Introduction to Machine Learning

Dr Pradipta Biswas, PhD (Cantab)
Assistant Professor
Indian Institute of Science
http://cpdm.iisc.ernet.in/PBiswas.htm

Contents

- Supervised and Unsupervised Learning
- Supervised Learning
  - Decision Tree
  - Linear Regression – Gradient Descent
  - Neural Network – Backpropagation Algorithm
- Clustering
  - K-means
  - K-medoid
  - Hiearchical
  - Cluster Validation Index
- IUI Case Studies

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Classification and Prediction

- What is classification? What is regression?
- Issues regarding classification and prediction
- Classification by decision tree induction

Classification vs. Prediction

- Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Regression:
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

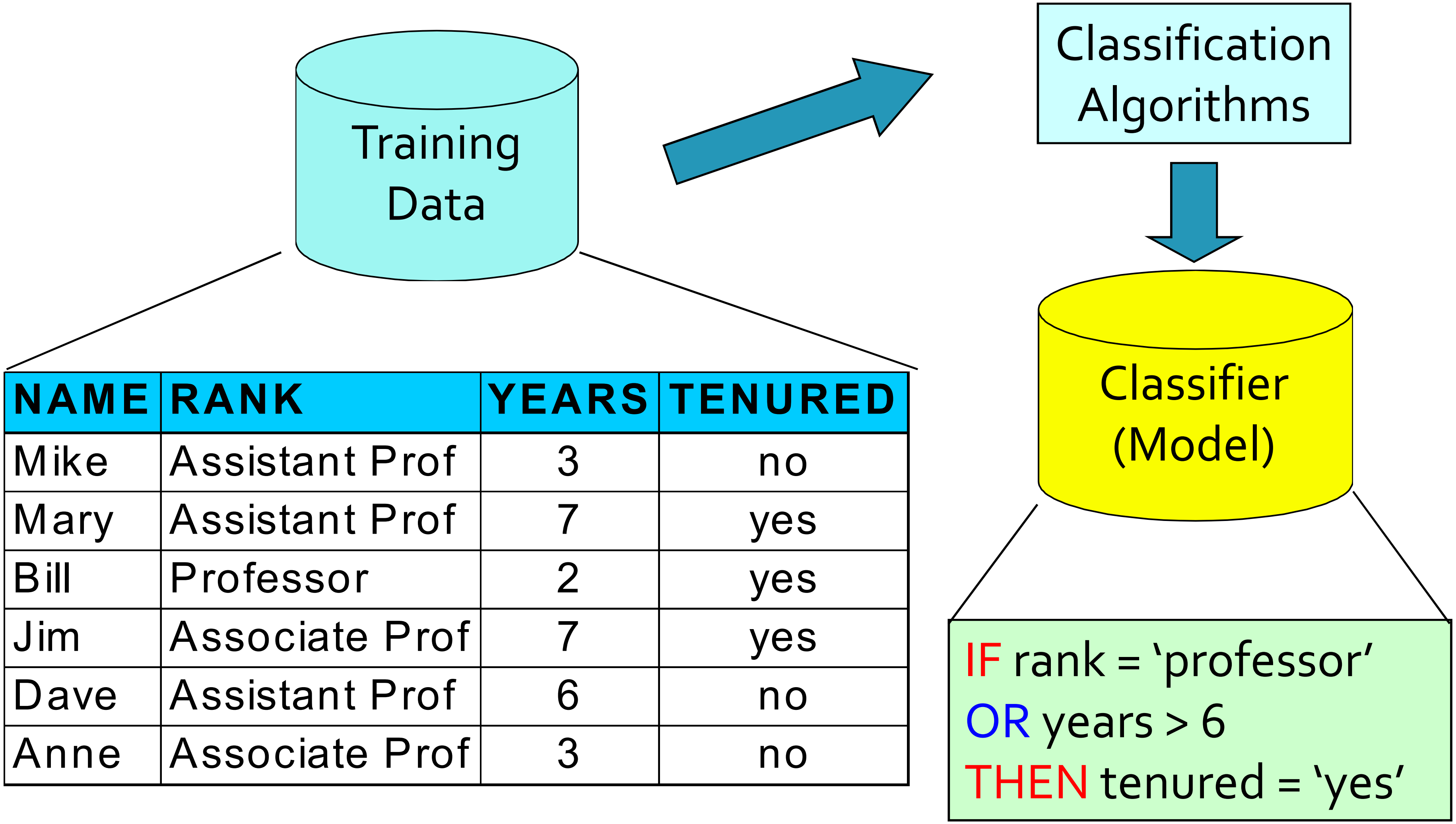
Why Classification? A motivating application

- Credit approval
  - A bank wants to classify its customers based on whether they are expected to pay back their approved loans
  - The history of past customers is used to train the classifier
  - The classifier provides rules, which identify potentially reliable future customers
  - Classification rule:
    - If age = "31...40" and income = high then credit_rating = excellent
  - Future customers
    - Paul: age = 35, income = high $\Rightarrow$ excellent credit rating
    - John: age = 20, income = medium $\Rightarrow$ fair credit rating

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test samples is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

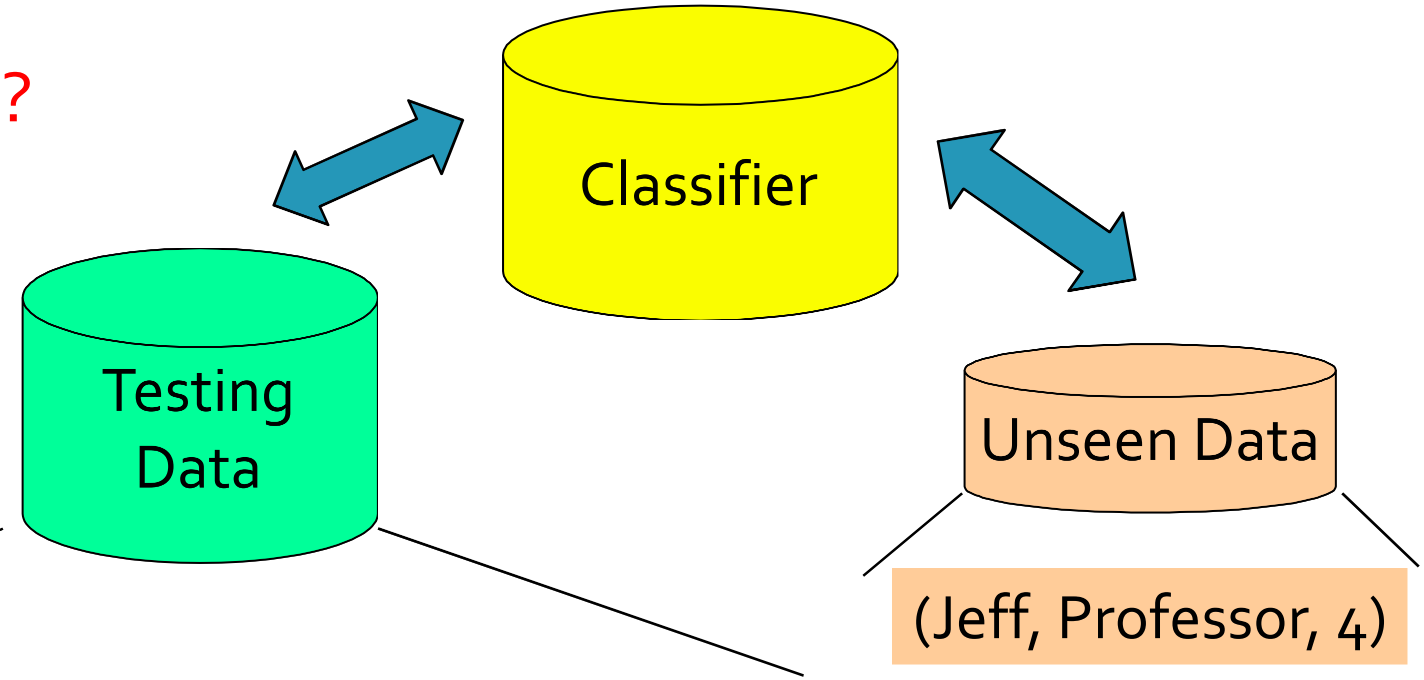
Classification Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

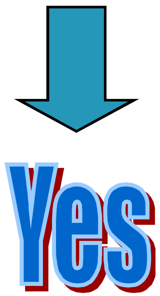
Classification Process (2): Use the Model in Prediction

Accuracy=?



