Regression and Classification

Dr. Tushar Sandhan

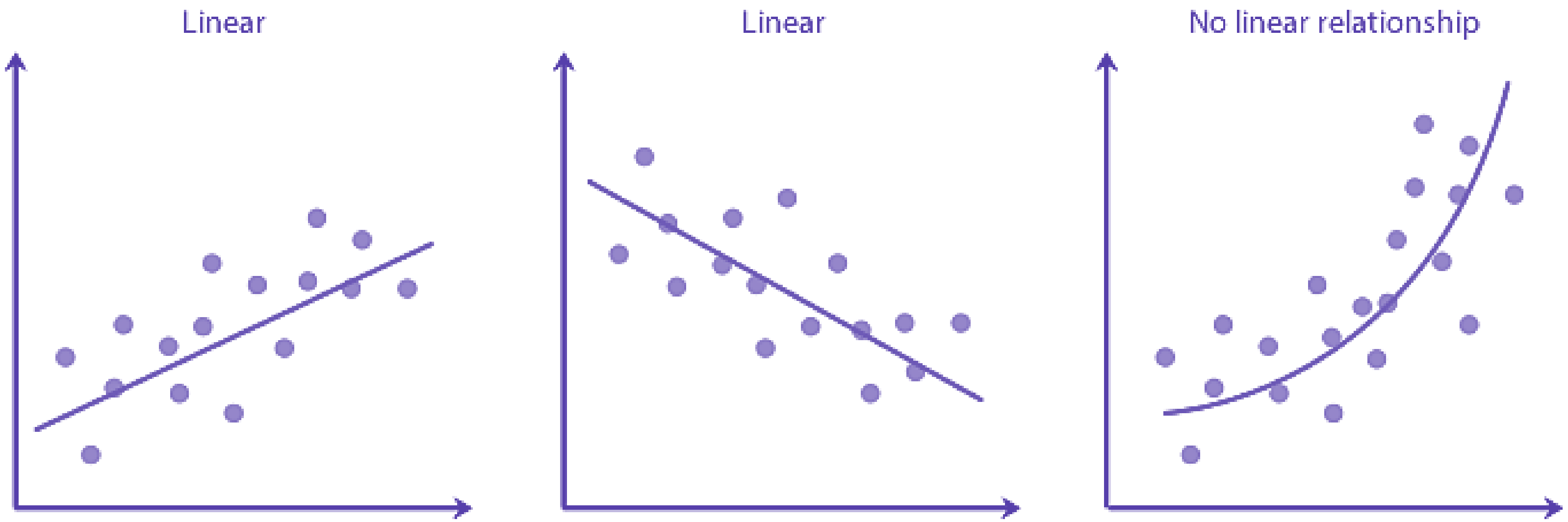
sandhan@iitk.ac.in

Content

- Curve fitting
- Modeling
  - Scalar linear model
- Loss function
- Multivariable regression
  - Vectorization
- Gradient descent optimization
- Polynomial regression
- Classification
  - Logistic regression

