Regression and Neural Nets

$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n}\sum_i \left[\frac{x_{ij}[y_i - \ln(1+\exp(\sum_j \beta_j x_{ij}))]\exp(\sum_j \beta_j x_{ij})}{1+\exp(\sum_j \beta_j x_{ij})} \right]$$

Example: Given observed data:

SoftPlus Activation

$$f(S_i) = \ln(1+e^{\beta_0+\beta_1 x_{1i}})$$



use the *Neural Net* structure

and *Gradient Descent* to iteratively estimate the parameters.

Normal Likelihood Score

$$Q = \frac{1}{n}\sum_i [y_i - \ln(1+e^{\beta_0+\beta_1 x_{1i}})]^2$$

Gradient

$$\nabla Q = \left(\frac{\partial Q}{\partial \beta_j} \right)$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

$$\gamma = .0001$$

3. Non-Linear Regression and Neural Nets

$$\frac{\partial Q}{\partial \beta_j} = \frac{2}{n}\sum_i \left[\frac{x_{ij}[y_i - \ln(1+\exp(\sum_j \beta_j x_{ij}))]\exp(\sum_j \beta_j x_{ij})}{1+\exp(\sum_j \beta_j x_{ij})} \right]$$

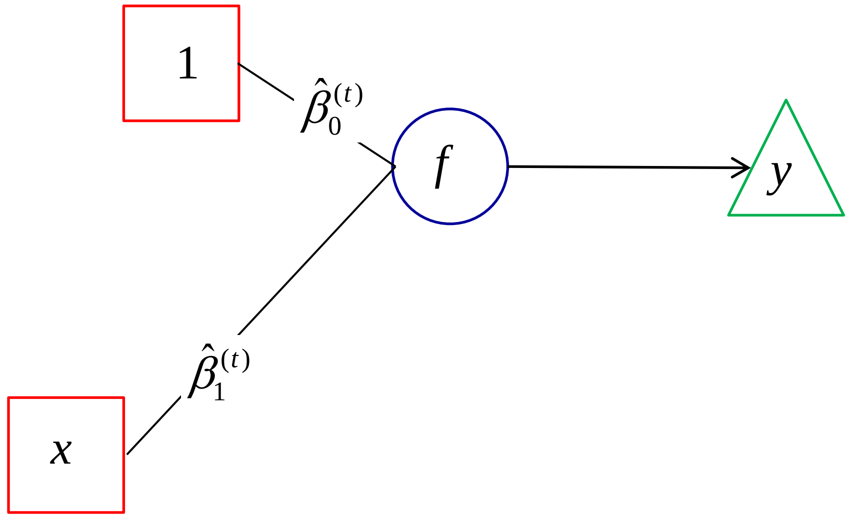
Example: Given observed data:

True Values

(β_0, β_1)

SoftPlus Activation

$$f(S_{ik}) = \ln(1+e^{S_{ik}})$$



$t=0$

$(\hat{\beta}_0^{(0)}, \hat{\beta}_1^{(0)})$

Run data through with $\hat{\beta}^{(t)} = (\hat{\beta}_0^{(t)}, \hat{\beta}_1^{(t)})'$

Calculate $\nabla Q(\hat{\beta}^{(t)})$, $\gamma = .0001$

Calculate new $\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$, $t=t+1$

3. Non-Linear Regression and Neural Nets

$$S_{ik} = \beta_{0k} + \beta_{1k} x_{1i} + \cdots + \beta_{qk} x_{qi}$$

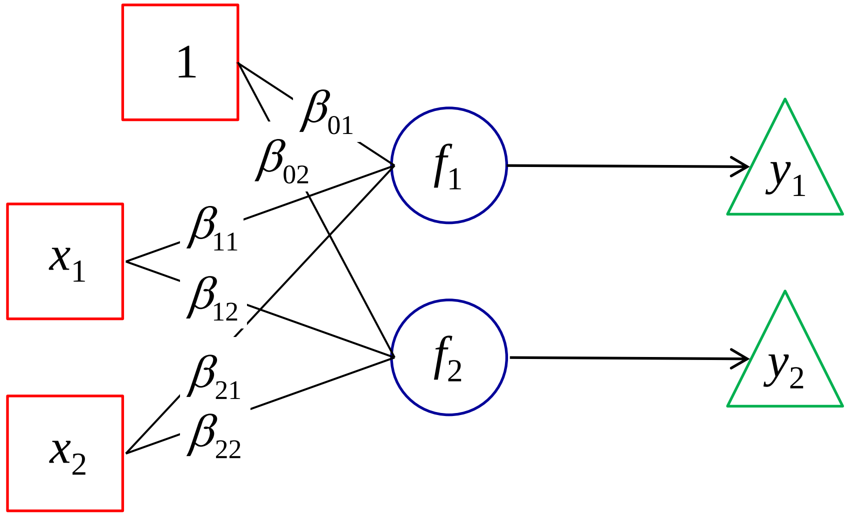
A *Neural Net* is a way to do *Multivariate Non-Linear Regression* with SoftPlus activation function and normal likelihood – least squares score function.

SoftPlus Activation

$$f(S_{ik}) = \ln(1 + e^{S_{ik}})$$

Normal Likelihood Score

$$Q_k = \frac{1}{n} \sum_i [y_i - \ln(1 + e^{S_{ik}})]^2$$



$$\nabla Q_k = \begin{bmatrix} \frac{\partial Q_k}{\partial \beta_{0k}} \\ \frac{\partial Q_k}{\partial \beta_{1k}} \\ \frac{\partial Q_k}{\partial \beta_{2k}} \end{bmatrix}$$

Derivatives

$$\frac{\partial Q_k}{\partial \beta_{jk}} = \frac{2}{n} \sum_i \left[\frac{x_{ij}[y_{ik} - \ln(1 + \exp(\sum_j \beta_{jk} x_{ij}))]\exp(\sum_j \beta_{jk} x_{ij})}{1 + \exp(\sum_j \beta_{jk} x_{ij})} \right]$$

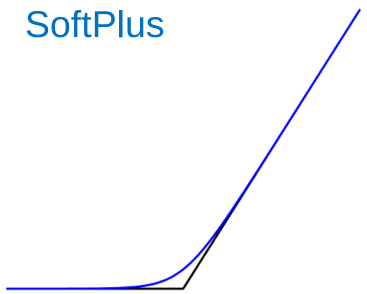
$j = 0,1,2 \qquad k = 1,2$

Gradient

$$\nabla Q_k = \left(\frac{\partial Q_k}{\partial \beta_{jk}} \right)$$

Gradient Descent

$$\hat{\beta}_k^{(t+1)} = \hat{\beta}_k^{(t)} - \gamma \nabla Q_k(\hat{\beta}_k^{(t)})$$



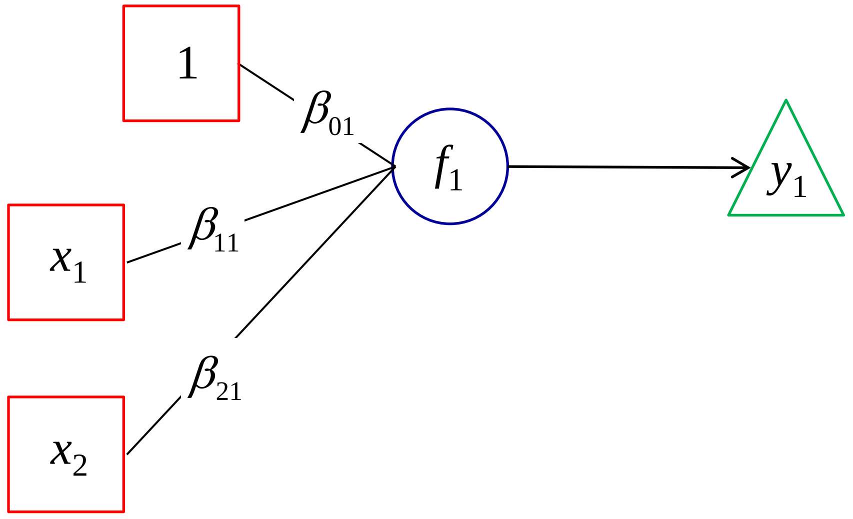
3. Non-Linear Regression and Neural Nets

$$S_{ik} = \beta_{0k} + \beta_{1k} x_{1i} + \cdots + \beta_{qk} x_{qi}$$

A *Neural Net* is a way to do *Multivariate Non-Linear Regression* with SoftPlus activation function and normal likelihood – least squares score function.