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Mellisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?



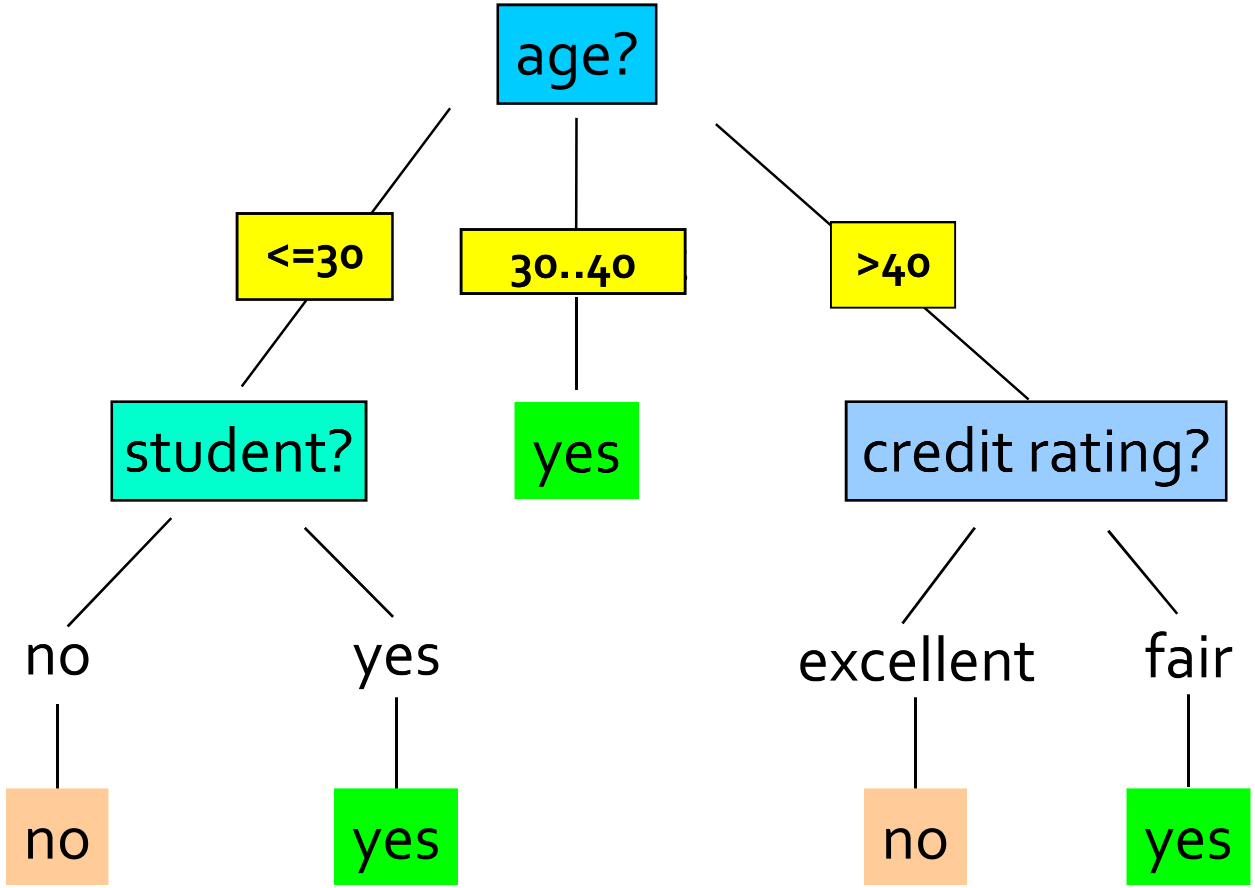
Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Training Dataset

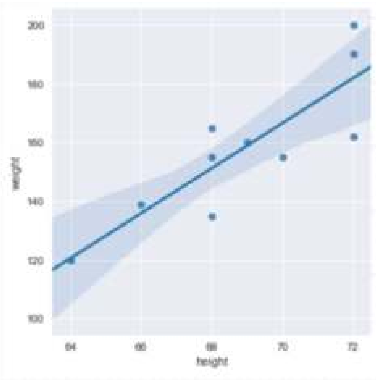
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

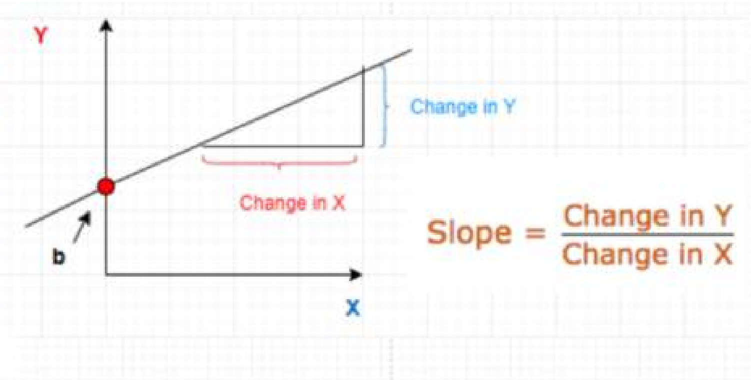
Output: A Decision Tree for "*buys_computer*"

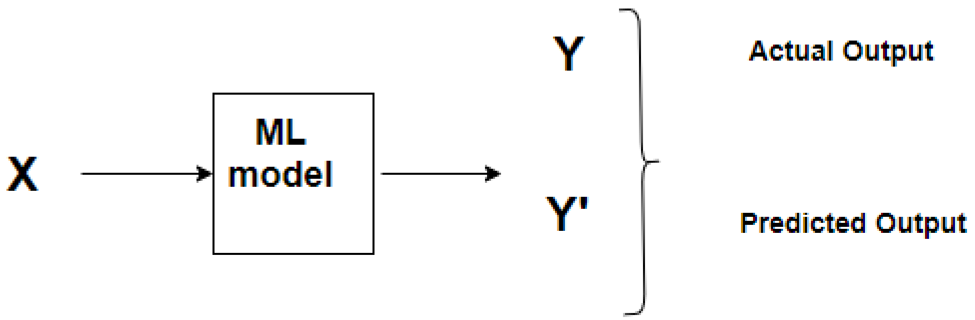


Linear Regression
Gradient Descent

- Linear regression technique
- Basic but powerful machine learning algorithm
- Fitting a straight line through a set of points





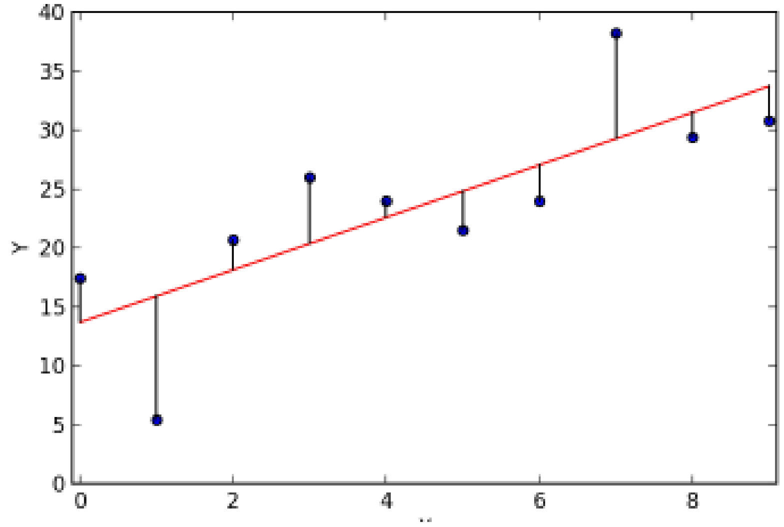


https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e

Error / Cost Function

- Cost Function/Loss Function evaluates the performance of Machine Learning Algorithm.
- Loss function computes the error for a single training example
- Cost function is the average of the loss functions for all the training examples

Error = Y'(Predicted) - Y(Actual)

