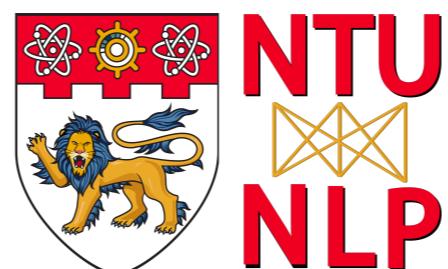


# Deep Learning for Natural Language Processing

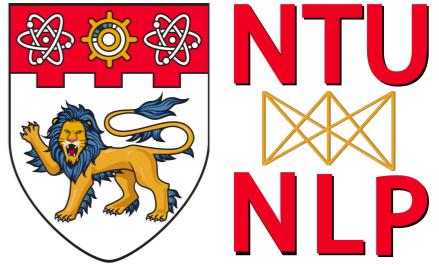
**CE/CZ 4045**

Shafiq Joty

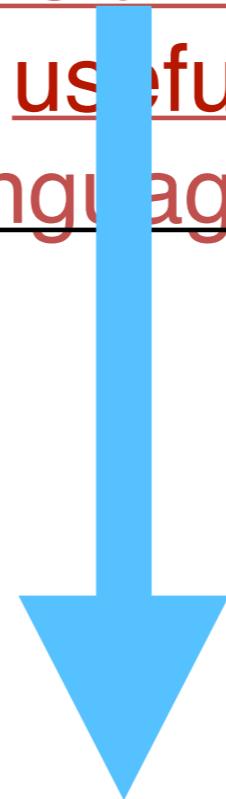


Lecture 2: Machine Learning Basics

# What is NLP



We study formalisms, models and algorithms to allow computers to perform useful tasks involving knowledge about human languages.

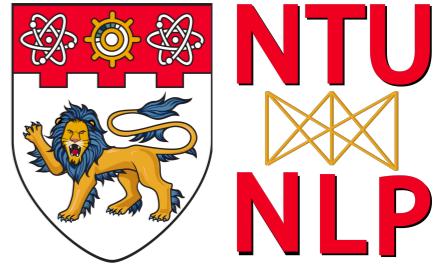


**Different Neural Models (This half)**

**Traditionally (First half):**

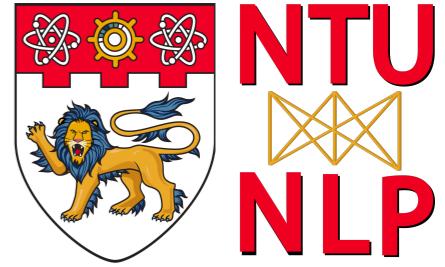
**Finite State Automata, Markov Models, Graphical Models**

# Lecture Plan



- What is Machine Learning?
  - Supervised vs. unsupervised learning
- Linear Regression
- Logistic Regression
  - Parameter estimation
  - Gradient-based optimization & SGD
  - Multi-class classification

# What is Machine Learning

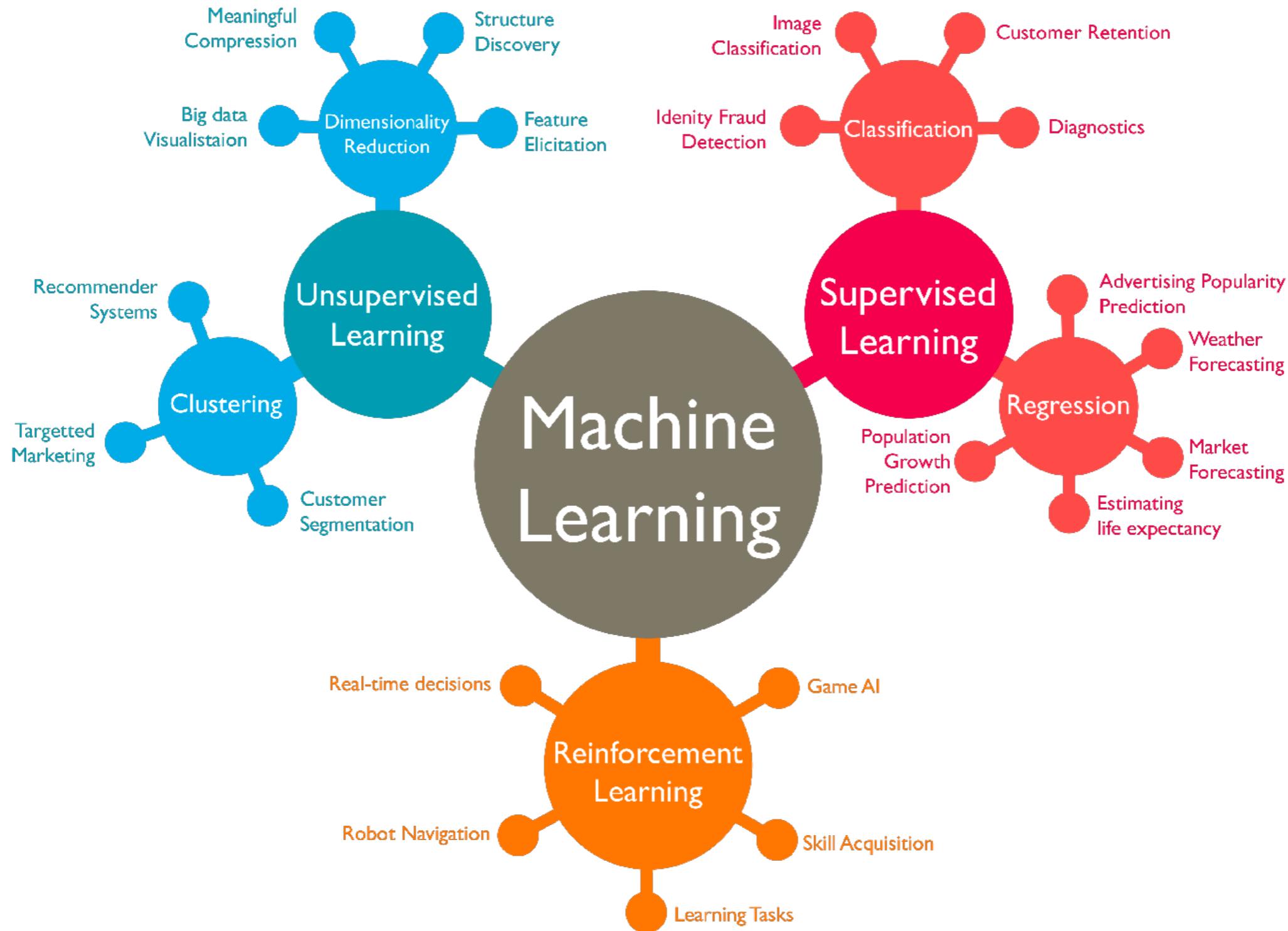
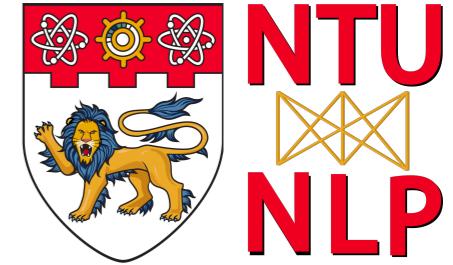


A set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty.

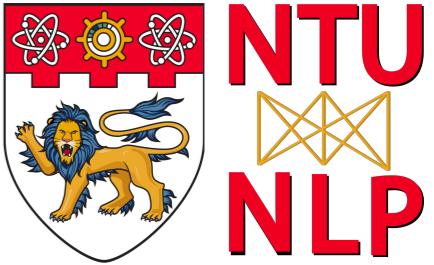
# Types of ML

- **Supervised/Predictive learning:**  
To learn a **mapping** from inputs X to outputs Y, given a labelled set of input-output pairs.
- **Unsupervised/Descriptive learning:**  
To find **interesting patterns** in the data from given input.  
Sometimes called Knowledge Discovery.
- **Reinforcement learning:**  
To learn **how to act or behave** when given occasional reward or punishment signals.

# Types of ML



# Supervised Learning



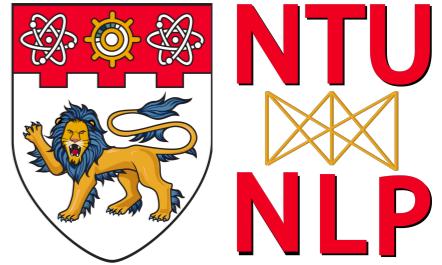
**Classification:** To learn a mapping from inputs  $x$  to outputs  $y$ , where  $y \in \{1, \dots, C\}$ , with  $C$  being the number of classes, which means  $y$  is categorical.