SoftPlus Activation

$$f(S_{i1}) = \ln(1 + e^{S_{i1}})$$



$$\nabla Q_1 = \begin{bmatrix} \dfrac{\partial Q_1}{\partial \beta_{01}} \\ \dfrac{\partial Q_1}{\partial \beta_{11}} \\ \dfrac{\partial Q_1}{\partial \beta_{21}} \end{bmatrix}$$

Normal Likelihood Score

$$Q_1 = \frac{1}{n} \sum_i [y_i - \ln(1 + e^{S_{i1}})]^2$$

Derivatives

$$\frac{\partial Q_1}{\partial \beta_{j1}} = \frac{2}{n} \sum_i \left[\frac{x_{ij}[y_{i1} - \ln(1 + \exp(\sum_j \beta_{j1} x_{ij}))]\exp(\sum_j \beta_{j1} x_{ij})}{1 + \exp(\sum_j \beta_{j1} x_{ij})} \right]$$

$j = 0,1,2$

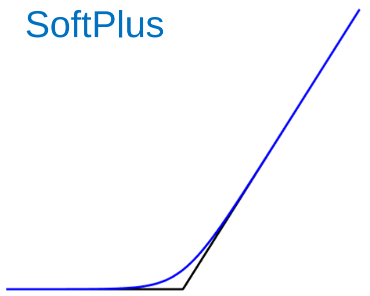
Gradient

$$\nabla Q_1 = \left(\frac{\partial Q_1}{\partial \beta_{j1}} \right)$$

Gradient Descent

$$\hat{\beta}_1^{(t+1)} = \hat{\beta}_1^{(t)} - \gamma \nabla Q_1(\hat{\beta}_1^{(t)})$$

SoftPlus



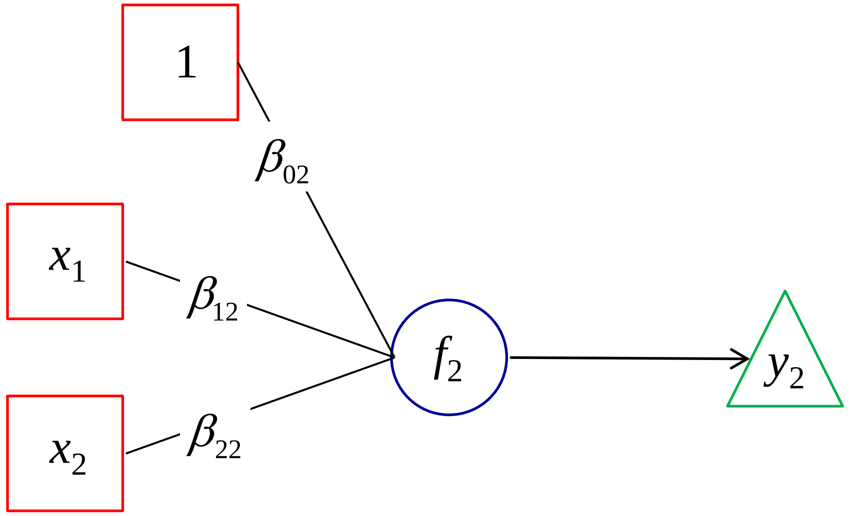
3. Non-Linear Regression and Neural Nets

$$S_{i2} = \beta_{02} + \beta_{12} x_{1i} + \cdots + \beta_{q2} x_{qi}$$

A *Neural Net* is a way to do *Multivariate Non-Linear Regression* with SoftPlus activation function and normal likelihood – least squares score function.

SoftPlus Activation

$$f(S_{i2}) = \ln(1 + e^{S_{i2}})$$



$$\nabla Q_2 = \begin{bmatrix} \frac{\partial Q_2}{\partial \beta_{02}} \\ \frac{\partial Q_2}{\partial \beta_{12}} \\ \frac{\partial Q_2}{\partial \beta_{22}} \end{bmatrix}$$

Normal Likelihood Score

$$Q_2 = \frac{1}{n} \sum_i [y_i - \ln(1 + e^{S_{i2}})]^2$$

Derivatives

$$\frac{\partial Q_2}{\partial \beta_{j2}} = \frac{2}{n} \sum_i \left[\frac{x_{ij}[y_{i2} - \ln(1 + \exp(\sum_j \beta_{j2} x_{ij}))] \exp(\sum_j \beta_{j2} x_{ij})}{1 + \exp(\sum_j \beta_{j2} x_{ij})} \right]$$

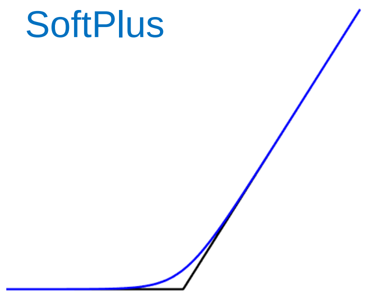
$j = 0,1,2$

Gradient

$$\nabla Q_2 = \left(\frac{\partial Q_2}{\partial \beta_{j2}} \right)$$

Gradient Descent

$$\hat{\beta}_2^{(t+1)} = \hat{\beta}_2^{(t)} - \gamma \nabla Q_2(\hat{\beta}_2^{(t)})$$



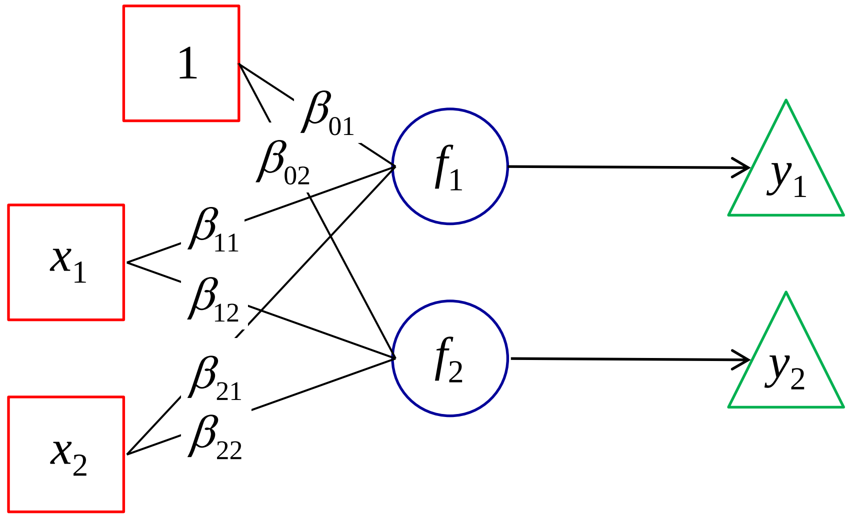
3. Non-Linear Regression and Neural Nets

$$S_{i2} = \beta_{02} + \beta_{12} x_{1i} + \cdots + \beta_{q2} x_{qi}$$

Two independent Non-Linear Regressions yield the same coefficient estimate as simultaneous Non-Linear Regression.

$$\hat{B} = (\hat{\beta}_1, \hat{\beta}_2)$$

$$Y = (y_1, y_2)$$



$\hat{\beta}_1$ The last one in iteration.

$\hat{\beta}_2$ The last one in iteration.

$y_1 = (y_{11}, \ldots, y_{n1})'$

$y_2 = (y_{12}, \ldots, y_{n2})'$

$$\hat{\beta}_1^{(t+1)} = \hat{\beta}_1^{(t)} - \gamma \nabla Q_1(\hat{\beta}_1^{(t)})$$

$$\hat{\beta}_2^{(t+1)} = \hat{\beta}_2^{(t)} - \gamma \nabla Q_2(\hat{\beta}_2^{(t)})$$

Now the trained *Neural Net* can be applied to new data.

4. Logistic Regression and Neural Nets

Often the probability p of an event E depends upon an independent variable x, such as the probability of getting an A on a class final depends on the number of hours that a student studies x.

So p is a function of x, $p(x)$. $0 \leq p(x) \leq 1$.