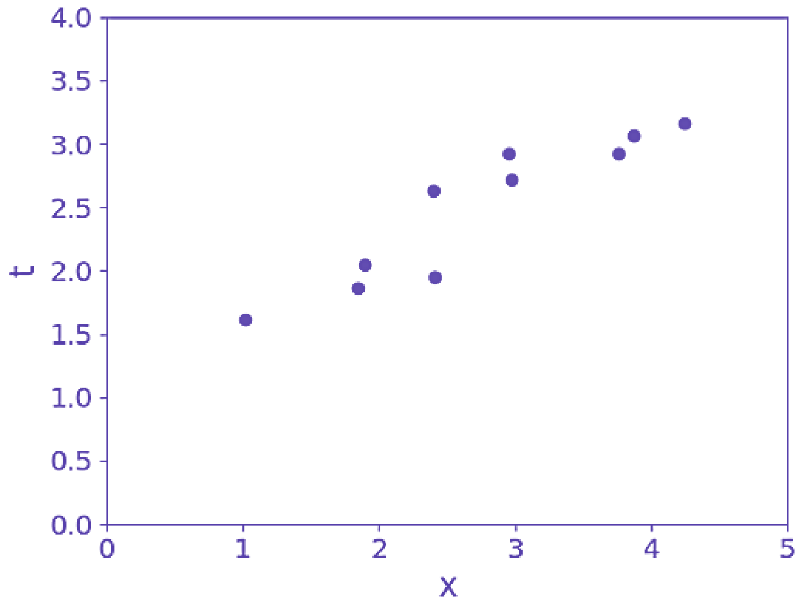
sandhan@iitk.ac.in

Curve fitting



Regression problem

- Given the inputs x, we need to estimate the targets t
- $\{(\boldsymbol{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)}$ inputs
  - $t^{(i)}$ targets

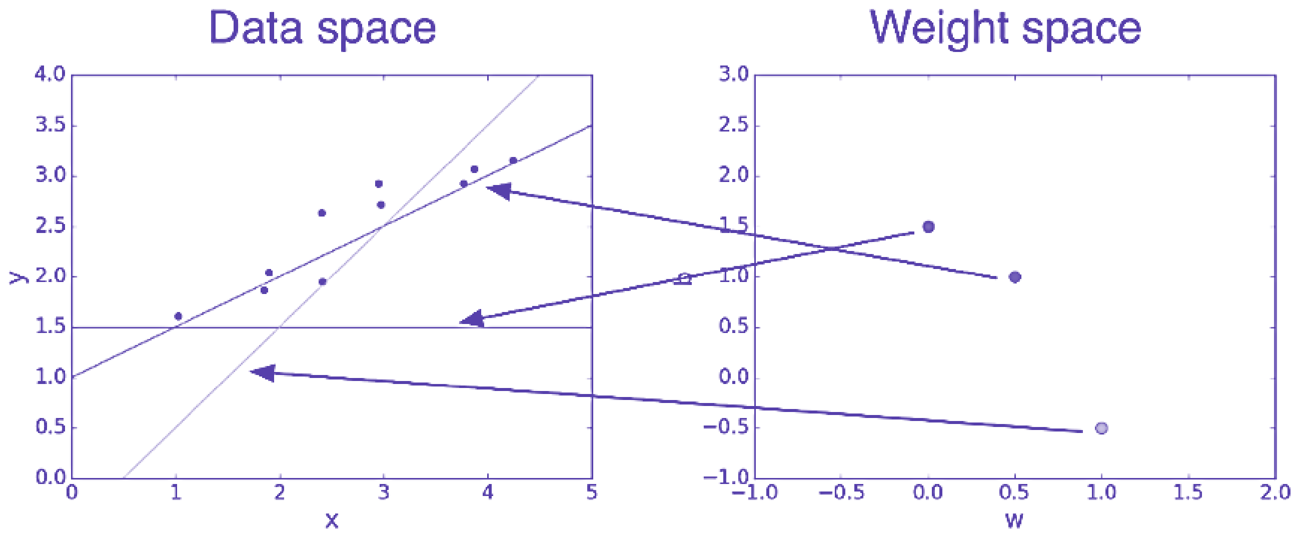


- Model:
  - $y = wx + b$

sandhan@iitk.ac.in

Modeling

- Given the inputs x, we need to estimate the targets t
- $\{(\boldsymbol{x}^{(i)}, t^{(i)})\}_{i=1}^{N}$
  - $\boldsymbol{x}^{(i)}$ inputs
  - t^{i} targets