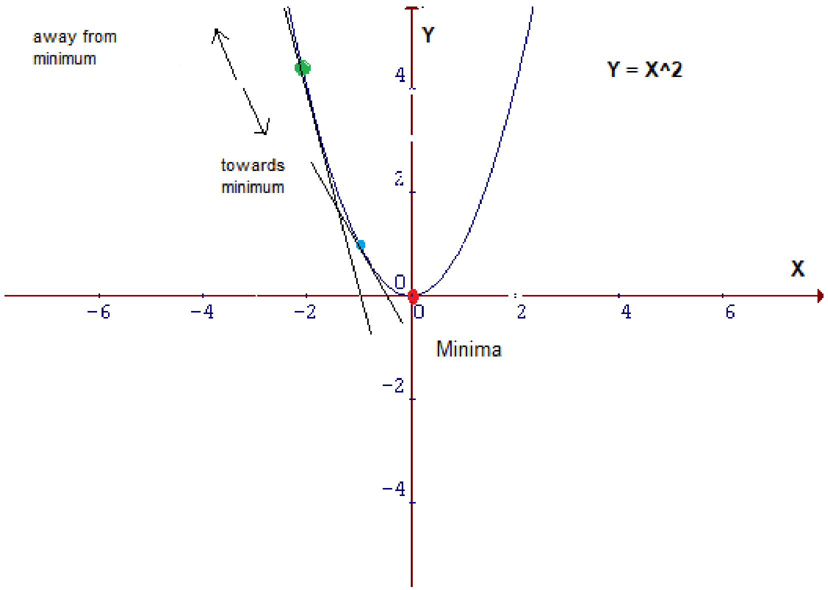


$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

Minimizing Error Function

Parameters with small changes:

$$m = m - \delta m$$
$$b = b - \delta b$$



Given Cost Function for 'N' no of samples

$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y_i' - Y_i)^2$$

Cost function is denoted by J where J is a function of m and b

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Y_i' - Y_i)^2$$

Substituting the term Y'-Y with error for simplicity

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

Derivatives

$Y = mx + c$

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

$$\frac{\partial J}{\partial m} = 2.\,Error.\frac{\partial}{\partial m}Error$$

$$\frac{\partial J}{\partial b} = 2 \cdot Error.\frac{\partial}{\partial b}Error$$

$$\frac{\partial}{\partial \mathrm{m}}\mathrm{Error} = \frac{\partial}{\partial \mathrm{m}}(\mathrm{Y'} - \mathrm{Y})$$

$$\frac{\partial}{\partial \mathrm{b}}\mathrm{Error} = \frac{\partial}{\partial \mathrm{b}}(\mathrm{Y'} - \mathrm{Y})$$

$$\frac{\partial}{\partial \mathrm{m}}\mathrm{Error} = \frac{\partial}{\partial \mathrm{m}}(\mathrm{mX} + \mathrm{b} - \mathrm{Y})$$

constants

$$\frac{\partial}{\partial \mathrm{b}}\mathrm{Error} = \frac{\partial}{\partial \mathrm{b}}(\mathrm{mX} + \mathrm{b} - \mathrm{Y})$$

constants

$$\frac{\partial}{\partial \mathrm{m}}\mathrm{Error} = \mathrm{x}$$

$$\frac{\partial}{\partial \mathrm{b}}\mathrm{Error} = 1$$

Iterations

$$\frac{\partial J}{\partial m} = 2.Error.\frac{\partial}{\partial m}Error$$

$$\frac{\partial J}{\partial b} = 2 \cdot Error.\frac{\partial}{\partial b}Error$$

$$\frac{\partial J}{\partial m} = 2.\mathrm{Error} * \mathrm{X} * \mathrm{Learning\ Rate}$$

Determines the direction to minimize the Error

Determines how large a step to take

$$\frac{\partial J}{\partial b} = 2.\mathrm{Error} * \mathrm{Learning\ Rate}$$

$$\frac{\partial J}{\partial m} = \mathrm{Error} * \mathrm{X} * \mathrm{Learning\ Rate}$$

$$\frac{\partial J}{\partial b} = \mathrm{Error} * \mathrm{Learning\ Rate}$$

$$Since\ m = m - \delta m$$

$$Since\ b = b - \delta b$$

$$m^1 = m^0 - \mathrm{Error} * \mathrm{X} * \mathrm{Learning\ Rate}$$

$$b^1 = b^0 - \mathrm{Error} * \mathrm{Learning\ Rate}$$

Neural Network

- Here x1 and x2 are normalized attribute value of data.
- y is the output of the neuron , i.e the class label.
- x1 and x2 values multiplied by weight values w1 and w2 are input to the neuron x.
- Value of x1 is multiplied by a weight w1 and values of x2 is multiplied by a weight w2.
- Given that
  - w1 = 0.5 and w2 = 0.5
  - Say value of x1 is 0.3 and value of x2 is 0.8,
  - So, weighted sum is :
  - sum= w1 x x1 + w2 x x2 = 0.5 x 0.3 + 0.5 x 0.8 = 0.55

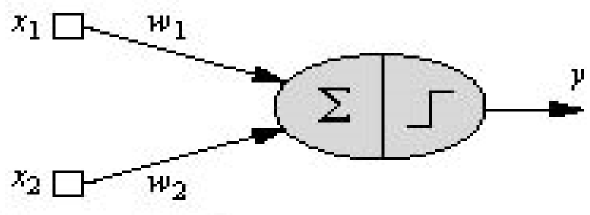


Fig1: an artificial neuron

One Neuron as a Network

- **The neuron receives the weighted sum as input and calculates the output as a function of input as follows :**

- **y = f(x) , where f(x) is defined as**

- **f(x) = 0 { when x< 0.5 }**
- **f(x) = 1 { when x >= 0.5 }**

- **For our example, x (weighted sum) is 0.55, so y = 1 ,**

- That means corresponding input attribute values are classified in class 1.

- **If for another input values , x = 0.45 , then f(x) = 0,**
- **so we could conclude that input values are classified to class 0.**

Neuron with Activation

- The neuron is the basic information processing unit of a NN. It consists of:

1 A set of links, describing the neuron inputs, with weights W_1, W_2, …, W_m

2. An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers):

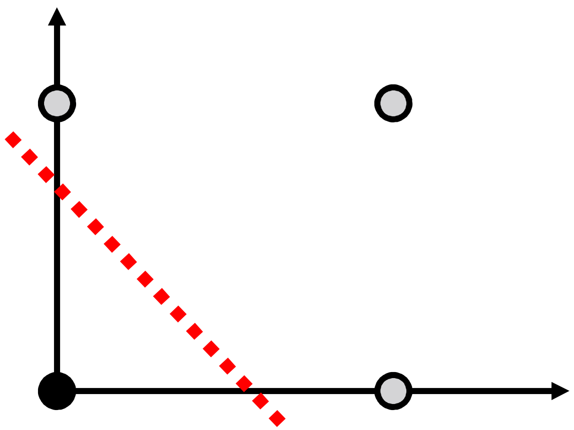
$$u = \sum_{j=1}^{m} w_j x_j$$

3 Activation function : for limiting the amplitude of the neuron output.

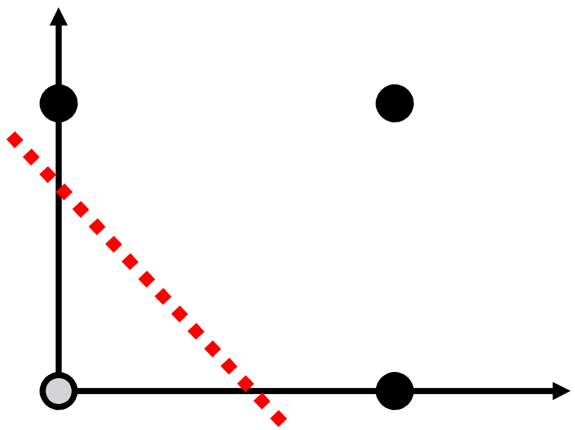
$$y = \varphi(u + b)$$

Why We Need Multi Layer ?