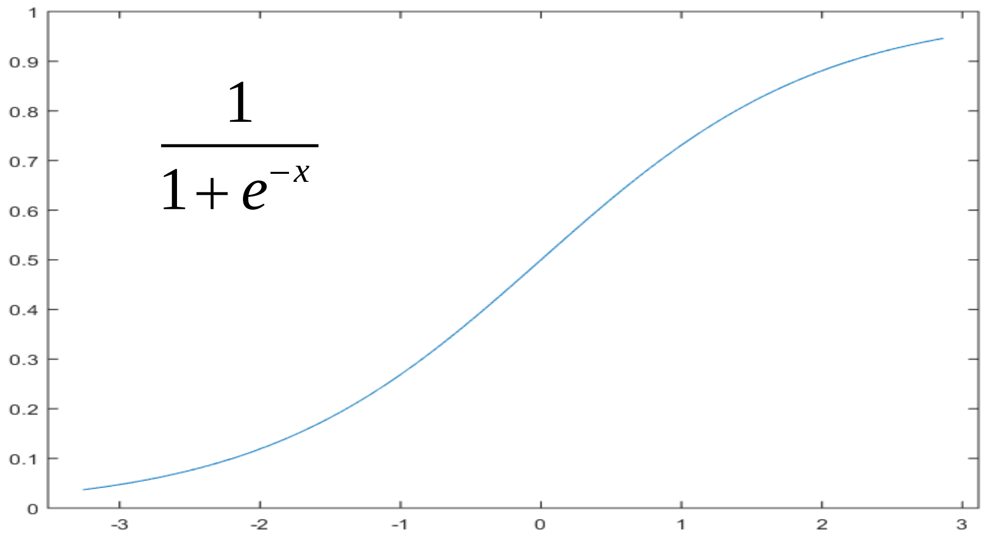
Hours (x)	A (y)
6	0
8	0
10	0
12	0
14	0
16	1
18	0
20	0
22	0
24	0
26	1
28	0
30	0
32	1
34	1
36	1
38	1
40	1

4. Logistic Regression and Neural Nets

This dependency of a probability $p(x)$, $0 \leq p(x) \leq 1$, on an independent variable x, $-\infty < x < \infty$, is generally described through a *link function*, here the logistic mapping function

$$p = p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \quad .$$

If the event E occurs, then we say $y=1$ and if not, $y=0$.
$P(y=1)=p$ and $P(y=0)=1-p$. … This is a Bernoulli trial.



4. Logistic Regression and Neural Nets

The likelihood function

$$L(\beta_0,\beta_1)=\prod_{i=1}^{n}[p(x_i)]^{y_i}[1-p(x_i)]^{1-y_i}$$

$y_i=\{0,1\}$

$-\infty < x_i < \infty$

where $p(x_i)=\dfrac{1}{1+e^{-\beta_0-\beta_1 x_i}}$

Caution!

Do not use least squares!

$$Q=\frac{1}{n}\sum_i\left[y_i-\frac{1}{1+e^{-\beta_0-\beta_1 x_i}}\right]^2$$

has log likelihood function

$$\begin{aligned}LL(\beta_0,\beta_1) &= \sum_{i=1}^{n}\left[y_i\ln[p(x_i)]+(1-y_i)\ln[1-p(x_i)]\right]\\ &= \sum_{i=1}^{n}y_i\ln[p(x_i)]+\sum_{i=1}^{n}(1-y_i)\ln[1-p(x_i)]\\ &= \sum_{i=1}^{n}\ln[1-p(x_i)]+\sum_{i=1}^{n}y_i\ln[p(x_i)/(1-p(x_i))]\\ &= \sum_{i=1}^{n}y_i[\beta_0+\beta_1 x_i]-\sum_{i=1}^{n}\ln[1+e^{\beta_0+\beta_1 x_i}]\,.\end{aligned}$$

4. Logistic Regression and Neural Nets

We can estimate the “best” logistic relationship between x and $0/1$ y using the score function

$$Q=\sum_i y_i(\sum_j \beta_j x_{ij})-\sum_i \ln[1+\exp(\sum_j \beta_j x_{ij})]$$

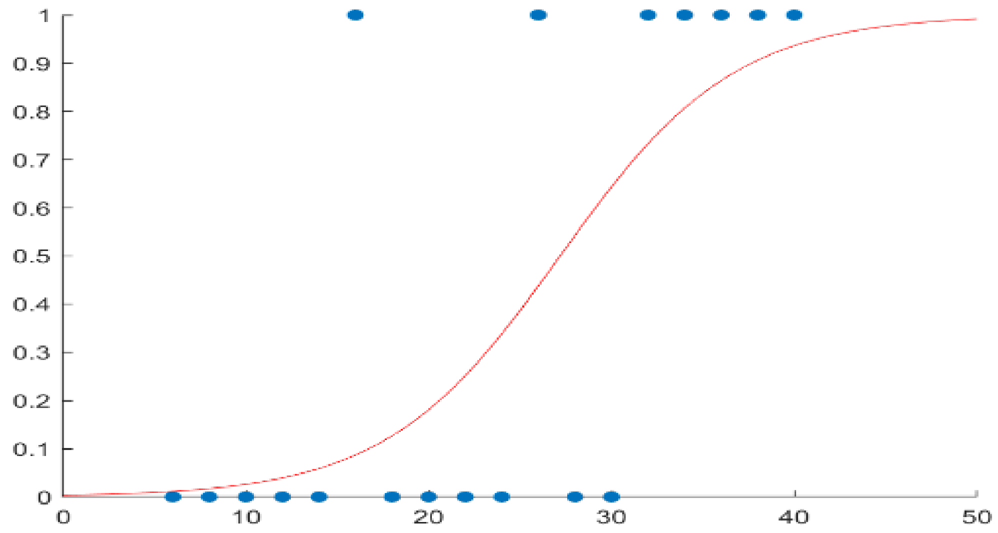
by taking derivatives with respect to the beta’s

$$\frac{\partial Q}{\partial \beta_j}=\sum_i x_{ij}y_i-\sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_j x_{ij})}$$

$$i=1,\ldots,n \quad j=1,\ldots,q$$
$$x_{i0}=1$$

with no closed form solution.

$$f(\cdot)=\frac{1}{1+\exp(-\sum_j \beta_j x_j)}$$

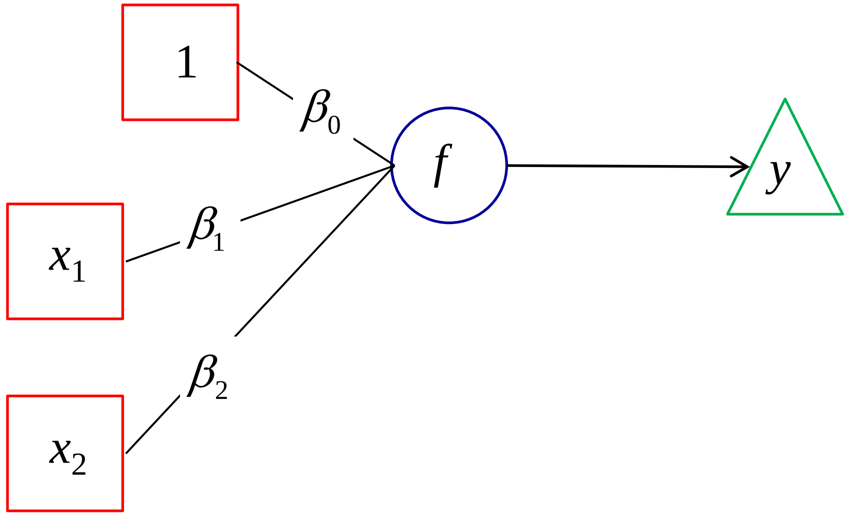


4. Logistic Regression and Neural Nets

A Neural Net is a way to represent *Multiple Logistic Regression* with Logistic activation and Bernoulli likelihood score function.

Logistic Activation

$$f(x) = \frac{1}{1+\exp(-\sum_j \beta_j x_j)}$$



$$\nabla Q = \begin{bmatrix} \frac{\partial Q}{\partial \beta_0} \\ \frac{\partial Q}{\partial \beta_1} \\ \frac{\partial Q}{\partial \beta_2} \end{bmatrix}$$

Bernoulli Likelihood Score

$$Q = \sum_i y_i (\sum_j \beta_j x_{ji}) - \sum_i \ln[1+\exp(\sum_j \beta_j x_{ji})]$$

Gradient

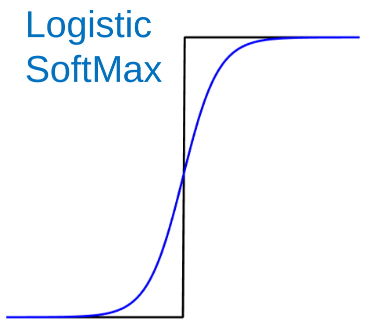
$$\nabla Q = \left(\frac{\partial Q}{\partial \beta_j} \right)$$

Derivatives

$$\frac{\partial Q}{\partial \beta_j} = \sum_i x_{ij} y_i - \sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_j x_{ij})} \qquad j = 0,1,2$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