- If  $C = 2$ , this is called **binary** classification (in which case we often assume  $y \in \{0, 1\}$ )
- If  $C > 2$ , this is called **multiclass** classification.

## Examples

- (1) Document classification and email spam filtering
- (2) Sentiment classification
- (3) Image classification and handwriting recognition
- (4) Face detection and recognition

# Supervised Learning

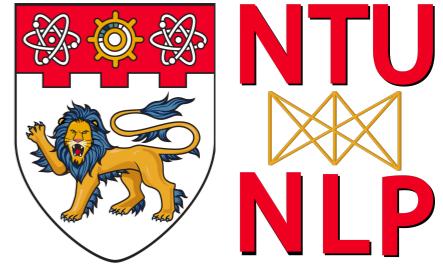


**Regression:** To learn a mapping from inputs  $x$  to outputs  $y$ , where  $y$  is **real-valued**.

## Examples

1. Predict tomorrow's stock market price given current market conditions and other possible side information.
2. Predict the age of a viewer watching a given video on YouTube.
3. Predict the location in 3d space of a robot arm end effector, given control signals (torques) sent to its various motors.
4. Predict the temperature at any location inside a building using weather data, time, door sensors.

# Unsupervised Learning



We are just given input data, without any outputs. The goal is to discover **interesting structure** in the data; this is sometimes called knowledge discovery.

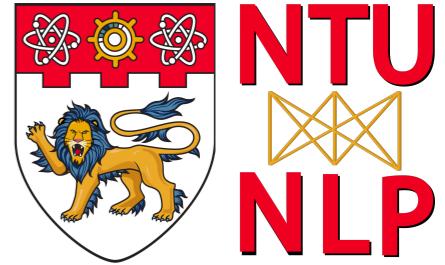
1. **Discovering clusters:** To cluster data into groups and estimate which cluster each data sample belongs to.
2. **Discovering latent factors:** To project the data to a lower dimensional subspace which captures the essence of the data. It is also called "Dimension Reduction". (eg, PCA)
3. **Discovering graph structure.**
4. Matrix completion: To "fill in" **missing data**. Such as image inpainting, collaborative filtering in the movie market.

# Machine Learning Steps

3 Steps of parametric model:

- ① Model Assumption. Also known as **inductive bias**.
- ② Fit/Learn/Train the model parameters.
- ③ Apply/Test the model.

# Linear Regression



## Problem Setup

Training Data is represented by:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$$

Supervised

Here,

$\mathcal{D}$  is called the **training set**.

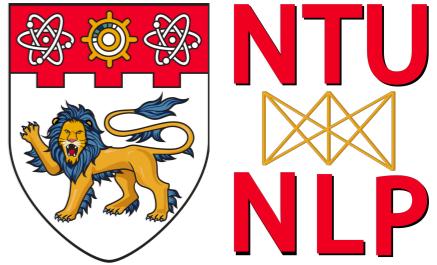
$N$  is the **number of training example**.

$\mathbf{x}_i$  is a  **$d$ -dimensional vector of features** (also called **attributes** or **covariates**) i.e. #features is  $d$ .

$y_i$  is called **output** or **response variable**.

For regression:  $y_i$  is **real-valued** i.e.  $y_i \in \mathbb{R}$

# Linear Regression

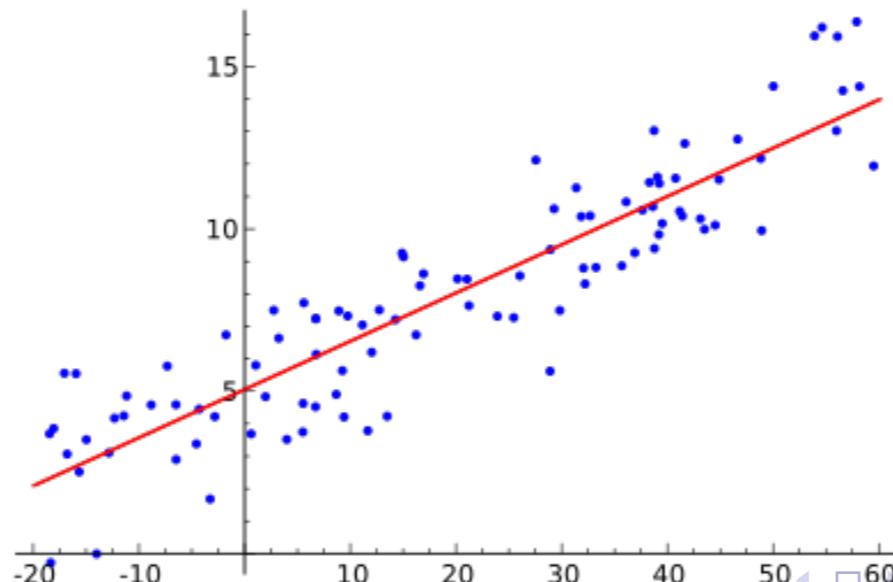


What is it?

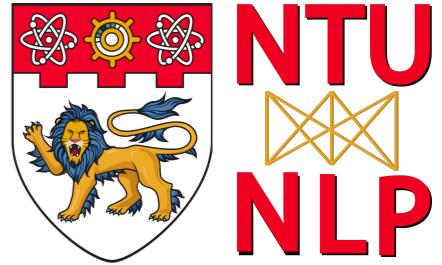
The basic idea behind regression is that, you want to model the relationship between a real valued outcome variable  $y$ , and a vector of explanatory variables  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ .

A linear regression relates  $y$  to a linear predictor function of  $\mathbf{x}$ .  
For a given data point  $i$ , the linear function is of the form:

$$\hat{y}_i = w_0 + w_1 x_{i1} + \dots + w_d x_{id}$$



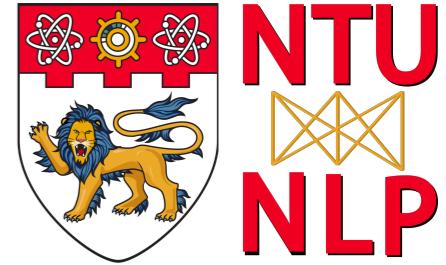
# Linear Regression



## Applications

- Predict tomorrow's stock market price given current market conditions and other possible side information.
- Predict the age of a viewer watching a given video on YouTube.
- Predict the location in 3D space of a robot arm end effector, given control signals (torques) sent to its various motors.
- Predict the amount of Prostate Specific Antigen (PSA) in the body as a function of a number of different clinical measurements.
- Predict the temperature at any location inside a building using weather data, time, door sensors, etc.

# Linear Regression



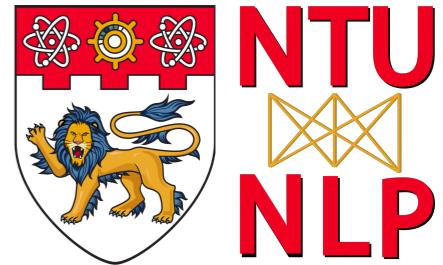
## Least Square Formulation of the Loss

In this formulation, **the least squares fit** is the line that **minimizes** the sum of the squared distances between observed data and predicted values, i.e. it minimizes the **Residual Sum of Squares (RSS)**:

$$\arg \min_{\mathbf{w}} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

where  $\hat{y}_i = w_0 + w_1 x_i$  is the **predicted outcome** for the  $i^{th}$  observation.