- Linear Separable:



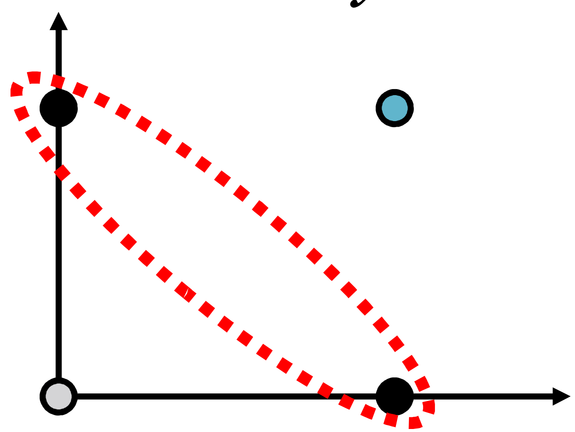
$$x \vee y$$



$$x \wedge y$$

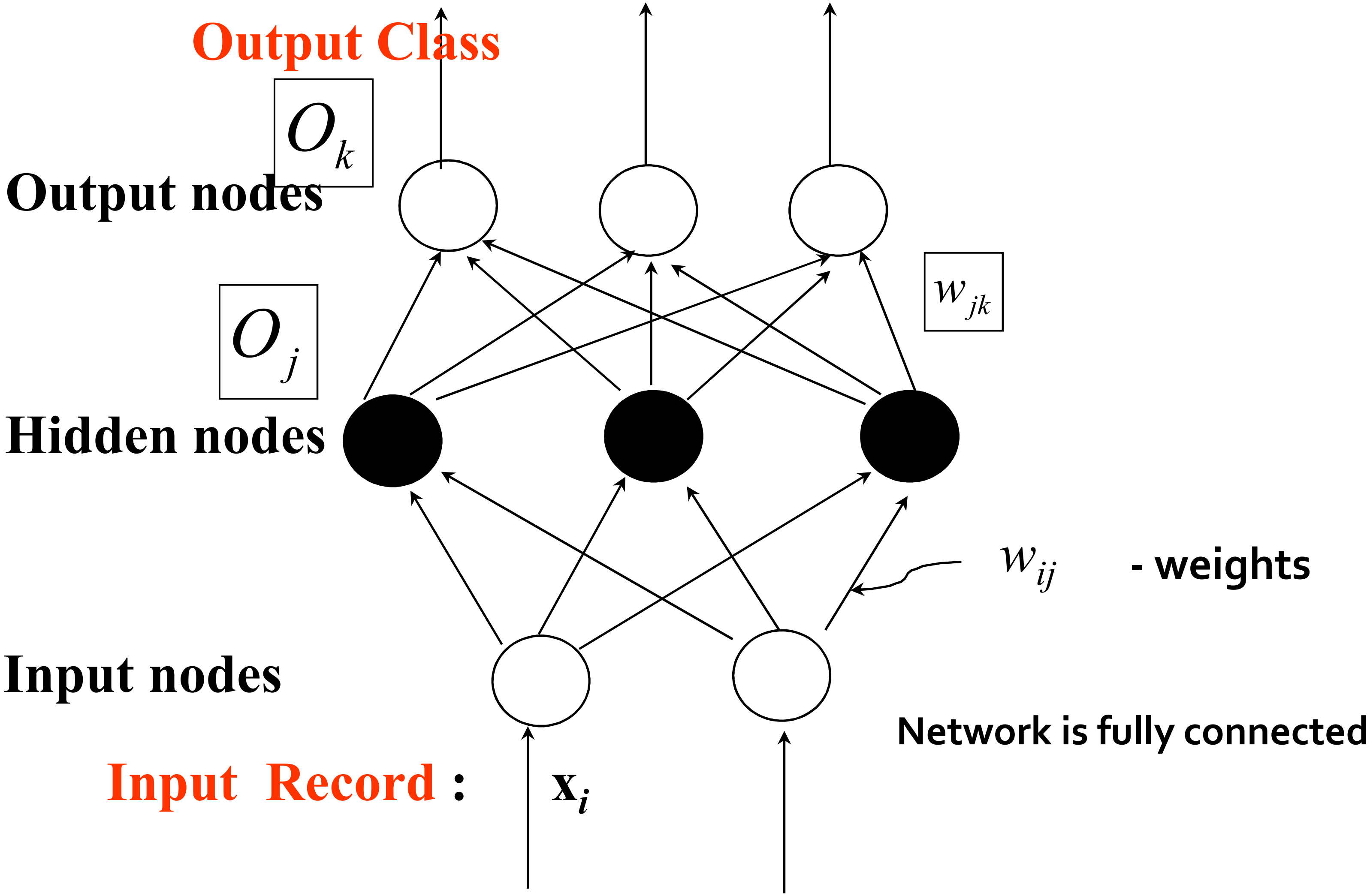
- Linear inseparable:

$$x \veebar y$$



- Solution?

A Multilayer Feed-Forward Neural Network



Neural Network Learning

- The inputs are fed simultaneously into the input layer.
- The weighted outputs of these units are fed into hidden layer.
- The weighted outputs of the last hidden layer are inputs to units making up the output layer.

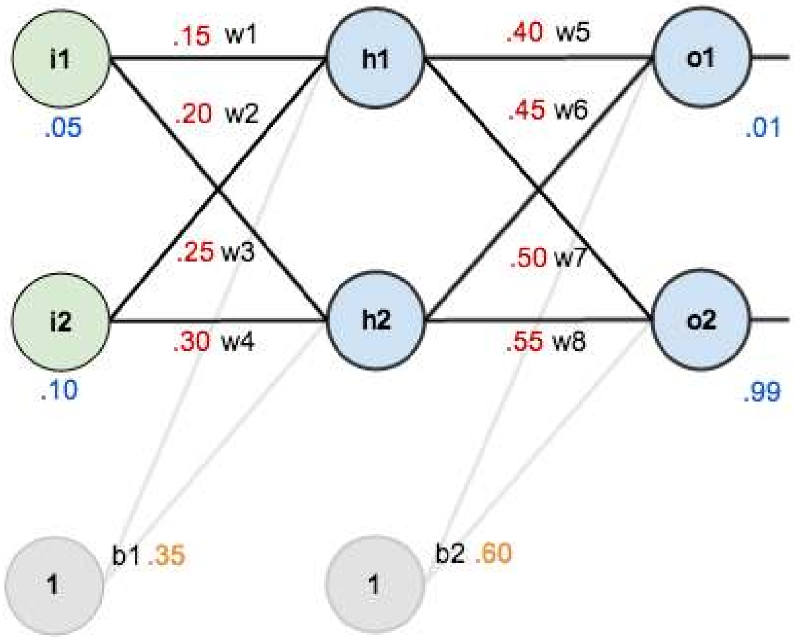
A Multilayer Feed Forward Network

- The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.
- A network containing two hidden layers is called a three-layer neural network, and so on.
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer.

Classification by Back propagation

- *Back Propagation learns by iteratively processing a set of training data (samples).*
- For each sample, weights are modified to minimize the error between network's classification and actual classification.

Backpropagation Example



$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

We then squash it using the logistic function to get the output of h_1:

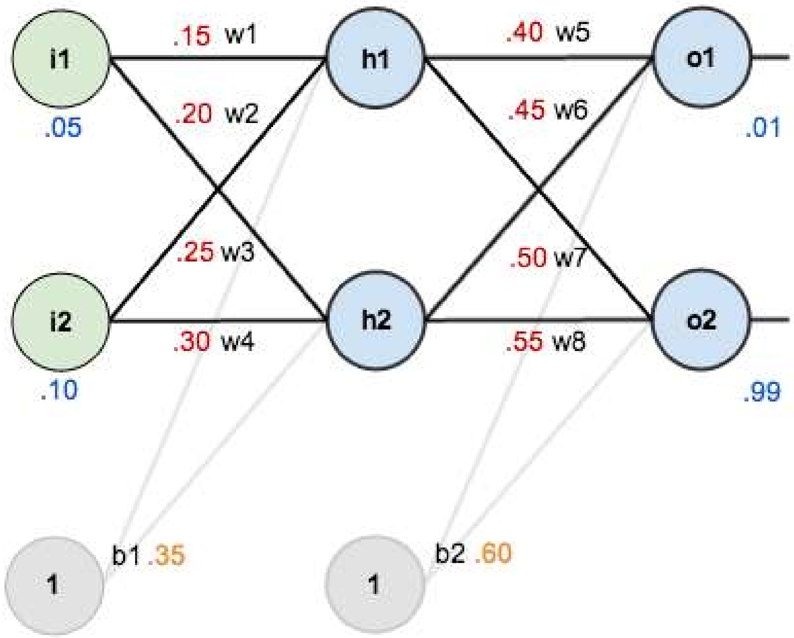
$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

Carrying out the same process for h_2 we get:

$$out_{h2} = 0.596884378$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Backpropagation Example



$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

And carrying out the same process for o_2 we get:

$$out_{o2} = 0.772928465$$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Error Calculation

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

For example, the target output for o_1 is 0.01 but the neural network output 0.75136507, therefore its error is:

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

Repeating this process for o_2 (remembering that the target is 0.99) we get:

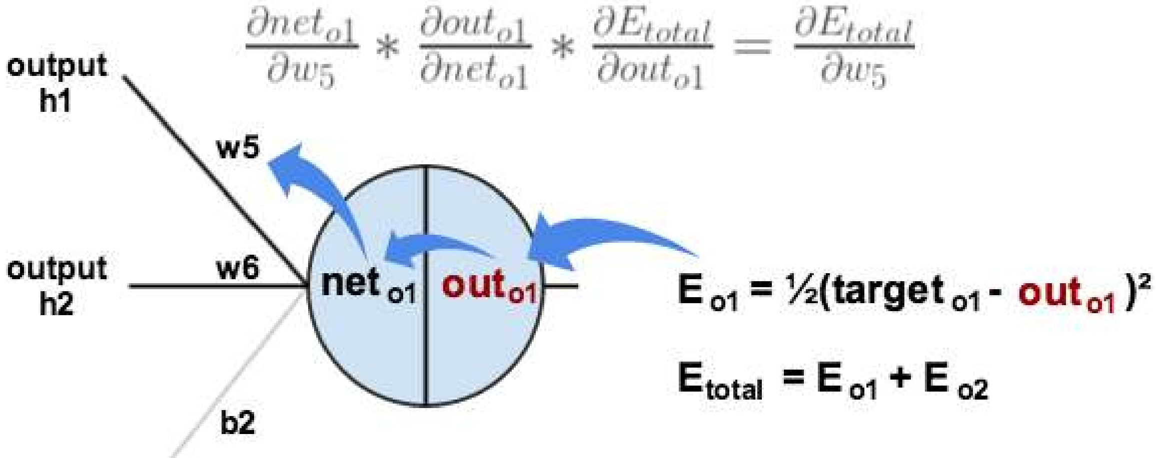
$$E_{o2} = 0.023560026$$

The total error for the neural network is the sum of these errors:

$$E_{total} = E_{o1} + E_{o2} = 0.274811083 + 0.023560026 = 0.298371109$$

Backward Pass

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$



$$E_{total} = \frac{1}{2}(target_{o1} - out_{o1})^2 + \frac{1}{2}(target_{o2} - out_{o2})^2$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = 2 * \frac{1}{2}(target_{o1} - out_{o1})^{2-1} * -1 + 0$$

$$\frac{\partial E_{total}}{\partial out_{o1}} = -(target_{o1} - out_{o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Backward Pass

$out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

$\frac{\partial out_{o1}}{\partial net_{o1}} = out_{o1}(1 - out_{o1}) = 0.75136507(1 - 0.75136507) = 0.186815602$

Finally, how much does the total net input of $o1$ change with respect to w_5?

$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$

$\frac{\partial net_{o1}}{\partial w_5} = 1 * out_{h1} * w_5^{(1-1)} + 0 + 0 = out_{h1} = 0.593269992$

Putting it all together:

$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$

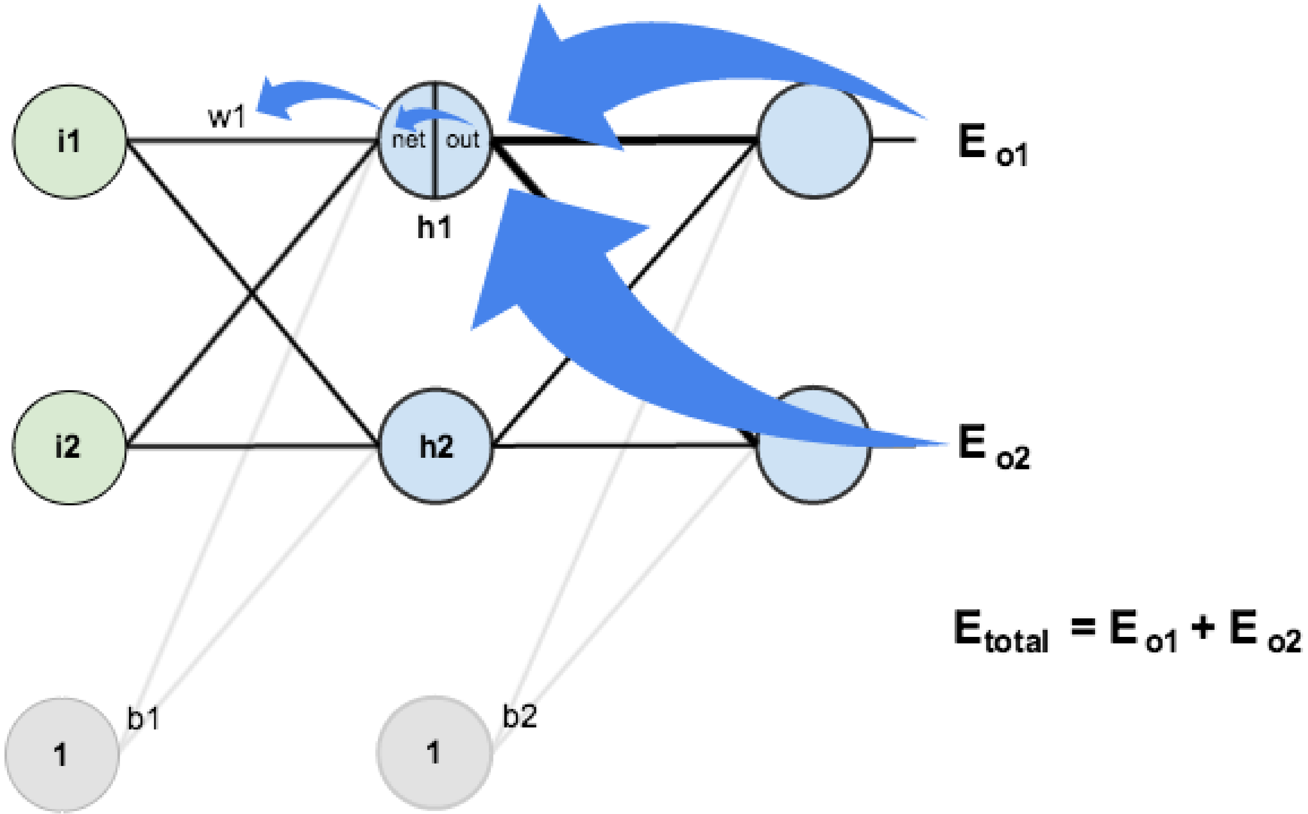
$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$