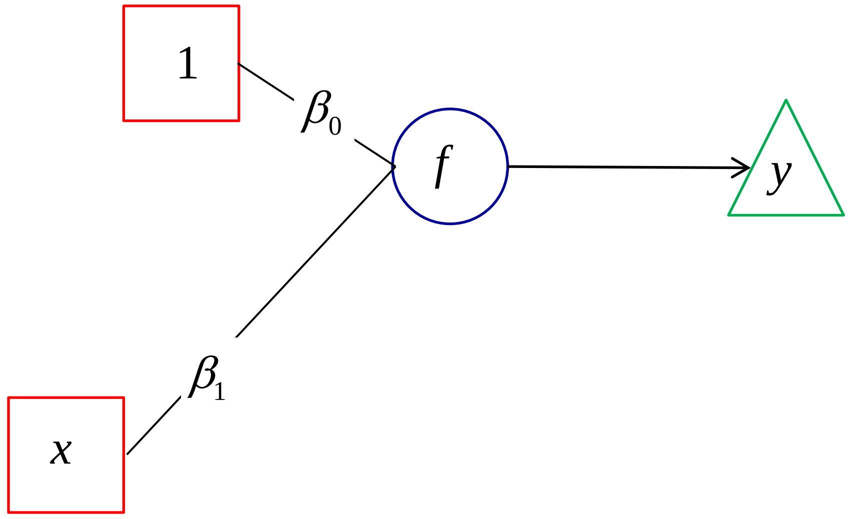


4. Logistic Regression and Neural Nets

Simple *Logistic Regression*
Given observed data:

Hours (x)	A (y)
6	0
8	0
10	0
12	0
14	0
16	1
18	0
20	0
22	0
24	0
26	1
28	0
30	0
32	1
34	1
36	1
38	1
40	1

$$f(x) = \frac{1}{1 + \exp(-\sum_j \beta_j x_j)}$$



use the *Neural Net* structure and *Gradient Descent* to iteratively estimate the parameters.

Gradient

$$\nabla Q = \left(\frac{\partial Q}{\partial \beta_j} \right)$$

Gradient Descent

$$\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

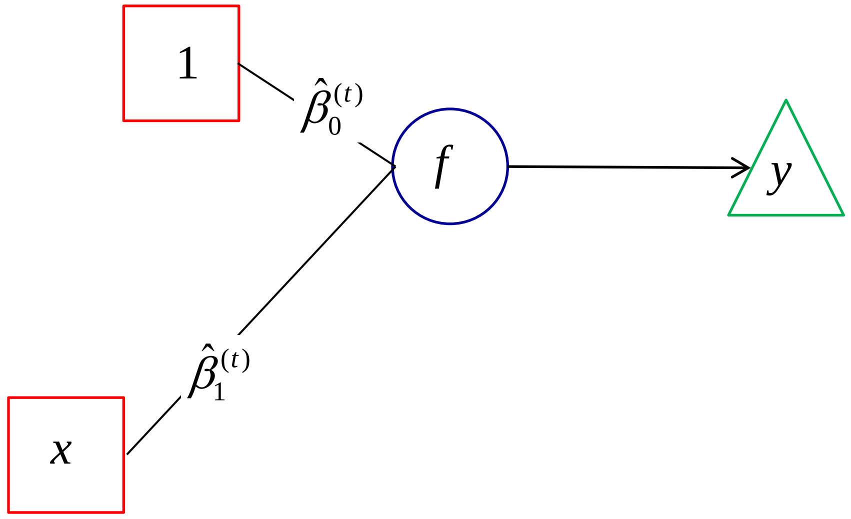
4. Logistic Regression and Neural Nets

Simple *Logistic Regression*
Given observed data:

Hours (x)	A (y)
6	0
8	0
10	0
12	0
14	0
16	1
18	0
20	0
22	0
24	0
26	1
28	0
30	0
32	1
34	1
36	1
38	1
40	1

True Values
$(\beta_0, \beta_1) = (-5.00, 0.20)$

$$f(x) = \frac{1}{1+\exp(-\sum_j \beta_j x_j)}$$



$t=0$

$$(\hat{\beta}_0^{(0)}, \hat{\beta}_1^{(0)}) = (3.00, 0.50)$$

Run data through with $\hat{\beta}^{(t)} = (\hat{\beta}_0^{(t)}, \hat{\beta}_1^{(t)})'$

Calculate $\nabla Q(\hat{\beta}^{(t)}) = \sum_i x_{ij} y_i - \sum_i \frac{x_{ij}}{1+\exp(-\sum \beta_j x_{ij})}$, $\gamma = .0001$

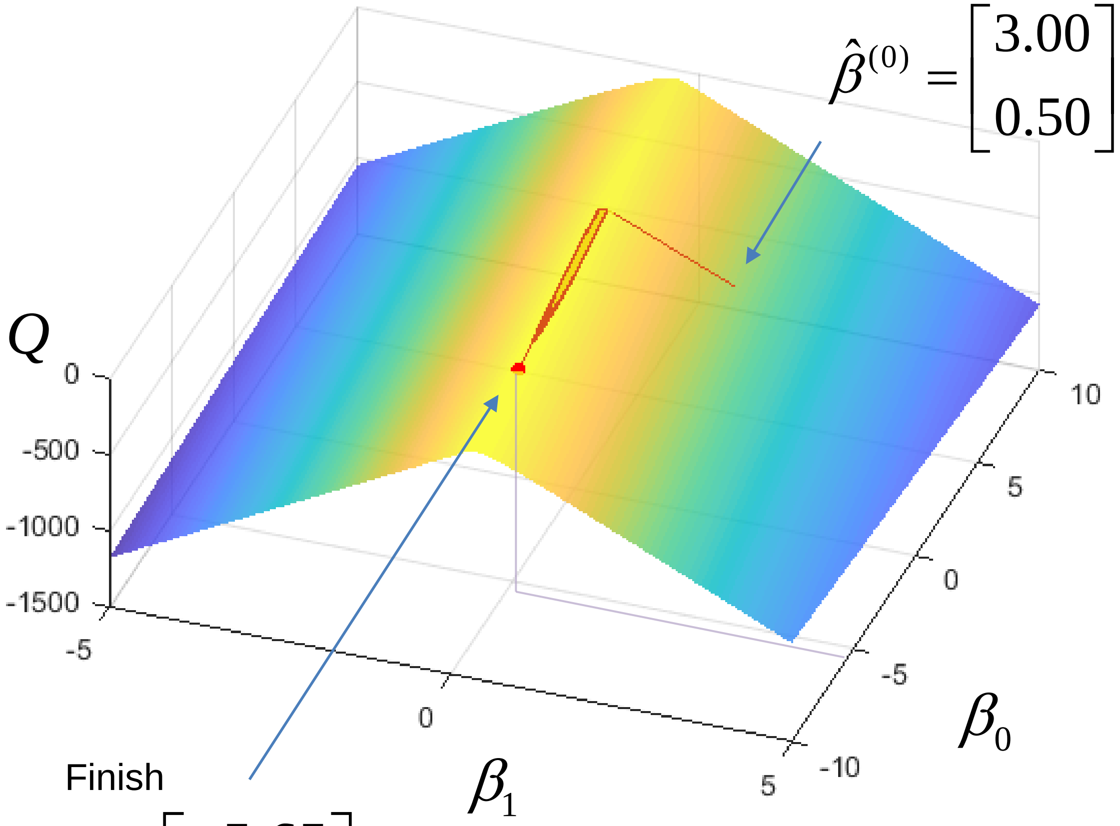
Calculate new $\hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$, $t=t+1$

4. Logistic Regression and Neural Nets

Simple *Logistic Regression* results:

Start

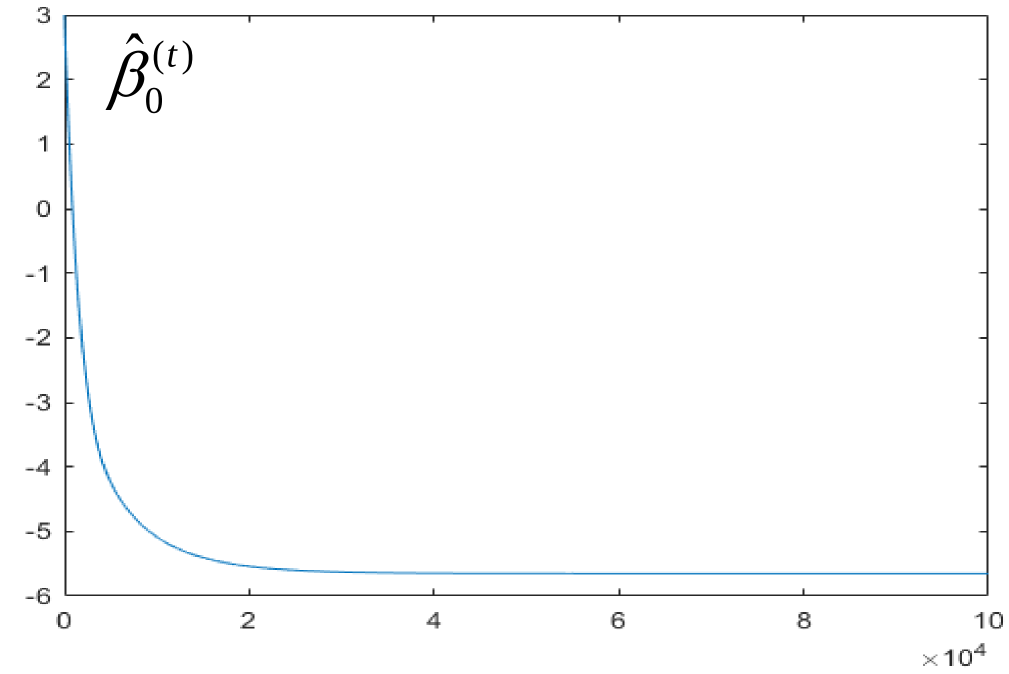
$$\hat{\beta}^{(0)} = \begin{bmatrix} 3.00 \\ 0.50 \end{bmatrix}$$