# Linear Regression

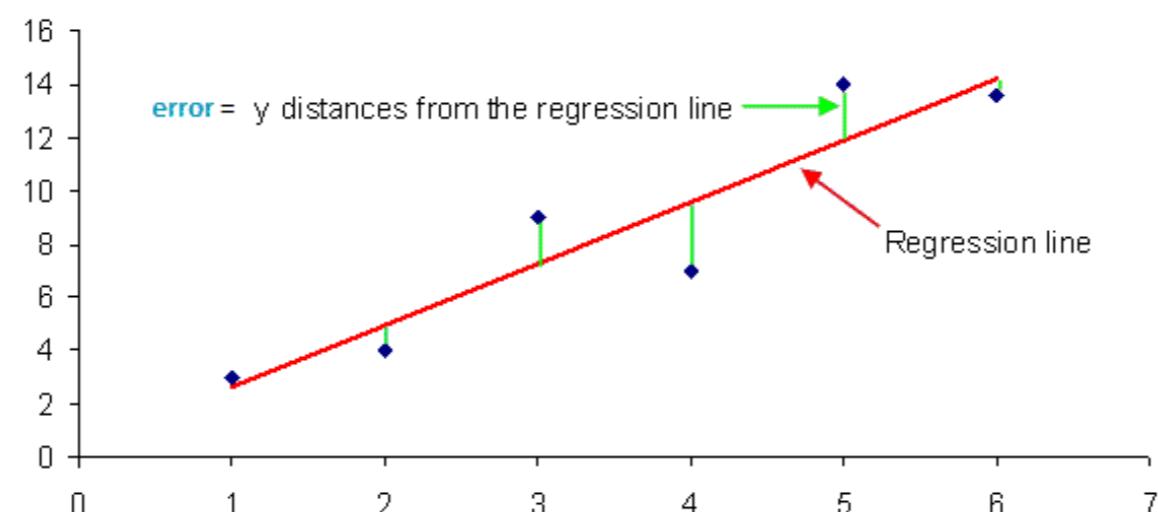


## Least Square Formulation

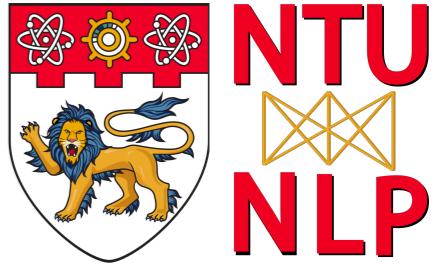
We can write the **actual value** of the  $i^{th}$  observation as:

$$y_i = w_0 + w_1 x_i + \epsilon_i$$

where  $\epsilon_i$  is the **residual, or error**, we get for the  $i^{th}$  observation, i.e. the difference between our linear predictions and the true response.



# Linear Regression



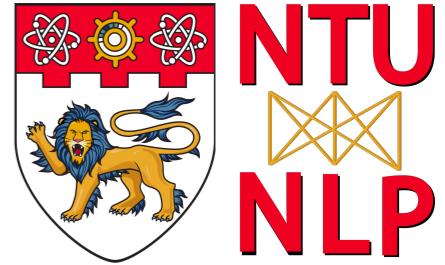
## Least Square Formulation

So our **objective function** is:

$$\begin{aligned} J(\theta) &= \arg \min_{\theta} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \\ &= \sum_{i=0}^N (y_i - \mathbf{x}_i^\top \mathbf{w})^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \end{aligned}$$

Vector notation

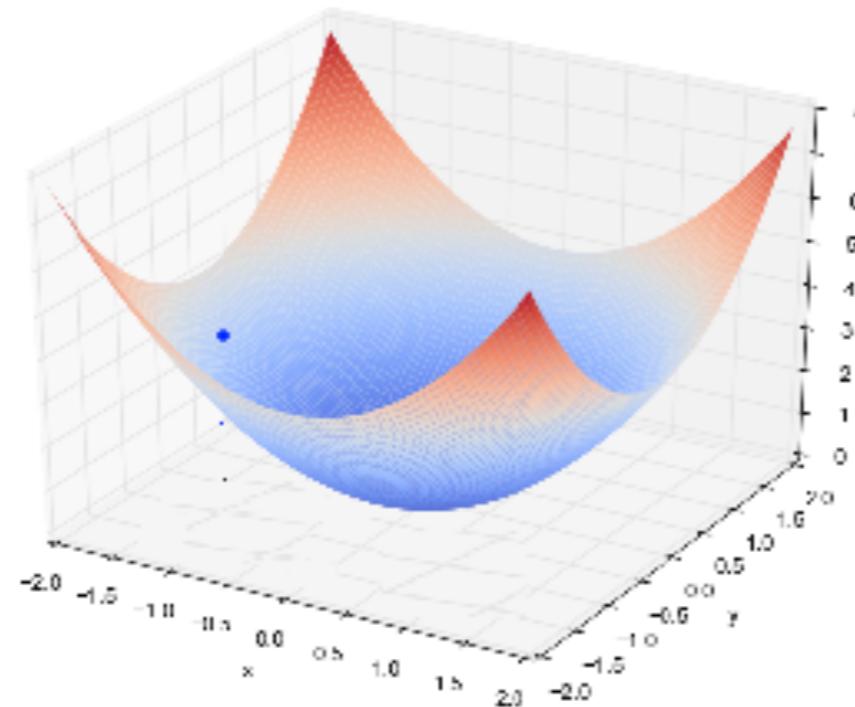
# Linear Regression



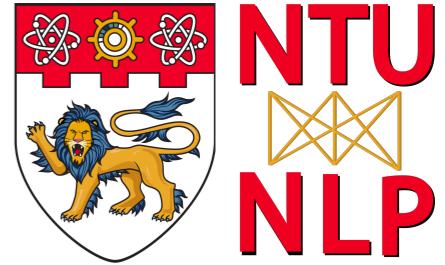
## Least Square Formulation

Notice that, we want to minimize  $J(\theta)$ .

We see that  $J(\theta)$  is **Quadratic**. Hence it will be **Bowl Shaped**.



# Linear Regression



## Least Square Formulation

$$\begin{aligned} J(\theta) &= (y - \mathbf{X}\mathbf{w})^\top(y - \mathbf{X}\mathbf{w}) = (y^\top - \mathbf{w}^\top\mathbf{X}^\top)(y - \mathbf{X}\mathbf{w}) \\ &= y^\top y - y^\top\mathbf{X}\mathbf{w} - \mathbf{w}^\top\mathbf{X}^\top y + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} \end{aligned}$$

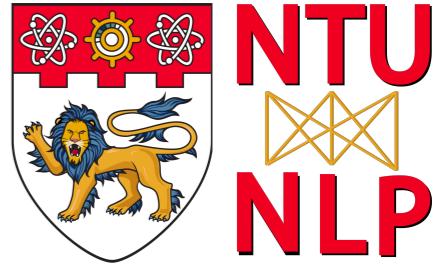
Taking partial derivative with respect to  $\mathbf{w}$ :

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} J(\theta) &= \frac{\partial}{\partial \mathbf{w}}(y^\top y - y^\top\mathbf{X}\mathbf{w} - \mathbf{w}^\top\mathbf{X}^\top y + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}) \\ &= 0 - 2\mathbf{X}^\top y + \mathbf{X}^\top\mathbf{X}\mathbf{w} \end{aligned}$$

Equating  $\frac{\partial}{\partial \mathbf{w}} J(\theta)$  to 0 we get,

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} J(\theta) = 0 &= -2\mathbf{X}^\top y + \mathbf{X}^\top\mathbf{X}\mathbf{w} \\ \mathbf{w} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top y \quad \text{Closed form Solution} \end{aligned}$$

# Linear Regression



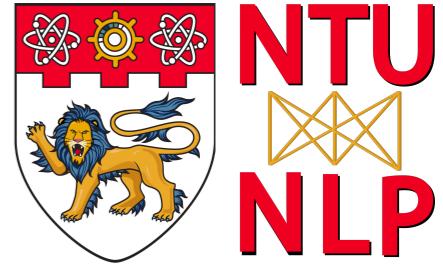
## Probabilistic Formulation

If we ever want to understand linear regression from a **Bayesian perspective** we need to start thinking **probabilistically**.

We need to *flip things over* and instead of thinking about the line minimizing the cost, think about it as **maximizing the likelihood of the observed data**.

As we will see, this amounts to the **exact same thing** - *mathematically speaking* - it's just a different way of looking at it.

# Linear Regression



## Probabilistic Formulation

If we ever want to understand linear regression from a **Bayesian perspective** we need to start thinking **probabilistically**.

We have seen previously:

$$y_i = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i$$

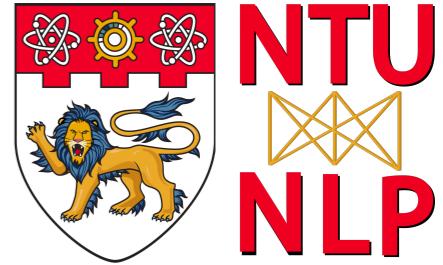
We often assume that  $\epsilon_i$  **normally distributed** with a mean of 0 i.e.