- Model:
  - $y = wx + b$

y is the **prediction**

w is the **weight**

b is the **bias**

w and b together are the **parameters**

Settings of the parameters are called **hypotheses**

sandhan@iitk.ac.in

Loss function

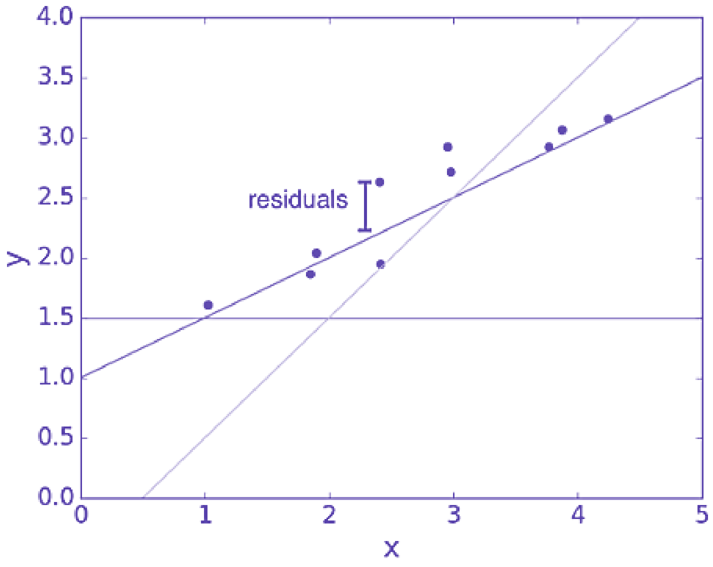
- Loss function: sq. error
- Residual
  - $y - t$
- Cost function:

$$\mathcal{L}(y,t) = \frac{1}{2}(y-t)^2$$

$$\mathcal{J}(w,b) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - t^{(i)}\right)^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\left(wx^{(i)} + b - t^{(i)}\right)^2$$

- Model:
  - $y = wx + b$



sandhan@iitk.ac.in

Multivariate regression

- MISO
  - Not much different than single input (scalar) case
- MIMO
  - targets also multiple
  - Multilayer perceptrons

- MISO:

$$y = \sum_j w_j x_j + b$$

Solution (1)

Partial derivatives: derivatives of a multivariate function with respect to one of its arguments.

$$y = \sum_j w_j x_j + b$$

$$\frac{\partial}{\partial x_1} f(x_1, x_2) = \lim_{h \to 0} \frac{f(x_1 + h, x_2) - f(x_1, x_2)}{h}$$

To compute, take the single variable derivatives, pretending the other arguments are constant.
Example: partial derivatives of the prediction y

$$\begin{aligned} \frac{\partial y}{\partial w_j} &= \frac{\partial}{\partial w_j}\left[\sum_{j'} w_{j'} x_{j'} + b\right] \\ &= x_j \\ \frac{\partial y}{\partial b} &= \frac{\partial}{\partial b}\left[\sum_{j'} w_{j'} x_{j'} + b\right] \\ &= 1 \end{aligned}$$

sandhan@iitk.ac.in

Solution (1)

Chain rule for derivatives:

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}y}\frac{\partial y}{\partial w_j}$$
$$= \frac{\mathrm{d}}{\mathrm{d}y}\left[\frac{1}{2}(y-t)^2\right]\cdot x_j$$
$$= (y-t)x_j$$
$$\frac{\partial \mathcal{L}}{\partial b} = y - t$$

The minimum must occur at a point where the partial derivatives are zero.

$$\frac{\partial \mathcal{J}}{\partial w_j} = 0 \qquad \frac{\partial \mathcal{J}}{\partial b} = 0.$$

Cost derivatives (average over data points):

$$\frac{\partial \mathcal{J}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\, x_j^{(i)}$$
$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} y^{(i)} - t^{(i)}$$

Optimal weights:

$$\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t}$$

sandhan@iitk.ac.in

Solution (2)

- Gradient descent
- Iterative algo
  - apply update repeatedly until convergence

Observe:

- if $\partial \mathcal{J} / \partial w_j > 0$, then increasing w_j increases $\mathcal{J}$.
- if $\partial \mathcal{J} / \partial w_j < 0$, then increasing w_j decreases $\mathcal{J}$.

Solution (2): Gradient descent

- Gradient descent
- Iterative algo
  - apply update repeatedly until convergence

The following update decreases the cost function:

$$
\begin{aligned}
w_j &\leftarrow w_j - \alpha \frac{\partial \mathcal{J}}{\partial w_j} \\
&= w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, x_j^{(i)}
\end{aligned}
$$

α is a **learning rate**. The larger it is, the faster $\mathbf{w}$ changes.