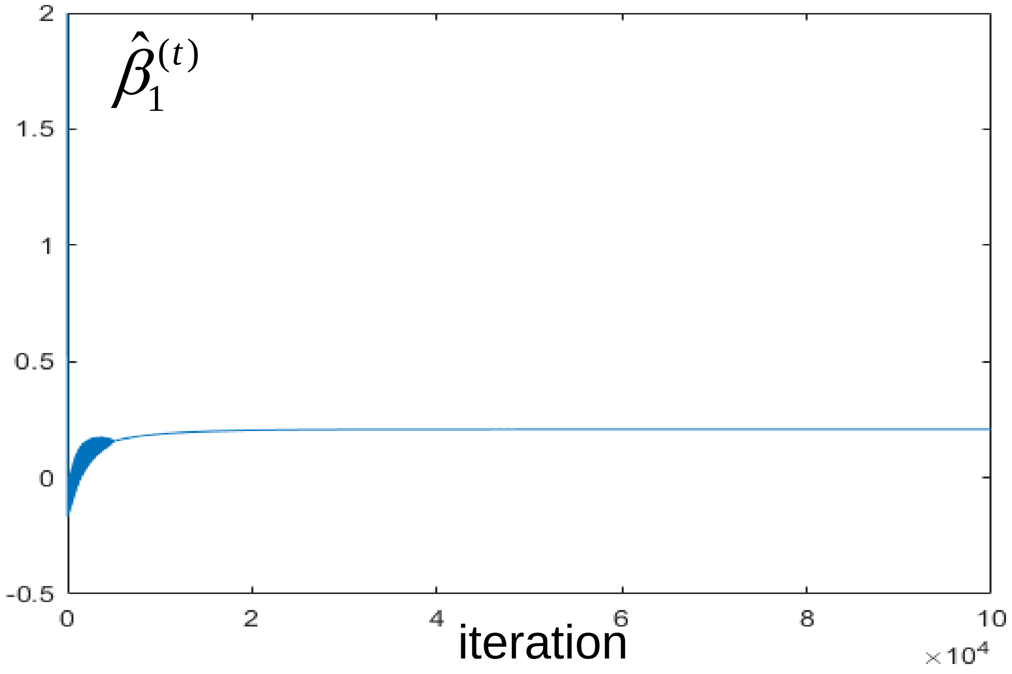
Finish

$$\hat{\beta} = \begin{bmatrix} -5.65 \\ 0.21 \end{bmatrix}$$

$\beta_0 = -5.00$



$\beta_1 = 0.20$

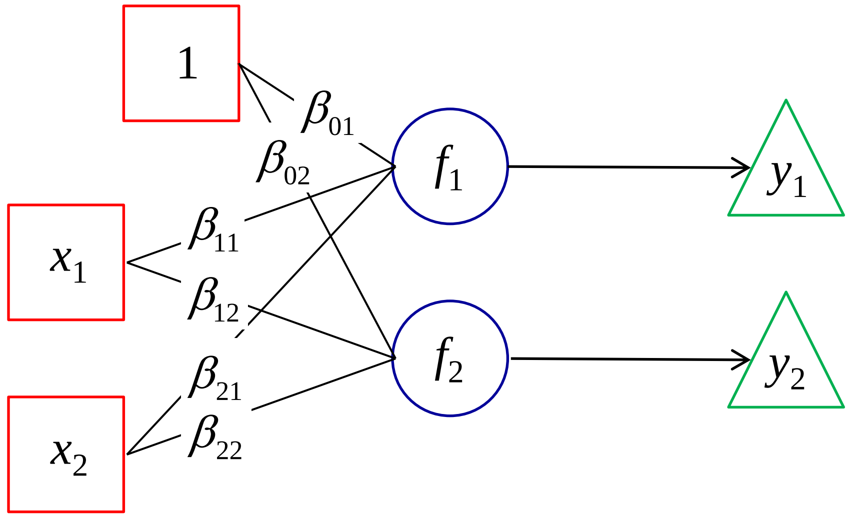


4. Logistic Regression and Neural Nets

A Neural Net is a way to represent *Multivariate Logistic Regression* with logistic activation and Bernoulli score function.

Logistic Activation

$$f_k(x) = \frac{1}{1+\exp(-\sum_j \beta_{jk} x_j)}$$



$$\nabla Q_k = \begin{bmatrix} \frac{\partial Q_k}{\partial \beta_{0k}} \\ \frac{\partial Q_k}{\partial \beta_{1k}} \\ \frac{\partial Q_k}{\partial \beta_{2k}} \end{bmatrix}$$

Bernoulli Likelihood Score

$$Q_k = \sum_i y_{ik} (\sum_j \beta_{jk} x_{ij}) - \sum_i \ln[1+\exp(\sum_j \beta_{jk} x_{ij})]$$

Derivatives

$$\frac{\partial Q_k}{\partial \beta_{jk}} = \sum_i x_{ij} y_{ik} - \sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_{jk} x_{ij})}$$

$$j = 0,1,2 \quad k = 1,2$$

Gradient

$$\nabla Q_k = \left(\frac{\partial Q_k}{\partial \beta_{jk}} \right)$$

Gradient Descent

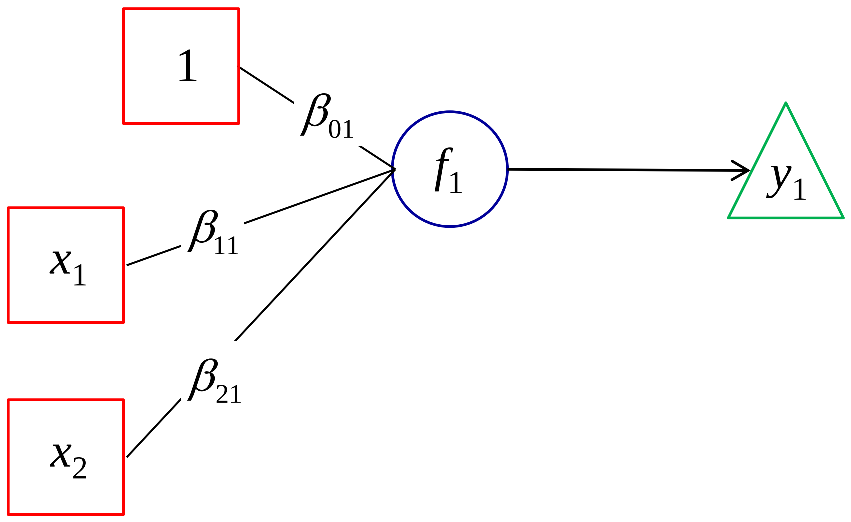
$$\hat{\beta}_k^{(t+1)} = \hat{\beta}_k^{(t)} - \gamma \nabla Q_k (\hat{\beta}_k^{(t)})$$

4. Logistic Regression and Neural Nets

A Neural Net is a way to represent *Multivariate Logistic Regression* with logistic activation and Bernoulli score function.