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

So, From the above equation we can say that:

$$y_i \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}_i, \sigma^2)$$

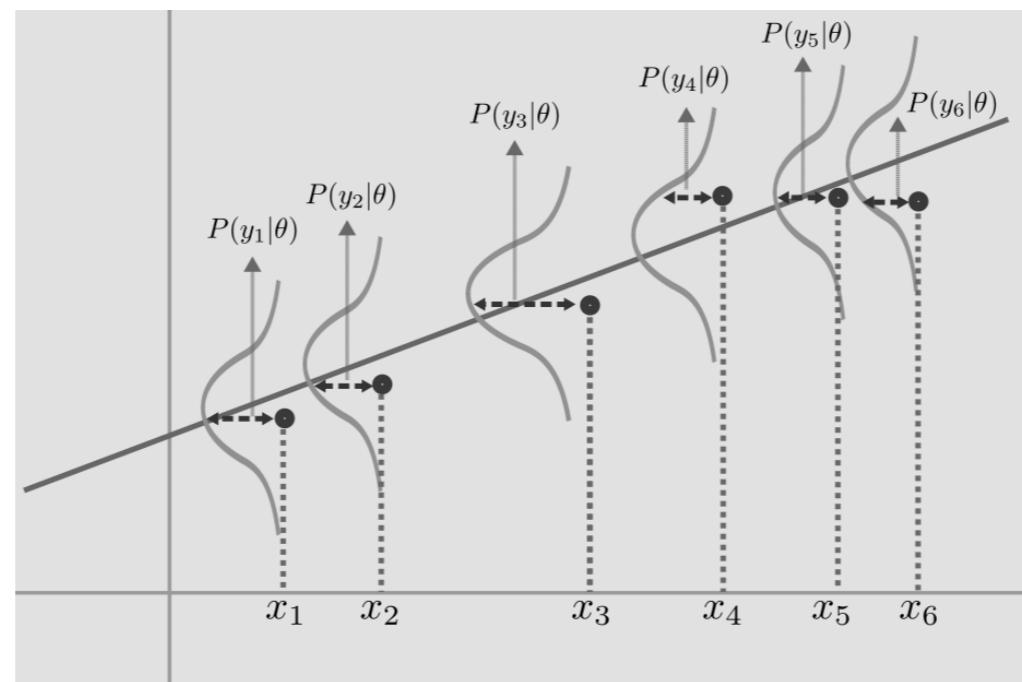
# Linear Regression



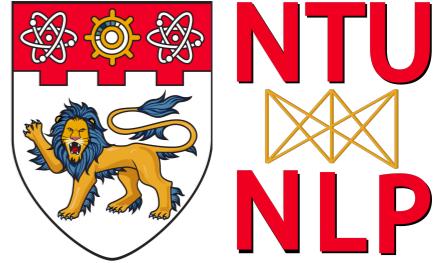
## Probabilistic Formulation

It is common to assume the training examples are **Independent and Identically Distributed**, commonly abbreviated to **IID**.

So, we assume all our training examples are **Normally distributed** with mean  $\mathbf{w}^T \mathbf{x}_i$  and variance  $\sigma^2$ .



# Linear Regression



## Maximum Likelihood Estimation

We can write Linear regression as a model of the form:

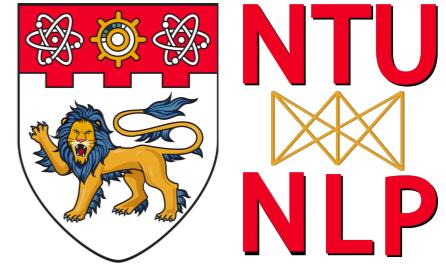
$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y|\mathbf{w}^T \mathbf{x}, \sigma^2)$$

### Maximum likelihood Estimation

A common way to estimate the parameters of a statistical model is to compute the Maximum likelihood Estimation(MLE), which is defined as:

$$\hat{\theta} \triangleq \operatorname{argmax}_{\theta} \log p(\mathcal{D}|\theta)$$

# Linear Regression



## Maximum Likelihood Estimation

### Log Likelihood

log function is a monotonically increasing function.

We know that maximizing the log of a value with respect to some parameter, coincides with maximizing the value itself.

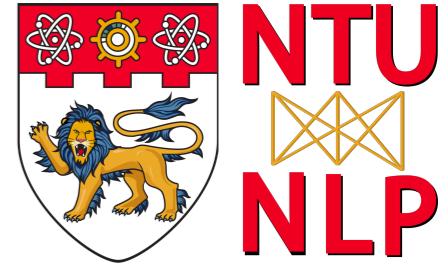
So, the log-likelihood can be written as:

$$\begin{aligned}\ell(\boldsymbol{\theta}) &\triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \sum_{i=1}^N \log \left[ \left( \frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left( -\frac{1}{2\sigma^2} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \right) \right] \\ &= \frac{-1}{2\sigma^2} \text{RSS}(\mathbf{w}) - \frac{N}{2} \log(2\pi\sigma^2)\end{aligned}$$

RSS stands for Residual Sum of Squares

$$\text{RSS}(\mathbf{w}) \triangleq \sum_{i=1}^N (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

# Linear Regression



## Maximum Likelihood Estimation

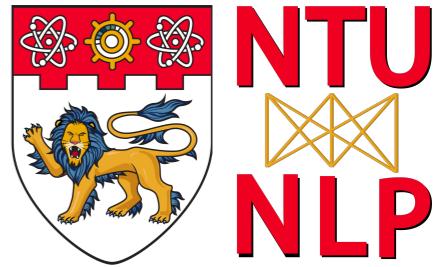
RSS is defined by:

$$RSS(\mathbf{w}) \triangleq \sum_{i=1}^N (y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

The RSS is also called the **Sum of Squared Errors**.

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

# Logistic Regression



## Problem Setup

Training Data is represented by

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$$

Supervised

Here

$\mathcal{D}$  is called the **training set**.

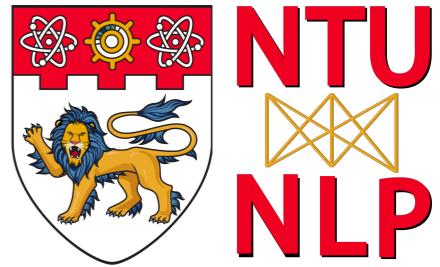
$N$  is the **number of training examples**.

$x_i$  is a **d-dimensional vector of features** (also called attributes or covariates) i.e. #features is d.

$y_i$  is called **output or response variable**.

For classification:  $y_i$  is **categorial** i.e.  $y_i \in \{1, \dots, C\}$

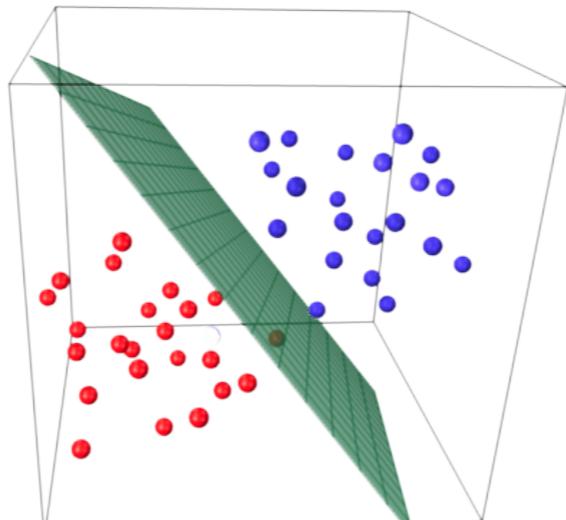
# Logistic Regression



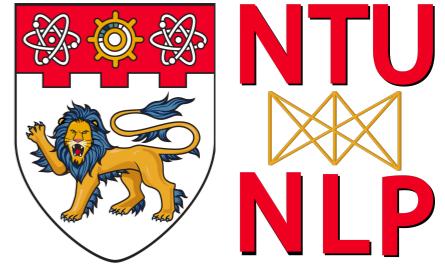
$y_i$  is **categorial** i.e.  $y_i \in \{1, \dots, C\}$

First, we will consider the case of **binary response variable** that is, where it can take only two values, "0" and "1".

Cases where the response variable has more than two outcome categories may be analyzed in **Multinomial Logistic Regression**.



# Logistic Regression



We can **generalize** linear regression to the (binary) classification setting by making two changes.