$$w_5^{+} = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

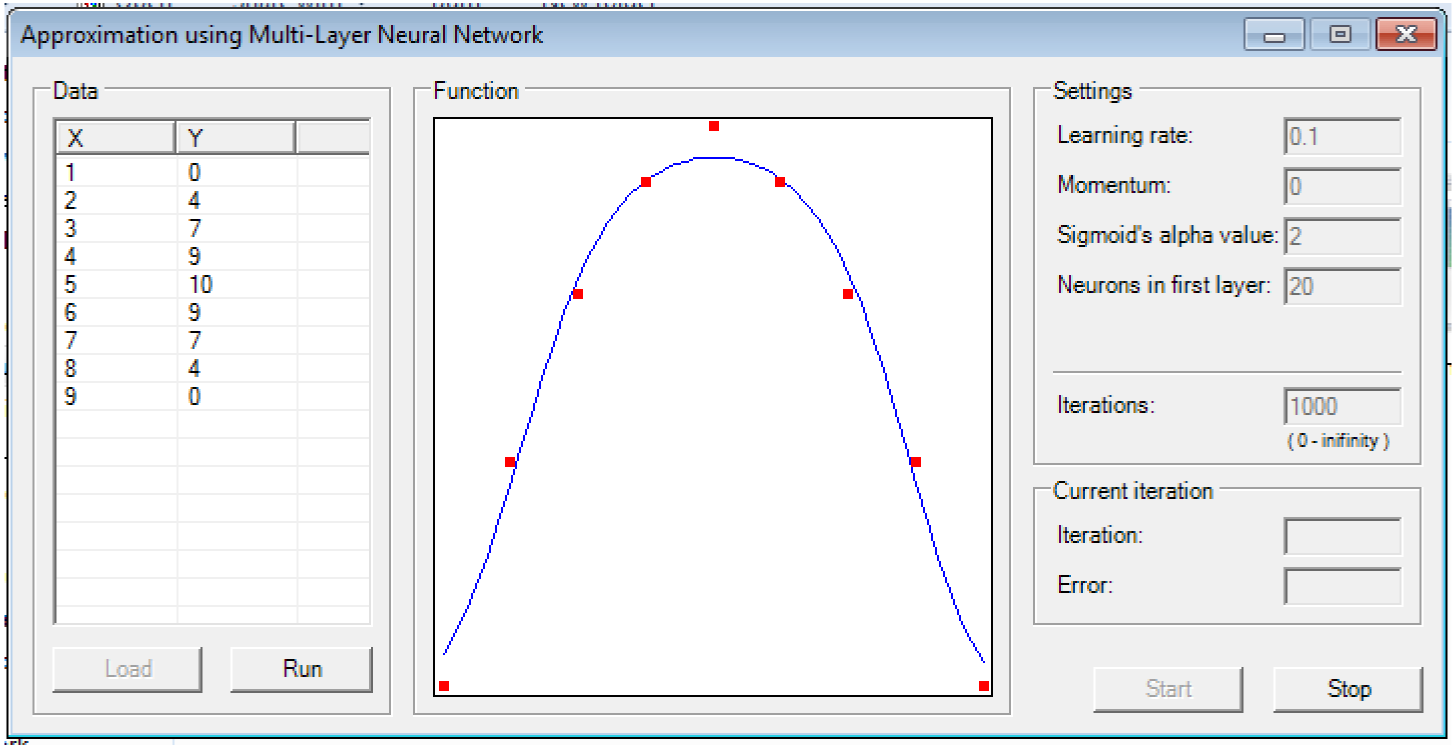
Backward Pass – Hidden Layer

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Demonstration

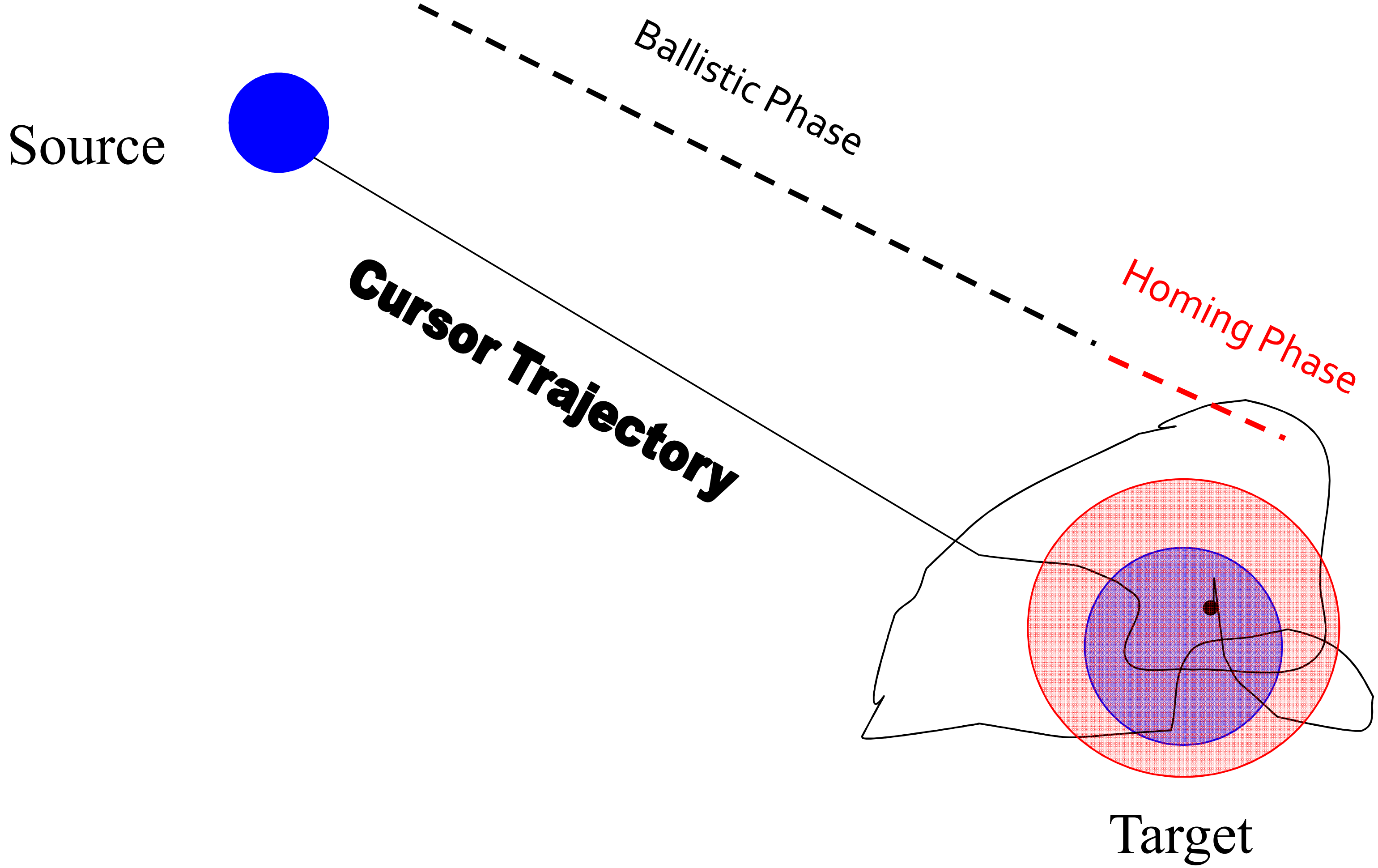


Validation

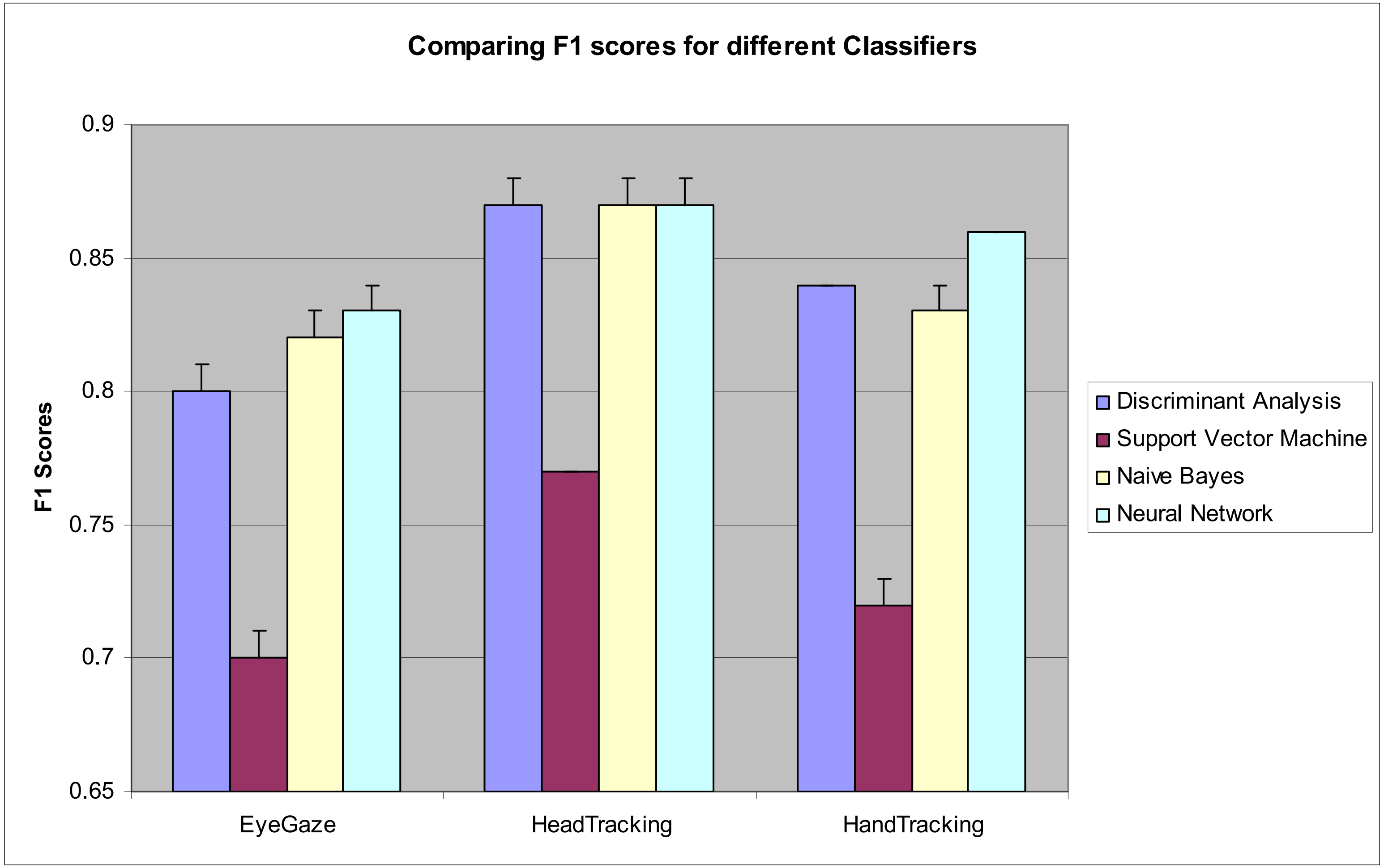
- Cross Validation (10-fold)
  - Randomly divide training data set in 10 segments
  - Train with 9 and test on remaining 1
  - Repeat the procedure 10 times
  - Training sample should be balanced
    - Nearly equal number of all possible classes
- Leave-1-out Validation: same as above, we take one sample as test set and train with the rest