- We'll see later how to tune the learning rate, but values are typically small, e.g. 0.01 or 0.0001

Update rule vector form

This gets its name from the **gradient**:

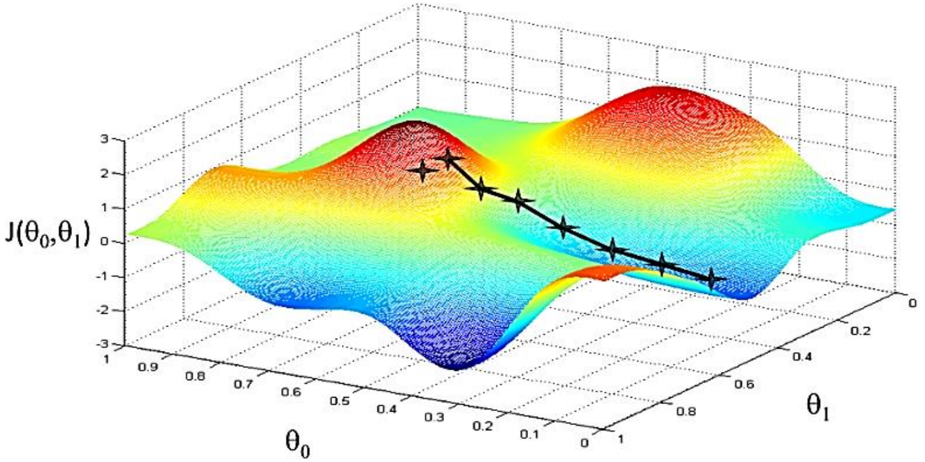
$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{J}}{\partial w_D} \end{pmatrix}$$

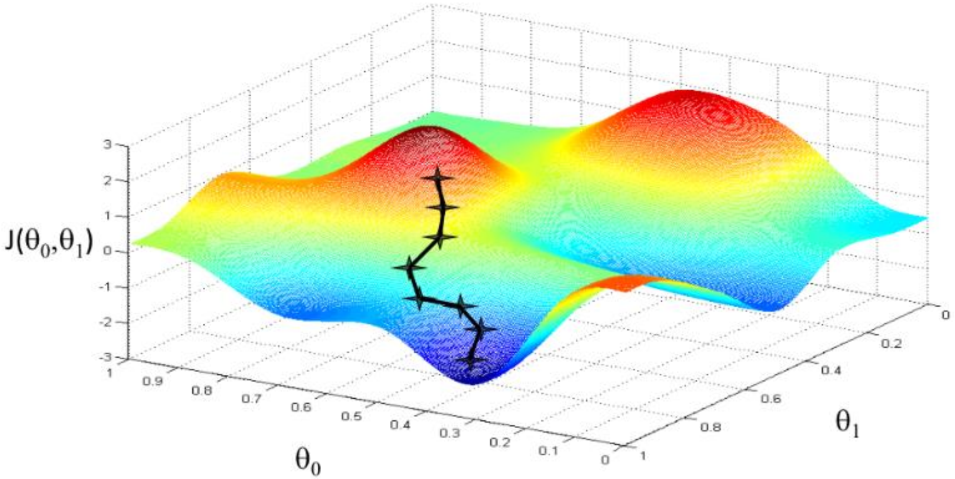
- This is the direction of fastest increase in $\mathcal{J}$.

Update rule in vector form:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{J}}{\partial \mathbf{w}} \\ &= \mathbf{w} - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})\, \mathbf{x}^{(i)} \end{aligned}$$

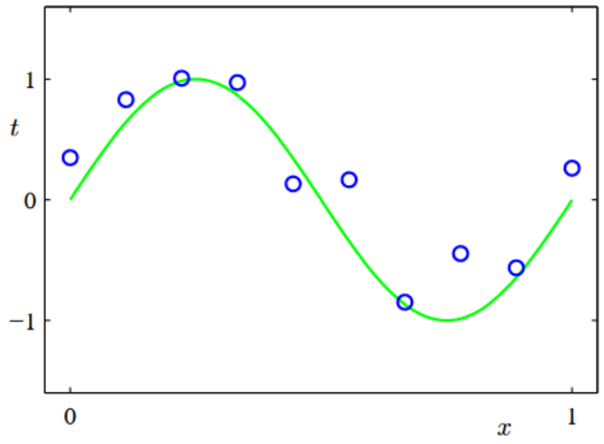
Hence, gradient descent updates the weights in the direction of fastest *decrease*.