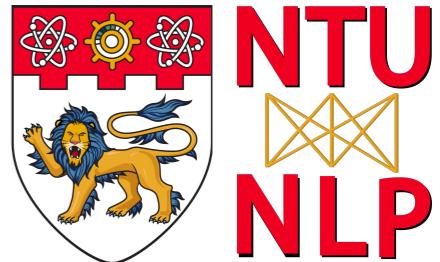
**First**, we replace the Gaussian distribution for  $y$  with a *Bernoulli distribution*. That is, we use

$$\mathcal{P}(y|\mathbf{x}, \mathbf{w}) = \text{Ber}(y|\mu(\mathbf{x}))$$

where  $\mu(\mathbf{x}) = \mathbb{E}[y|\mathbf{x}] = \mathcal{P}(y = 1|\mathbf{x})$

**Second**, we compute a linear combination of the inputs as before, but then we *pass this through a sigmoid function* that ensures  $0 \leq \mu(\mathbf{x}) \leq 1$ . So we define  $\mu(\mathbf{x}) = \text{sigm}(\mathbf{w}^\top \mathbf{x})$

# Logistic Regression



## Sigmoid Function

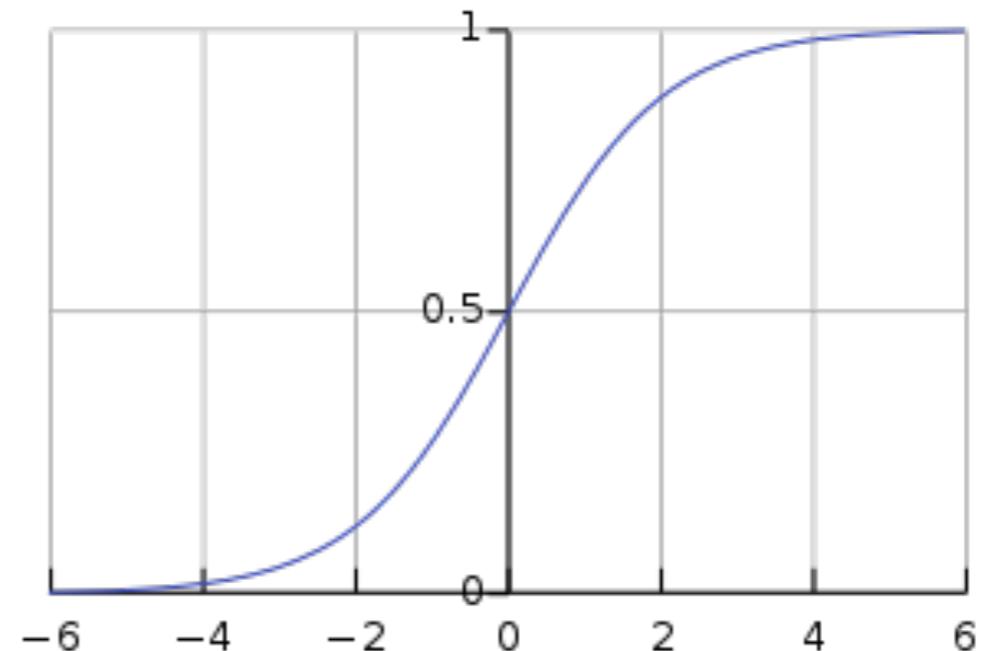
Sigmoid is a **squashing function**.

It maps the whole real line to [0,1].

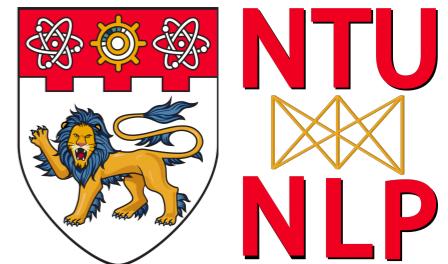
It is also known as *logistic* or *logit* function

The term “sigmoid” means S-shaped. This is defined as:

$$\text{sigm}(\eta) \triangleq \frac{1}{1 + \exp(-\eta)} = \frac{e^\eta}{e^\eta + 1}$$

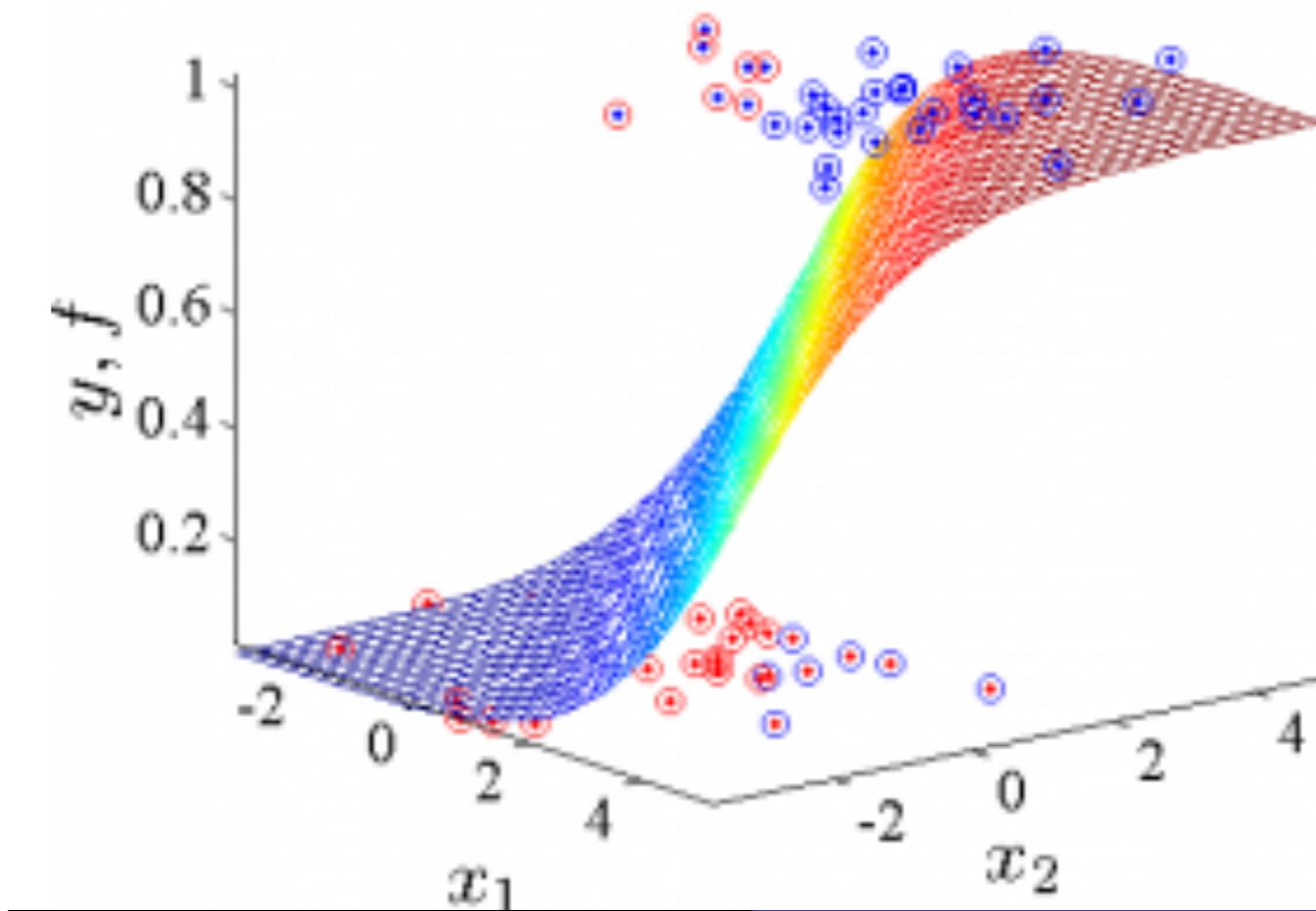


# Logistic Regression

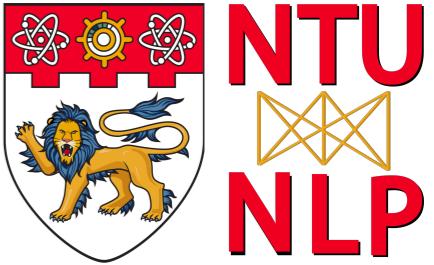


## Model Definition

$$\mathcal{P}(y|\mathbf{x}, \mathbf{w}) = Ber(y|sigm(\mathbf{w}^\top \mathbf{x}))$$