Case Study – Classification for Target Prediction

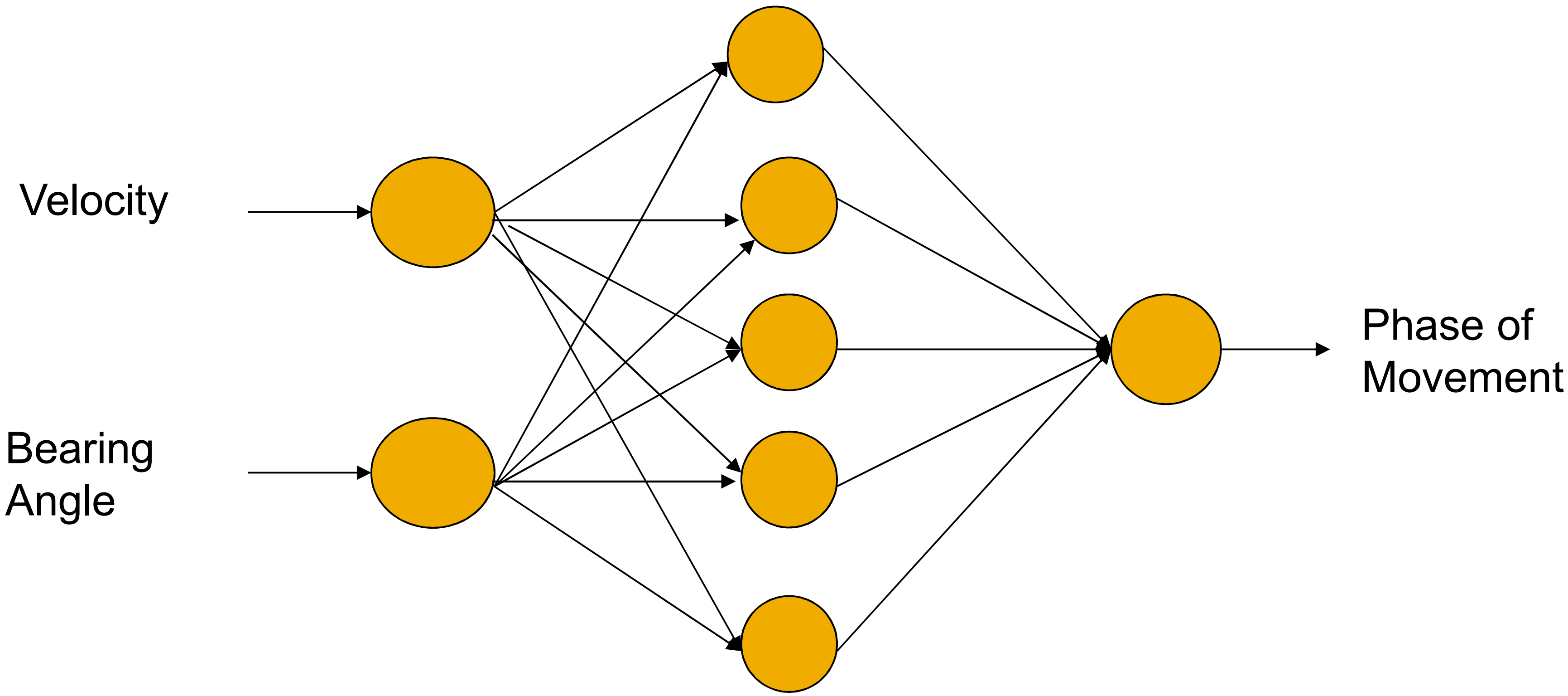
Analysing Trajectory



Classifier Result



Neural Network



Back Propagation Neural Network

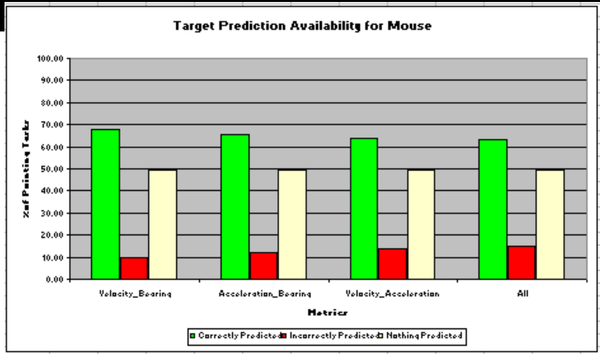
Algorithm – Neural Network

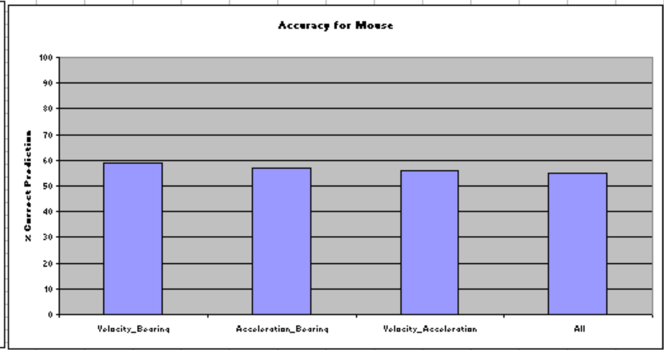
- For every change in position of pointer in screen
  - Calculate angle of movement
  - Calculate velocity of movement
  - Calculate acceleration of movement
- Run Neural Network with Angle, Velocity and Acceleration
- Check output
- If output predicts homing phase
  - Find direction of movement
  - Find nearest target from current location towards direction of movement

Evaluation Criteria

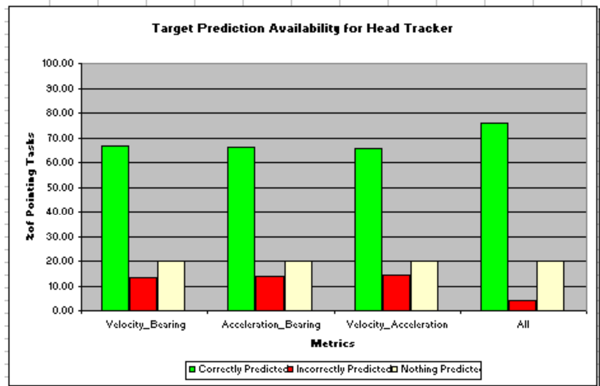
- **Availability**: In how many pointing tasks the algorithm makes a successful prediction.
- **Accuracy**: Percentage of correct prediction among all predictions
- **Sensitivity**: How quickly an algorithm can detect intended target