Logistic Activation

$$f_1(x) = \frac{1}{1+\exp(-\sum_j \beta_{j1} x_j)}$$



$$\nabla Q_1 = \begin{bmatrix} \frac{\partial Q_1}{\partial \beta_{01}} \\ \frac{\partial Q_1}{\partial \beta_{11}} \\ \frac{\partial Q_1}{\partial \beta_{21}} \end{bmatrix}$$

Bernoulli Likelihood Score

$$Q_1 = \sum_i y_{i1}(\sum_j \beta_{j1} x_{ij}) - \sum_i \ln[1+\exp(\sum_j \beta_{j1} x_{ij})]$$

Derivatives

$$\frac{\partial Q_1}{\partial \beta_{j1}} = \sum_i x_{ij} y_{i1} - \sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_{j1} x_{ij})}$$

$j=0,1,2$

Gradient

$$\nabla Q_k = \left(\frac{\partial Q_k}{\partial \beta_{jk}} \right)$$

Gradient Descent

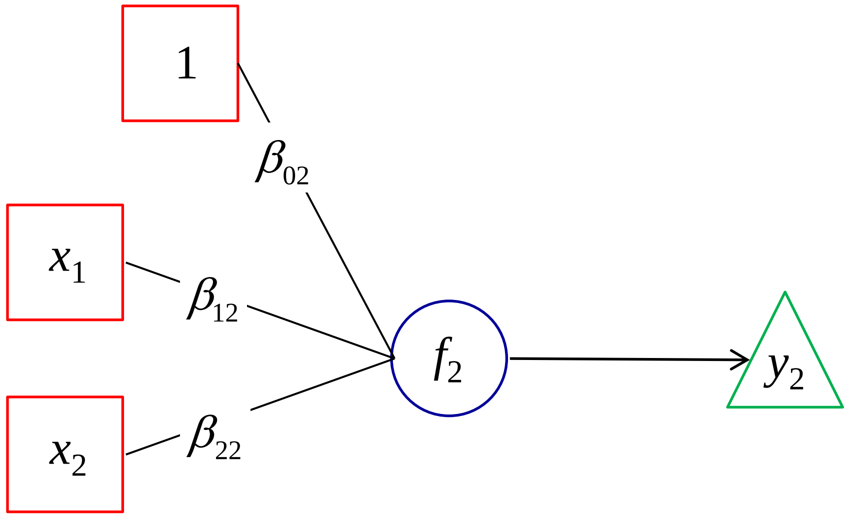
$$\hat{\beta}_k^{(t+1)} = \hat{\beta}_k^{(t)} - \gamma \nabla Q_k(\hat{\beta}_k^{(t)})$$

4. Logistic Regression and Neural Nets

A Neural Net is a way to represent *Multivariate Logistic Regression* with logistic activation and Bernoulli score function.

Logistic Activation

$$f_2(x)=\frac{1}{1+\exp(-\sum_j \beta_{j2}x_j)}$$



$$\nabla Q_2=\begin{bmatrix}\frac{\partial Q_2}{\partial \beta_{02}}\\ \frac{\partial Q_2}{\partial \beta_{12}}\\ \frac{\partial Q_2}{\partial \beta_{22}}\end{bmatrix}$$

Bernoulli Likelihood Score

$$Q_2=\sum_i y_{i2}(\sum_j \beta_{j2}x_{ij})-\sum_i \ln[1+\exp(\sum_j \beta_{j2}x_{ij})]$$

Derivatives

$$\frac{\partial Q_2}{\partial \beta_{j2}}=\sum_i x_{ij}y_{i2}-\sum_i \frac{x_{ij}}{1+\exp(-\sum_j \beta_{j2}x_{ij})}$$

$j=0,1,2$

Gradient

$$\nabla Q_2=\left(\frac{\partial Q_2}{\partial \beta_{j2}}\right)$$

Gradient Descent

$$\hat{\beta}_2^{(t+1)}=\hat{\beta}_2^{(t)}-\gamma\nabla Q_2(\hat{\beta}_2^{(t)})$$

4. Logistic Regression and Neural Nets

Two independent parallel regressions yield the *"Multivariate" Logistic Regression* results we are after.

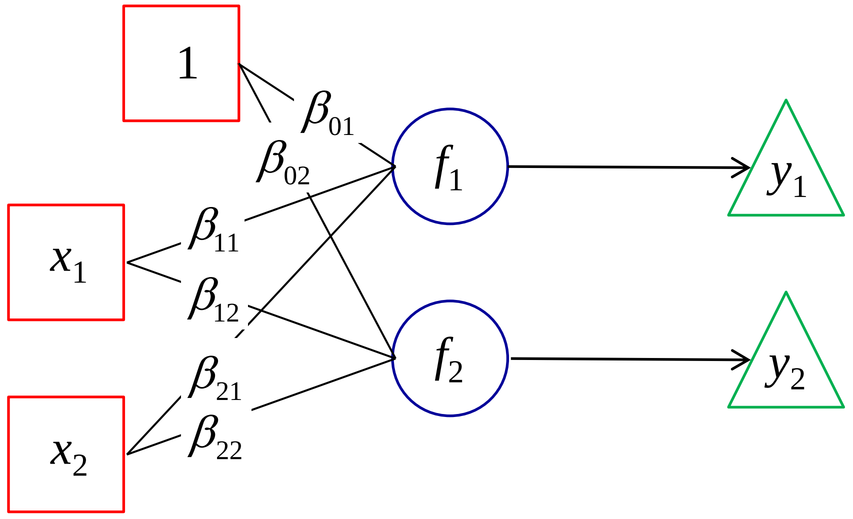
Estimated Coefficients

$$\hat{B} = (\hat{\beta}_1, \hat{\beta}_2)$$

Maximizing Score Functions

$$Q_1 = \sum_i y_{i1}(\sum_j \beta_{j1} x_{ij}) - \sum_i \ln[1+\exp(\sum_j \beta_{j1} x_{ij})]$$

$$Q_2 = \sum_i y_{i2}(\sum_j \beta_{j2} x_{ij}) - \sum_i \ln[1+\exp(\sum_j \beta_{j2} x_{ij})]$$



$$\nabla Q_k = \begin{bmatrix} \frac{\partial Q_k}{\partial \beta_{0k}} \\ \frac{\partial Q_k}{\partial \beta_{1k}} \\ \frac{\partial Q_k}{\partial \beta_{2k}} \end{bmatrix}$$

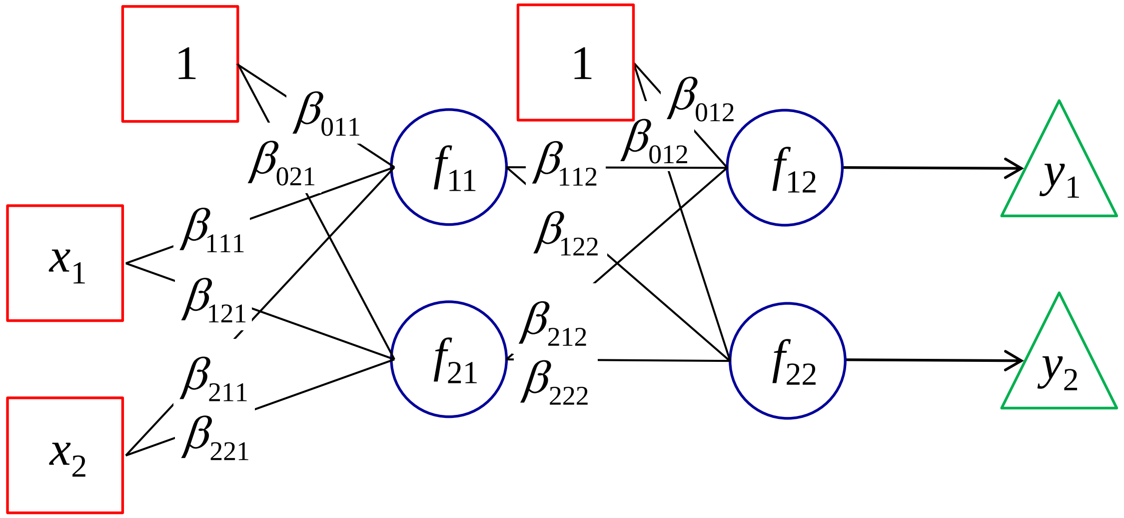
Independently Estimated Via Gradient Descent

$$\hat{\beta}_1^{(t+1)} = \hat{\beta}_1^{(t)} + \gamma \nabla Q_1(\hat{\beta}_1^{(t)})$$

$$\hat{\beta}_2^{(t+1)} = \hat{\beta}_2^{(t)} + \gamma \nabla Q_2(\hat{\beta}_2^{(t)})$$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one "hidden" layer.
The outputs from one layer become the inputs to the next.

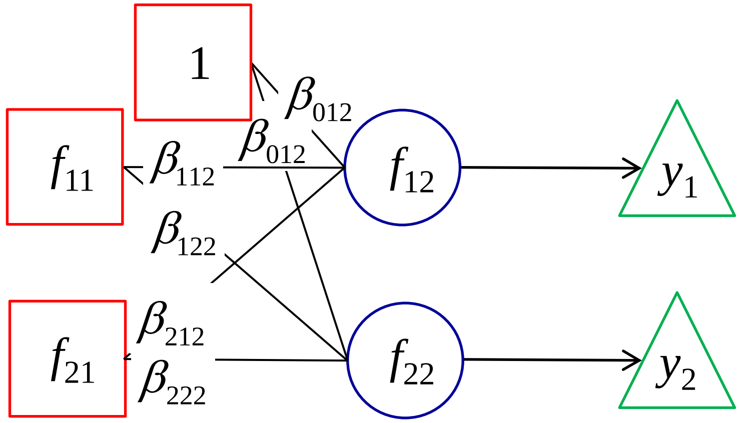


Let's go through the process as multiple one layer Neural Nets,
from right to left, Backpropagation.

coefficient	node	layer
$j = 0,1,2$	$k = 1,2$	$\ell = 1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one "hidden" layer.
The outputs from one layer become the inputs to the next.