# Logistic Regression



— At the decision boundary, both classes are **equiprobable**. Thus:

$$\mathcal{P}(y = 1 | \mathbf{x}, \mathbf{w}) = \mathcal{P}(y = 0 | \mathbf{x}, \mathbf{w})$$

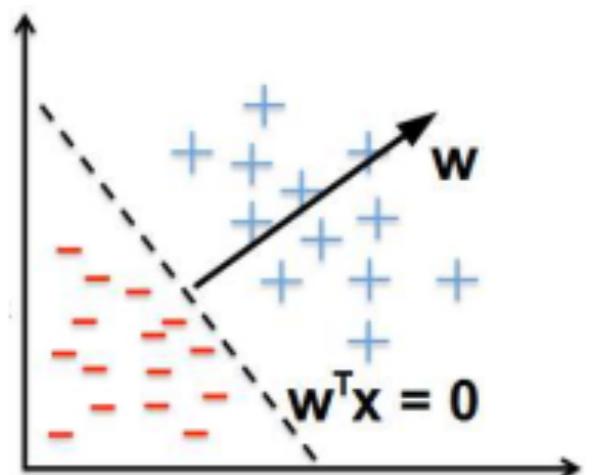
$$\frac{\exp(\mathbf{w}^T \mathbf{x})}{1 + \exp(\mathbf{w}^T \mathbf{x})} = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x})}$$

$$\exp(\mathbf{w}^T \mathbf{x}) = 1$$

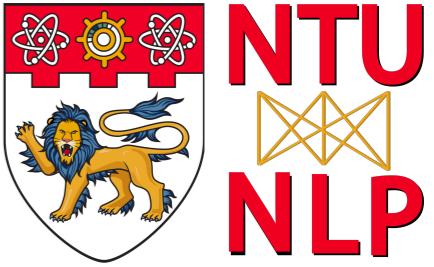
$$\mathbf{w}^T \mathbf{x} = 0$$

Thus the decision boundary of Logistic Regression is nothing but a **linear hyperplane**.

- High positive score  $\mathbf{w}^T \mathbf{x}$ : **High probability** of label 1
- High negative score  $\mathbf{w}^T \mathbf{x}$ : **Low probability** of label 1



# Logistic Regression



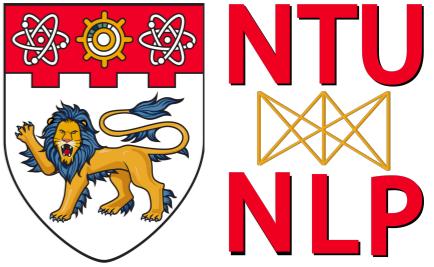
Instead of **maximizing** the log-likelihood, we can equivalently **minimize** the **negative log likelihood** or NLL.

The negative log-likelihood for logistic regression is given by

$$\begin{aligned} NLL(\mathbf{w}) &= - \sum_{i=1}^N \log[\mu_i^{1(y_i=1)} \times (1 - \mu_i^{1(y_i=0)})] \\ &= - \sum_{i=1}^N [y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i)] \end{aligned}$$

This is also called the **Cross Entropy Error Function**.

# Logistic Regression



## Entropy

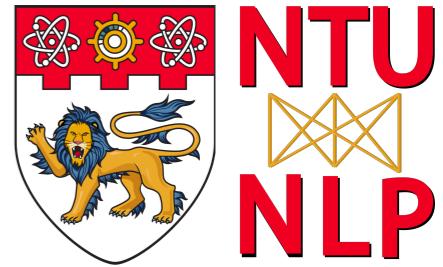
The entropy of a random variable  $X$  with distribution  $p$ , denoted by  $\mathbb{H}(X)$ , is a measure of its **uncertainty**. It is defined by

$$\mathbb{H}(X) \triangleq - \sum_{k=1}^K p(X=k) \log p(X=k)$$

## Cross Entropy

Cross entropy of two probability distribution  $p$  and  $q$  is denoted by  $\mathbb{H}(p, q)$  and is defined by  $\mathbb{H}(p, q) \triangleq - \sum_k p_k \log q_k$

# Logistic Regression



Unlike linear regression, we can no longer write down the MLE in **closed form**. Instead, we need to use an **optimization algorithm** to compute it.

For this, we need to derive the **gradient** and **Hessian**.

$$\mathbf{g} = \frac{d}{d\mathbf{w}} f(\mathbf{w}) = - \sum_i (\mu_i - y_i) \mathbf{x}_i = \mathbf{X}^\top (\boldsymbol{\mu} - \mathbf{y})$$

$$\mathbf{H} = \frac{d}{d\mathbf{w}} \mathbf{g}(\mathbf{w})^\top = - \sum_i (\nabla_{\mathbf{w}} \mu_i \mathbf{x}_i^\top) = - \sum_i \mu_i (1 - \mu_i) \mathbf{x}_i \mathbf{x}_i^\top = \mathbf{X}^\top \mathbf{S} \mathbf{X}$$

where  $\mathbf{S} \triangleq \text{diag}(\mu_i(1 - \mu_i))$

One can also show that **H** is **positive definite**.

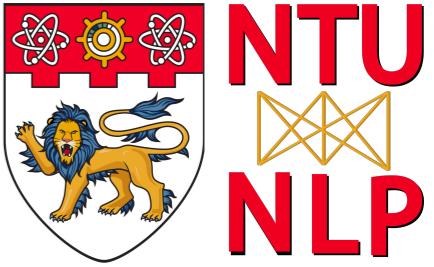
Hence the **NLL** is **convex** and has a **unique global minimum**.

# Multiclass Logistic Regression

Logistic regression can be extended to handle more than two classes. This is called **Multi-class logistic regression**.

Also called **Multinomial logistic regression**. Sometimes called a **Maximum Entropy Classifier**.

# Multiclass Logistic Regression



In this case, response variable  $y_i \in \{0, 1, 2, \dots, (k - 1)\}$

**Model Specification:**

$$\mathcal{P}(y = c | \mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_{c'=1}^C \exp(\mathbf{w}_{c'}^\top \mathbf{x})}$$

Softmax

# Multiclass Logistic Regression

Let,  $\mu_{ic} = \mathcal{P}(y_i = c | \mathbf{x}_i, \mathbf{W}) = \mathcal{S}(\eta_i)_c$

Where  $\eta_i = \mathbf{W}^\top \mathbf{x}_i$

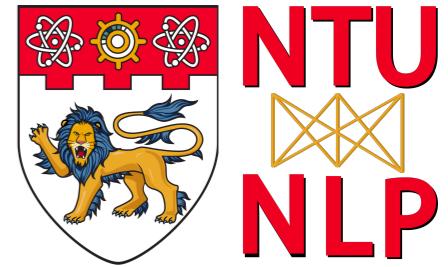
Also Let,  $y_{ic} = \mathbb{I}(y_i = c)$

With this the log likelihood can be written as:

$$\begin{aligned}\ell(\mathbf{W}) &= \log \prod_{i=1}^N \prod_{c=1}^C \mu_{ic}^{y_{ic}} = \sum_{i=1}^N \sum_{c=1}^C y_{ic} \log \mu_{ic} \\ &= \sum_{i=1}^N \left[ \left( \sum_{c=1}^C y_{ic} \mathbf{w}_c^\top \mathbf{x}_i \right) - \log \left( \sum_{c'=1}^C \exp(\mathbf{w}_{c'}^\top \mathbf{x}_i) \right) \right]\end{aligned}$$

Can do MLE for  $\mathbf{W}$  similar to the binary logistic regression case.

# Gradient Descent



## Parameter estimation

Perhaps the simplest algorithm for unconstrained optimization is gradient descent, also known as **steepest descent**. This can be written as follows:

$$\theta_{k+1} = \theta_k - \eta_k g_k$$

where  $\eta_k$  is the step size or learning rate.