Polynomial regression

- Regression using higher degree polynomials
- Data distributed non-linear way



$$y = w_3 x^3 + w_2 x^2 + w_1 x + w_0$$

Define the **feature map**

$$\psi(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix}$$

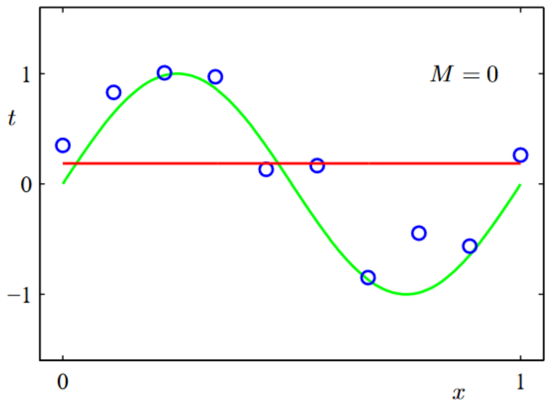
Polynomial regression model:

$$y = \mathbf{w}^\top \psi(x)$$

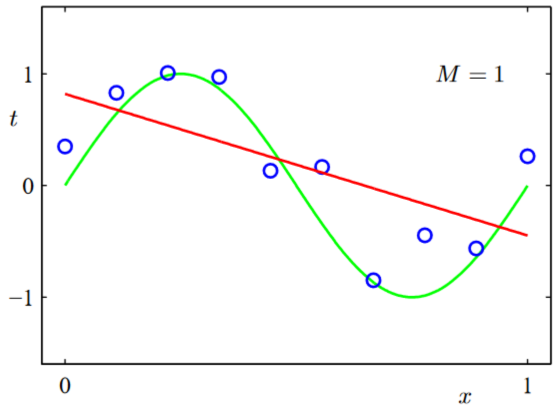
sandhan@iitk.ac.in

Hyper-parameter: degree of polynomial

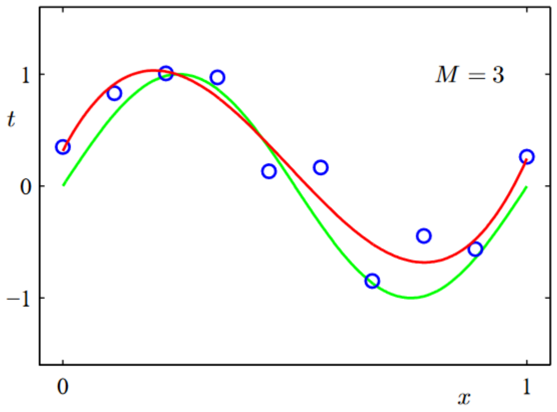
$y = w_0$



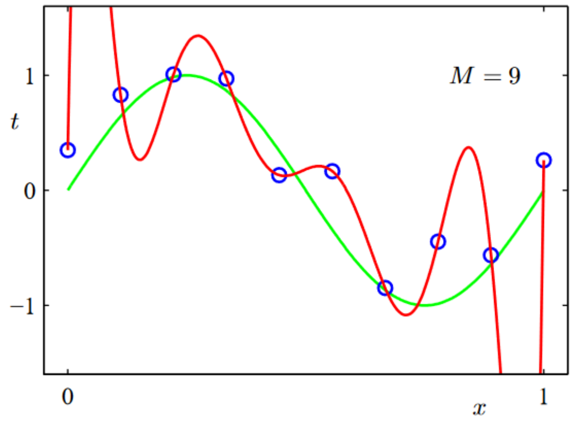
$y = w_0 + w_1 x$



$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$

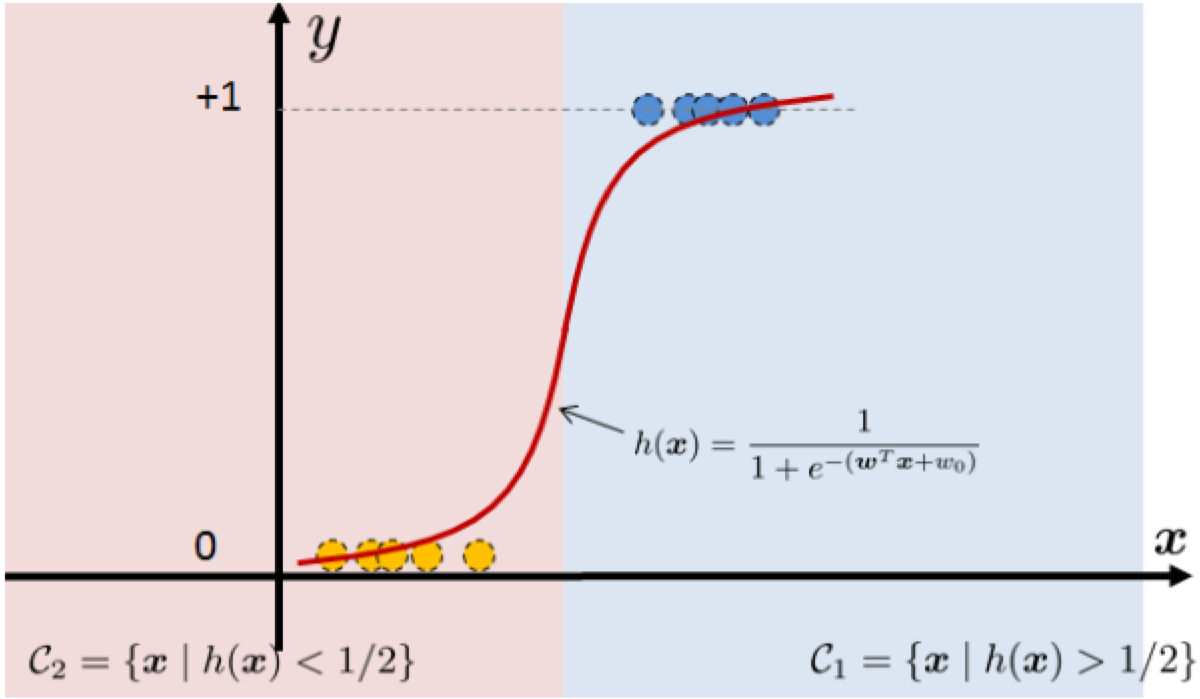


$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots + w_9 x^9$