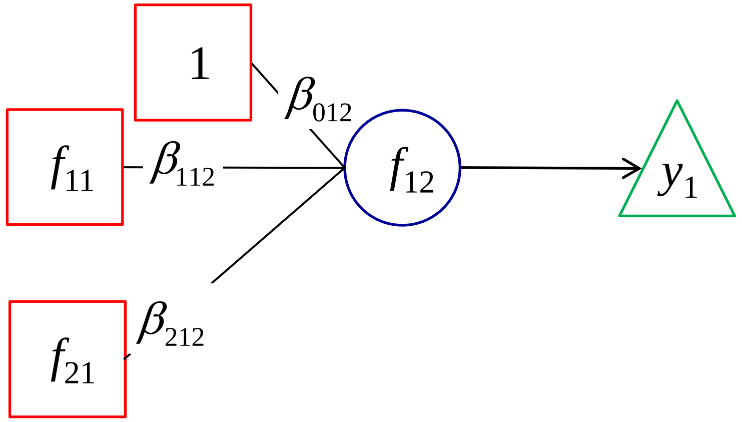


We consider the first output layer as input to the second layer.
Estimate coefficients.

coefficient	node	layer
$j=0,1,2$	$k=1,2$	$\ell=1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one "hidden" layer.
The outputs from one layer become the inputs to the next.

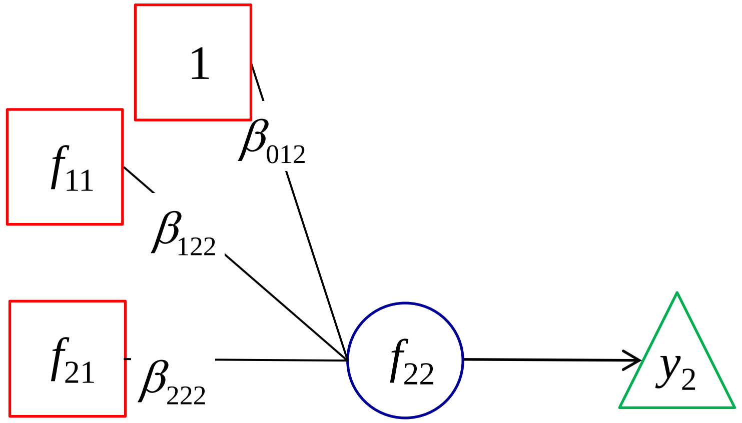


And first focus only on the first node.
Estimate coefficients.

coefficient	node	layer
$j=0,1,2$	$k=1,2$	$\ell=1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one “hidden” layer.
The outputs from one layer become the inputs to the next.

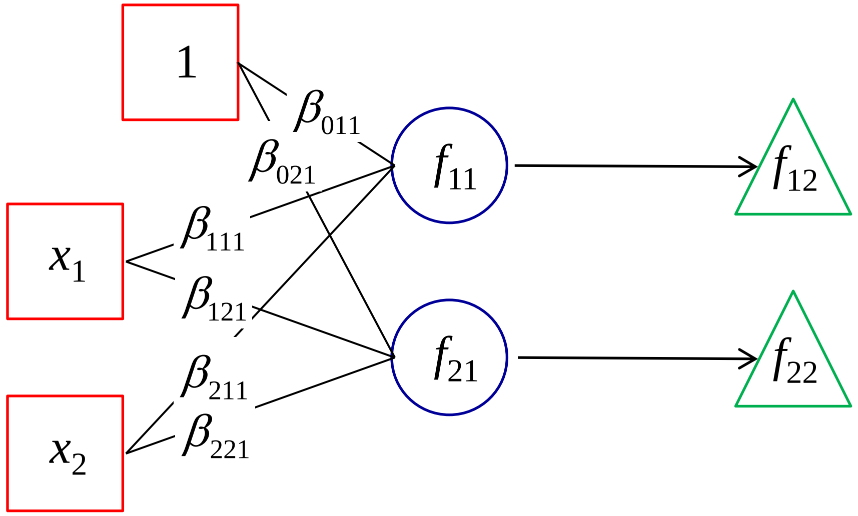


Then focus on the second node.
Estimate coefficients.

coefficient	node	layer
$j = 0,1,2$	$k = 1,2$	$\ell = 1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one "hidden" layer.
The outputs from one layer become the inputs to the next.

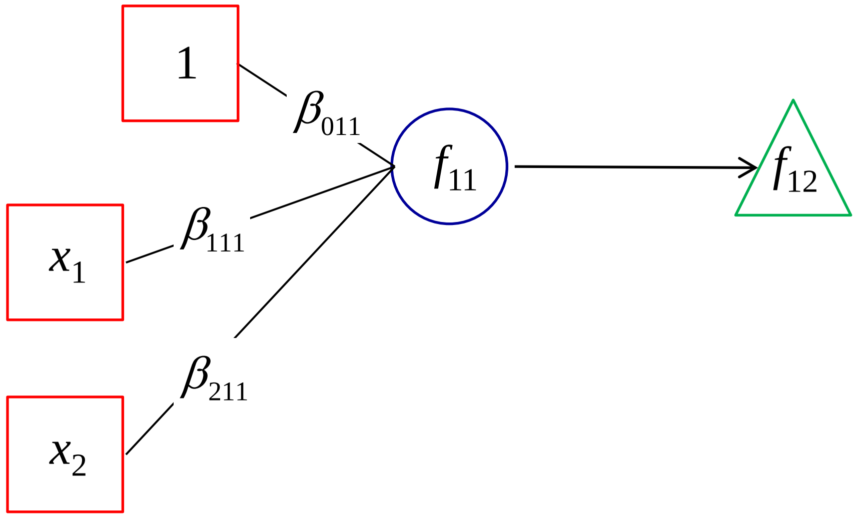


Let's go through the process as multiple one layer Neural Nets,
from right to left, Backpropagation.

coefficient	node	layer
$j = 0,1,2$	$k = 1,2$	$\ell = 1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one “hidden” layer.
The outputs from one layer become the inputs to the next.

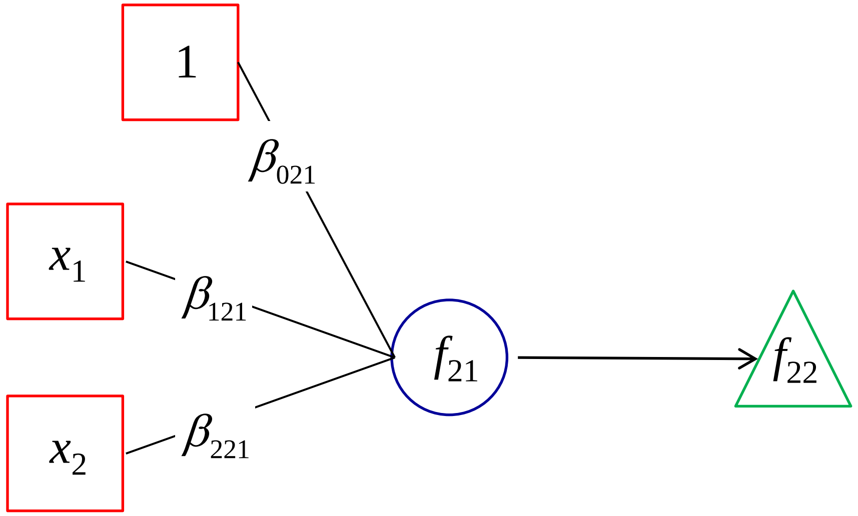


And first focus only on the first node.
Estimate coefficients.

coefficient	node	layer
$j=0,1,2$	$k=1,2$	$\ell=1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one "hidden" layer.
The outputs from one layer become the inputs to the next.