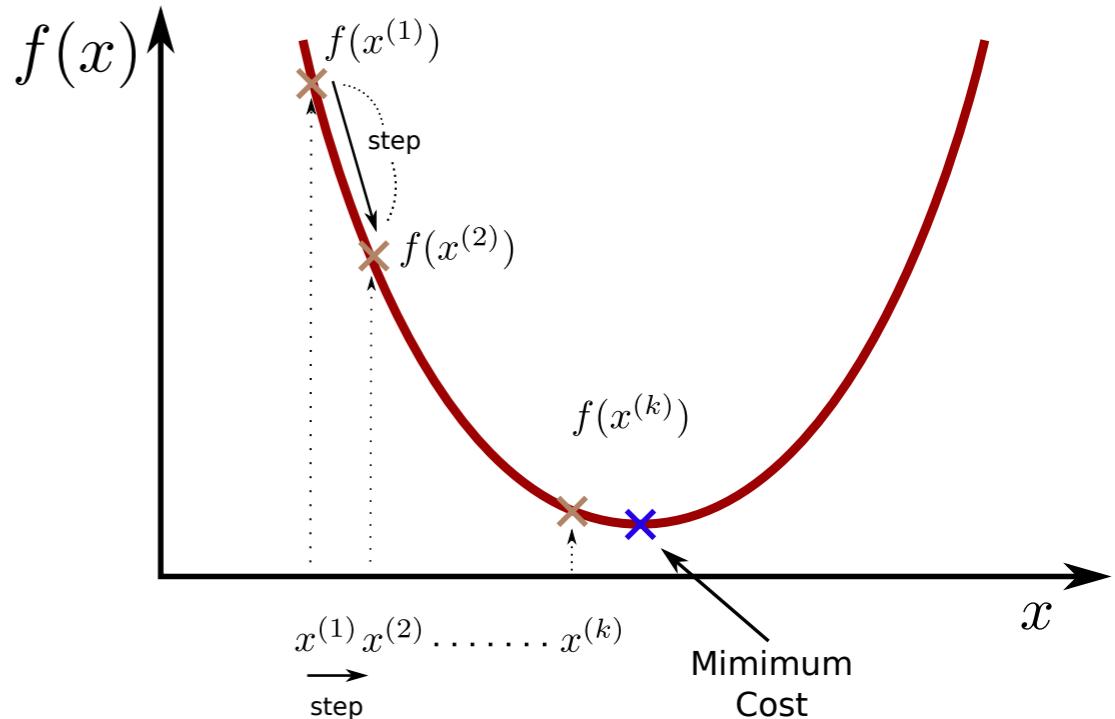
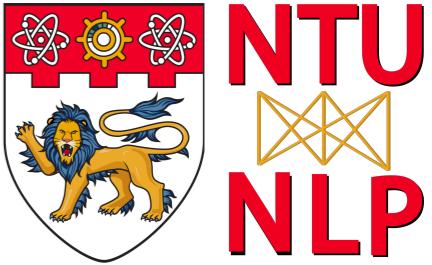


Figure: Illustration of gradient descent

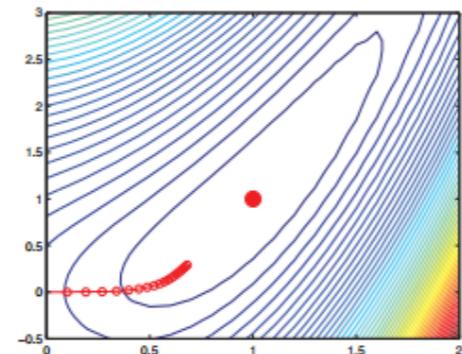
# Gradient Descent



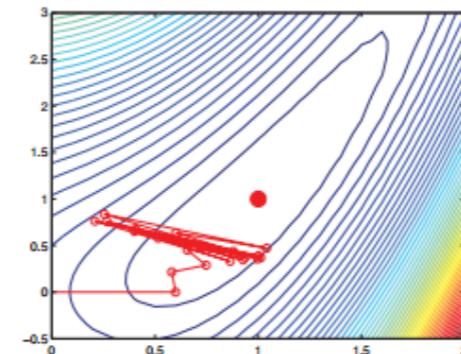
The main issue in gradient descent is:

How should we set the step size?

This turns out to be quite tricky. If we use a constant learning rate, but make it **too small**, convergence will be very slow, but if we make it **too large**, the method can fail to converge at all.

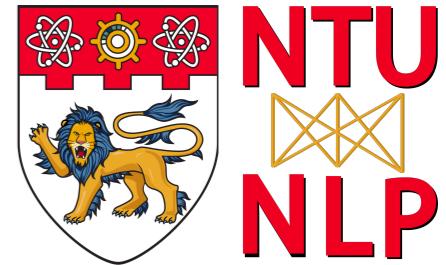


(a)



(b)

# Gradient Descent

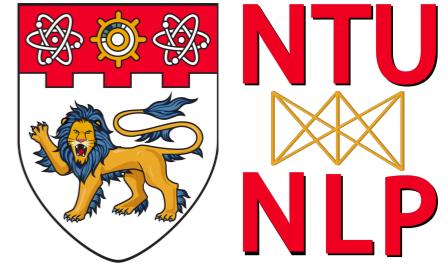


Several ways to remedy this.

- Choose the optimal step size  $\eta$  by **line-search**.
- Add a **momentum term** to the updates.
- Use methods such as **Conjugate Gradient**
- Use **second-order methods** (e.g., **Newtons method**) to exploit the **curvature of the objective function**. This requires the Hessian matrix.

**Use Adaptive Methods with Stochastic Gradient Descent**

# Online Learning



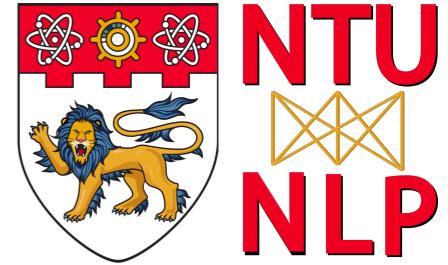
Suppose we receive an infinite stream of samples from the distribution. **One way** to optimize stochastic objectives is to perform the update at each step. This is called **stochastic gradient descent** or SGD.

If we don't have an infinite data stream, we can **simulate** one by sampling data points at random from our training set.

In theory, we should **sample with replacement**, although in practice it is usually better to **randomly permute the data and sample without replacement**, and then to repeat.

A single such pass over the entire data set is called an **epoch**.

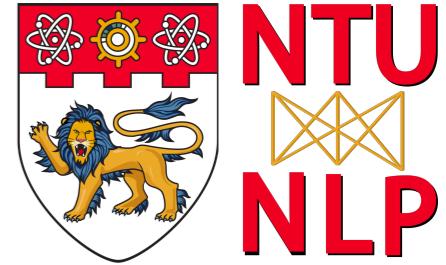
# Online Learning



In this ofine case, it is often better to compute the gradient of a mini-batch of  $B$  data cases.

If  $B = 1$ , this is standard SGD,  
and if  $B = N$ , this is standard steepest descent.  
Typically  $B \approx 100$  is used.

# Online Learning



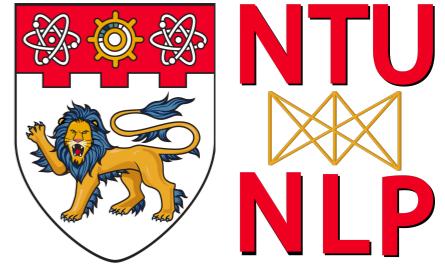
In this ofine case, it is often better to compute the gradient of a mini-batch of  $B$  data cases.

If  $B = 1$ , this is **standard SGD**,  
and if  $B = N$ , this is **standard steepest descent**.

Typically  $B \approx 100$  is used.

Although a simple first-order method, SGD performs surprisingly well on some problems, especially ones with **large data sets**.

# Online Learning



Let us consider how to compute the **MLE for linear regression** in an **online fashion**.

The online gradient at iteration  $k$  is given by

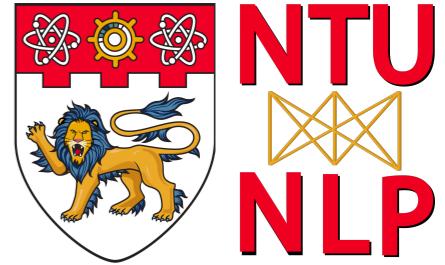
$$\mathbf{g}_k = \mathbf{x}_i(\theta_k^T \mathbf{x}_i - y_i)$$

where  $i = i(k)$  is the training example to use at iteration  $k$ .