Classification: logistic regression

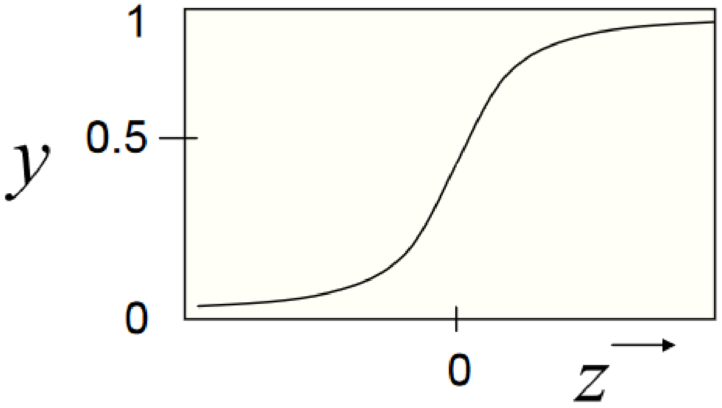
- Target distribution is quantized (class labels/probabilities)



$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$



sandhan@iitk.ac.in

Classification

- Logistic regression can be considered as last layer of neural network
- Inputs are x^i, weights are w
- Sigmoid function is non-linear activation
- Similar manner, we compare prediction error and minimize loss by updating weights

$$\boxed{\begin{aligned} J(\theta) &= \sum_{n=1}^{N} \mathcal{L}(h_\theta(\boldsymbol{x}_n), y_n) \\ &= \sum_{n=1}^{N} -\left\{ y_n \log h_\theta(\boldsymbol{x}_n) + (1 - y_n) \log(1 - h_\theta(\boldsymbol{x}_n)) \right\} \end{aligned}}$$

Thank you