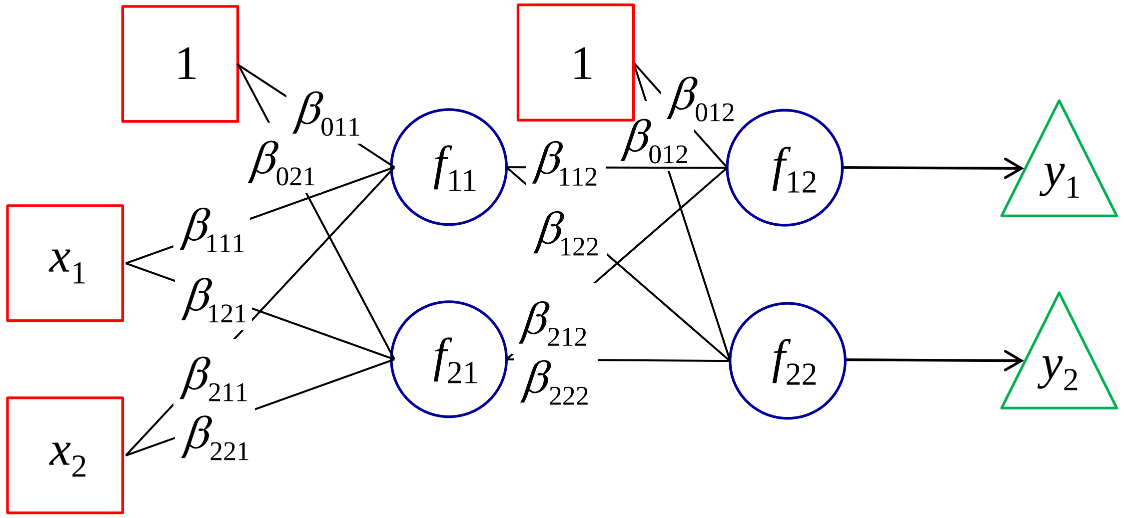


Then focus on the second node.
Estimate coefficients.

coefficient	node	layer
$j = 0,1,2$	$k = 1,2$	$\ell = 1,2$

5. Multi-Layer deep Neural Nets

Neural Nets can have more than one “hidden” layer.
The outputs from one layer become the inputs to the next.

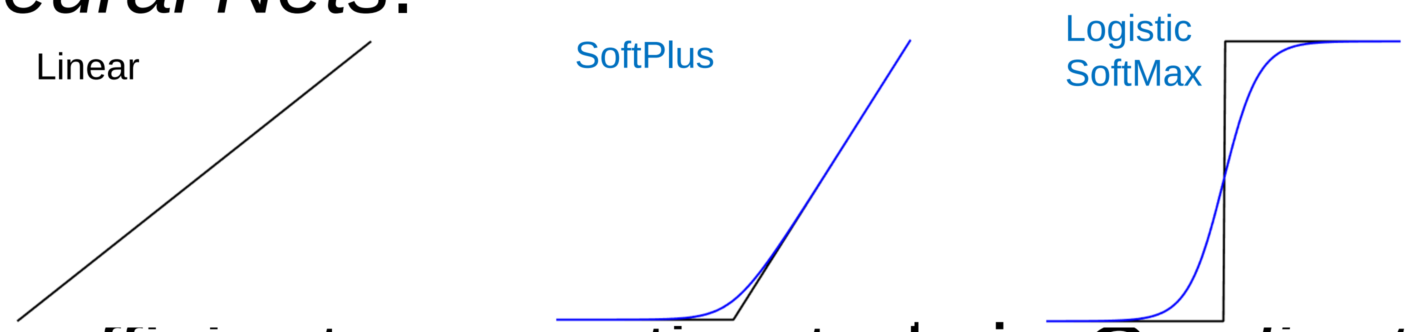


And hence solve the two or multi layer problem.

coefficient	node	layer
$j = 0,1,2$	$k = 1,2$	$\ell = 1,2$

5. Discussion

Linear, *Logistic*, and *Non-Linear Regression* can be represented as *Neural Nets*.

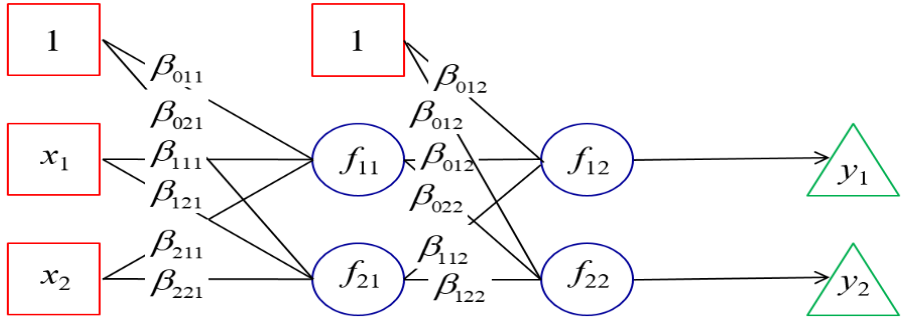


Coefficients are estimated via *Gradient Descent*.

$$\nabla Q = \left(\frac{\partial Q}{\partial \beta_j} \right) \qquad \hat{\beta}^{(t+1)} = \hat{\beta}^{(t)} - \gamma \nabla Q(\hat{\beta}^{(t)})$$

Discussed foundational ideas of Neural Nets.

These ideas can be expanded in many directions.



Discussion

Questions?

Homework 9

1. Implement the *Artificial Neural Net* simple *Linear Regression* example using *Gradient Descent* for the (x,y) data.

Data (x,y)
(1,1.4)
(2,2.3)
(3,1.7)
(4,3.0)
(5,3.4)

$t=0$ $(\hat{\beta}_0^{(0)},\hat{\beta}_1^{(0)})=(1.50,1.00)$ Initial Value

Run data through with $\hat{\beta}^{(t)}=(\hat{\beta}_0^{(t)},\hat{\beta}_1^{(t)})'$

Calculate $\nabla Q(\hat{\beta}^{(t)})=-\frac{2}{n}(X'y-X'X\hat{\beta}^{(t)})$ (Direction), $\gamma=.0001$ (Step Size)

Calculate new $\hat{\beta}^{(t+1)}=\hat{\beta}^{(t)}-\gamma\nabla Q(\hat{\beta}^{(t)})$

Until convergence. Keep track of each $(\beta_0^{(t)},\beta_1^{(t)})$ and $Q(\beta_0^{(t)},\beta_1^{(t)})$.

Calculate $Q(\beta_0,\beta_1)$ for a range of coefficient values and plot the surface.

Plot the Gradient Descent values on the same plot.

Homework 9

$$\underset{n\times p}{Y} = \underset{n\times(q+1)}{X}\ \underset{(q+1)\times p}{\beta} + \underset{n\times p}{E}$$

2. Generate one data set of son/daughter heights y_{is} and y_{id}

$$(y_{is}, y_{id}) = (1, x_{if}, x_{im})\begin{pmatrix} \beta_{0s} & \beta_{0d} \\ \beta_{1s} & \beta_{1d} \\ \beta_{2s} & \beta_{2d} \end{pmatrix} + (\varepsilon_{is}, \varepsilon_{id}) \qquad (\varepsilon_{is}, \varepsilon_{id})' \sim N(0, \Sigma) \quad i = 1, \ldots, n \qquad B = \begin{pmatrix} 14.1 & 10.8 \\ 0.41 & 0.39 \\ 0.43 & 0.43 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 2.250 & 1.125 \\ 1.125 & 2.250 \end{pmatrix}$$

using combinations

$$x_f = \{66,67,68,69,70,71,72,73,74,75\}$$
$$x_m = \{60,61,62,63,64,65,66,67,68,69\}$$

$$\hat{B}^{(t+1)} = \hat{B}^{(t)} - \gamma \frac{2}{n}(X'Y - X'X\hat{B}^{(t)})$$
$$\hat{\Sigma} = \frac{1}{n-q-1}(Y - X\hat{B})'(Y - X\hat{B})$$

estimate B and hence Σ using Gradient Descent to train this Artificial Neural Network with linear activation. Compare to $\hat{B} = (X'X)^{-1}X'Y$.

Homework 9

3. Assume β_{01}=-7.0, β_{11}=0.3, β_{02},=-5.0, β_{12}=0.2. Use x=(10:5:60)′.
   Generate two simulated data sets, $(y_{11},\ldots,y_{n1})$, $(y_{12},\ldots,y_{n2})$.
   Estimate β_{01}, β_{11}, β_{02}, β_{12}, and p_i's by Gradient Descent and hence train the network with logistic activation and Bernoulli score function.

   On one plot graph (x_{ik},y_{ik}) data, (x_{i1},p_{i1}) and (x_{i2},p_{i2}).
   Make a surface plot and show your estimated and true values for each node (k).

   Make comparisons and comment on results.

Homework 9

4. Invent your own example of multivariate linear regression with two layers. Set your own true parameter values, generate a simulated data set and estimate the parameters. Comment.

5*. Invent your own example of multivariate logistic regression with two layers. Set your own true parameter values, generate a simulated data set and estimate the parameters. Comment.

*Show off question.

Homework 9

6*. Invent your own example of ReLU non-linear linear regression with one layer. Set your own true parameter values, generate a simulated data set and estimate the parameters. Comment.

*Show off question.