After computing the gradient, we take a step along it as follows:

$$\theta_{k+1} = \theta_k - \eta_k (\hat{y}_k - y_k) \mathbf{x}_k$$

# Online Learning



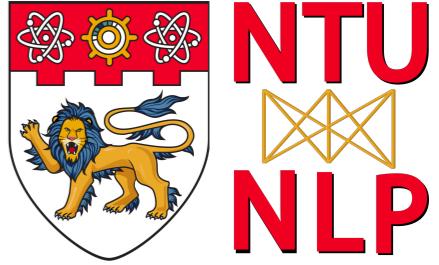
After computing the gradient, we take a step along it as follows:

$$\theta_{k+1} = \theta_k - \eta_k (\hat{y}_k - y_k) \mathbf{x}_k$$

**Interpretation:** it is the feature vector  $\mathbf{x}_k$  weighted by the difference between what we predicted,  $\hat{y}_k = \theta_k^T \mathbf{x}_k$ , and the true response,  $y_k$ ; hence the gradient acts like an **error signal**.

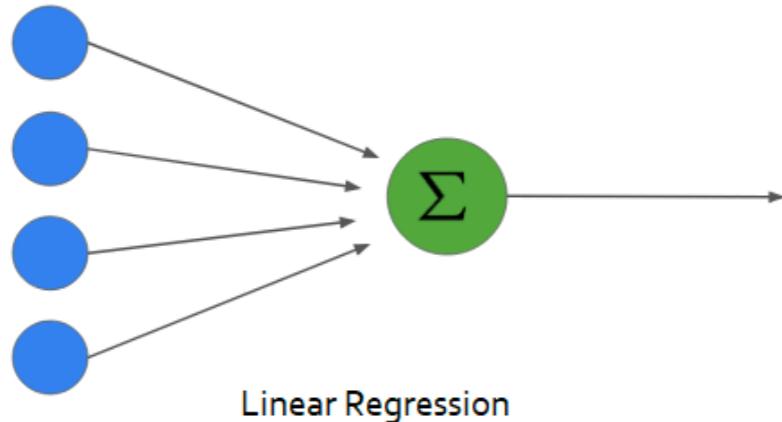
This algorithm is called the **Least Mean Squares** or LMS algorithm

# Outline of Models

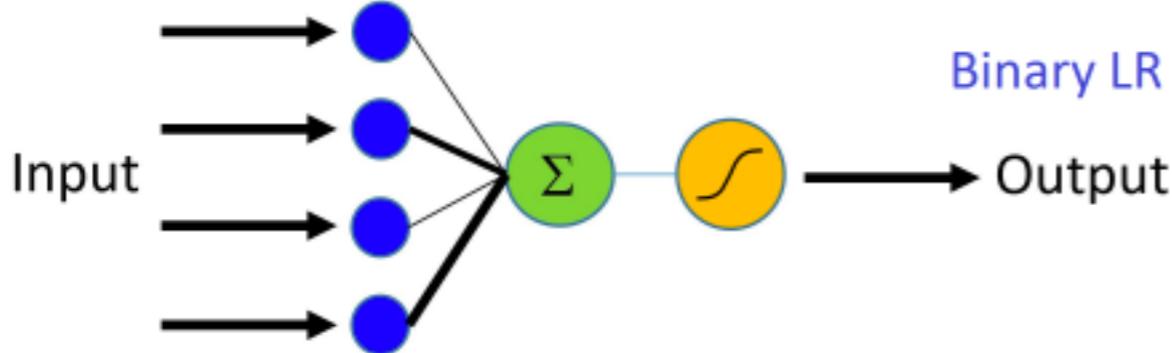


Linear Models	GD, SGD, L-BFGS
Feedforward NN/MLP	SGD + Backprop
Convolutional Nets	SGD + Backprop
Recurrent Nets	SGD+BPTT
Transformer Nets	SGD+Backprop

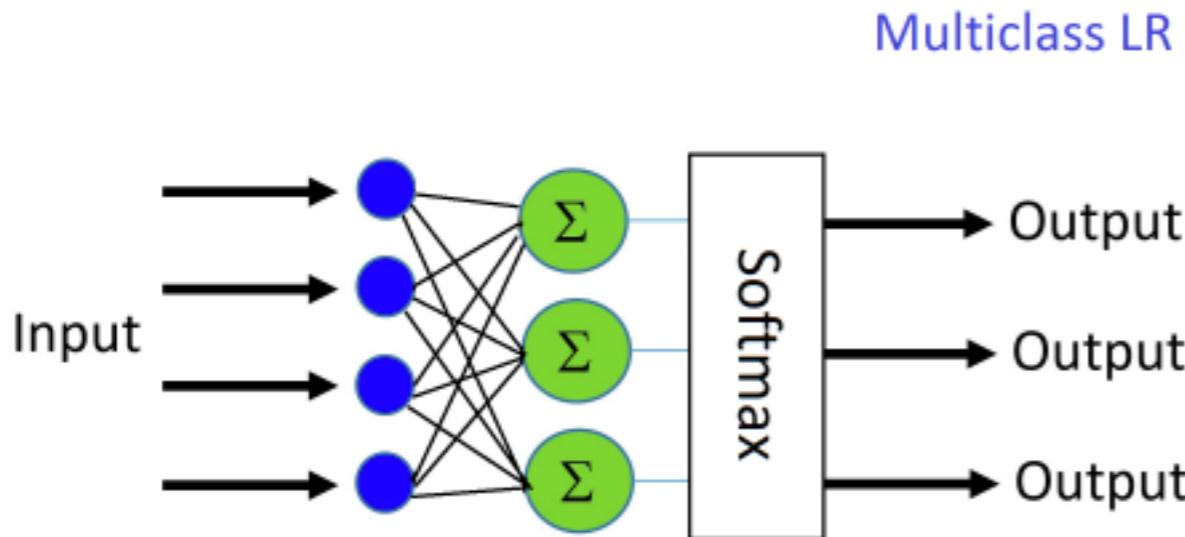
# Summary



Linear Regression



Binary LR



Multiclass LR

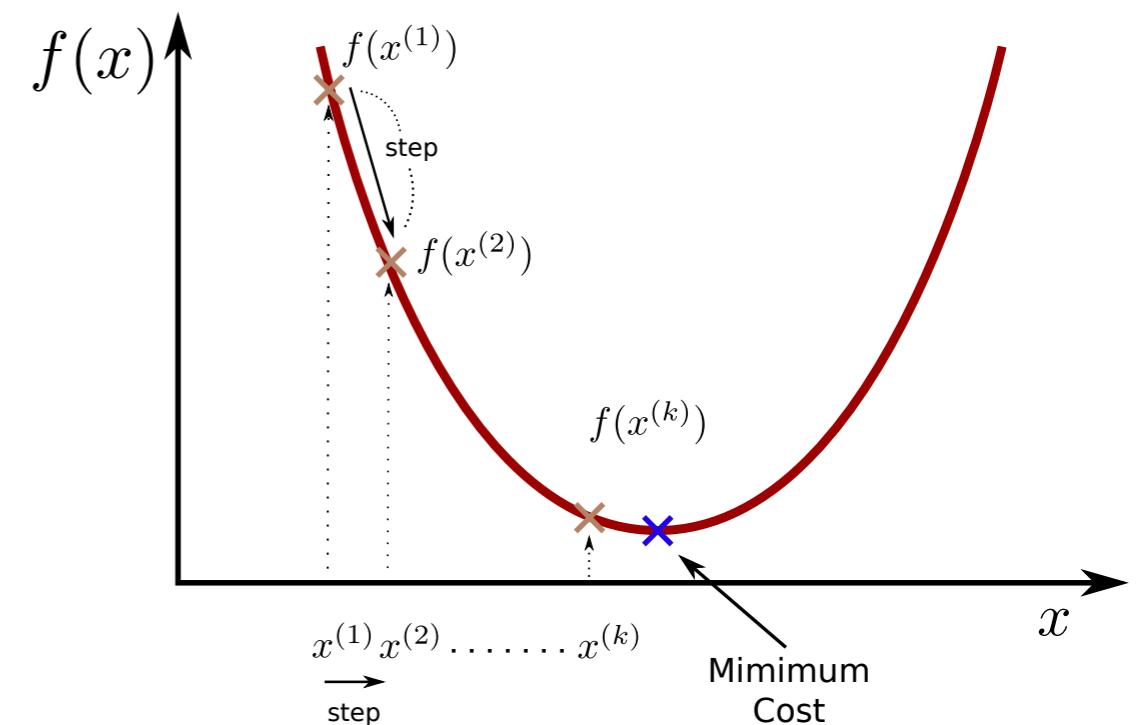


Figure: Illustration of gradient descent

$$\theta_{k+1} = \theta_k - \eta_k g_k$$