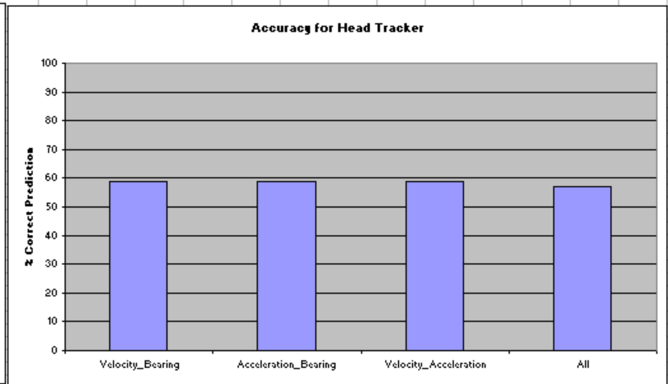
Results

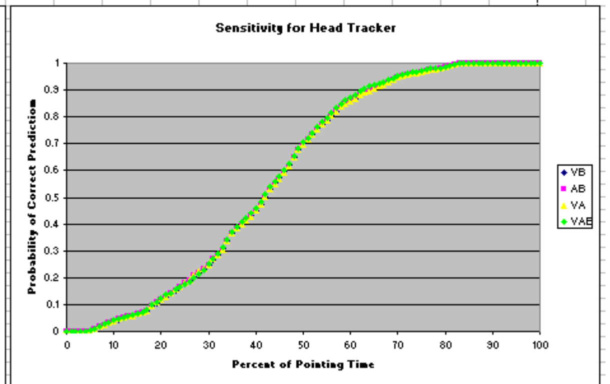

Target Prediction Availability for Mouse
%of Pointing Tasks
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All
Metrics
Correctly Predicted
Incorrectly Predicted
Nothing Predicted

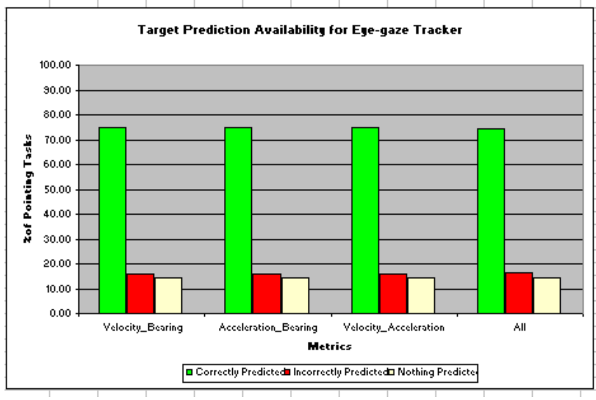

Accuracy for Mouse
% Correct Prediction
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All

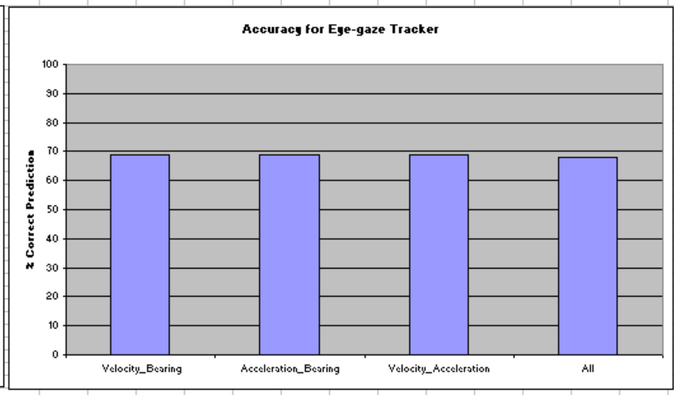

Sensitivity for Mouse
Probability of Correct Prediction
Percent of Pointing Time
VB
AB
VA
VAB

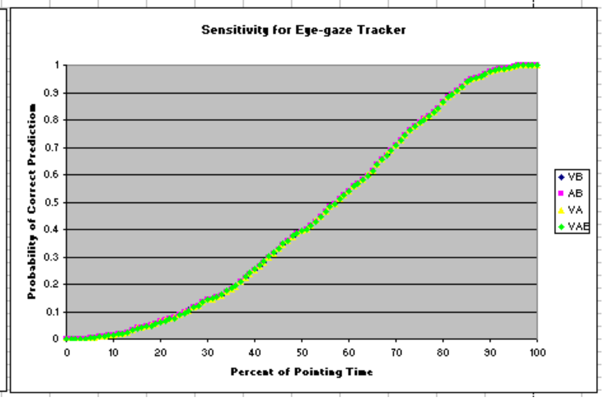

Target Prediction Availability for Head Tracker
%of Pointing Tasks
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All
Metrics
Correctly Predicted
Incorrectly Predicted
Nothing Predicted


Accuracy for Head Tracker
% Correct Prediction
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All

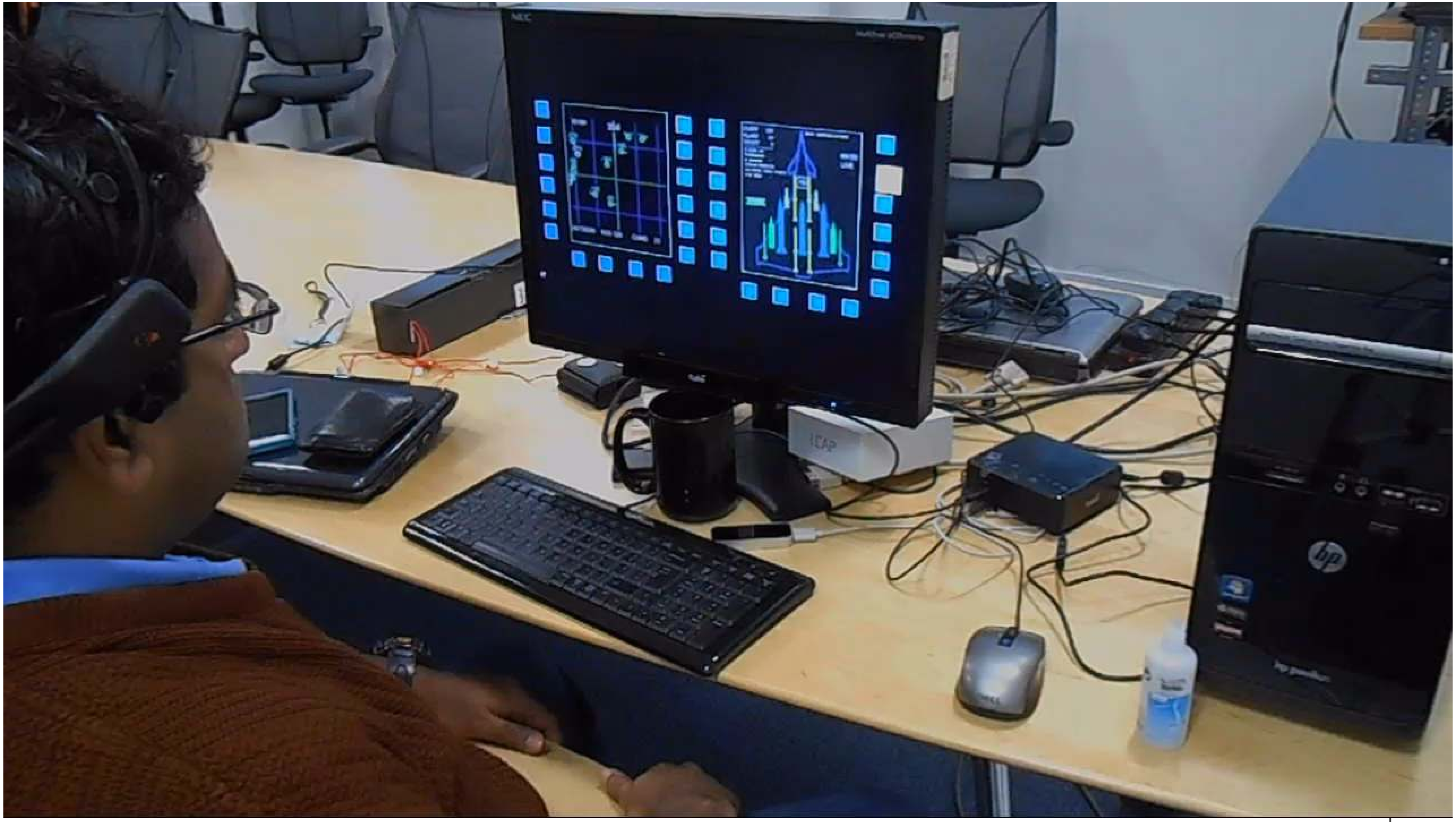

Sensitivity for Head Tracker
Probability of Correct Prediction
Percent of Pointing Time
VB
AB
VA
VAB


Target Prediction Availability for Eye-gaze Tracker
%of Pointing Tasks
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All
Metrics
Correctly Predicted
Incorrectly Predicted
Nothing Predicted


Accuracy for Eye-gaze Tracker
% Correct Prediction
Velocity_Bearing
Acceleration_Bearing
Velocity_Acceleration
All


Sensitivity for Eye-gaze Tracker
Probability of Correct Prediction
Percent of Pointing Time
VB
AB
VA
VAB

Demonstration



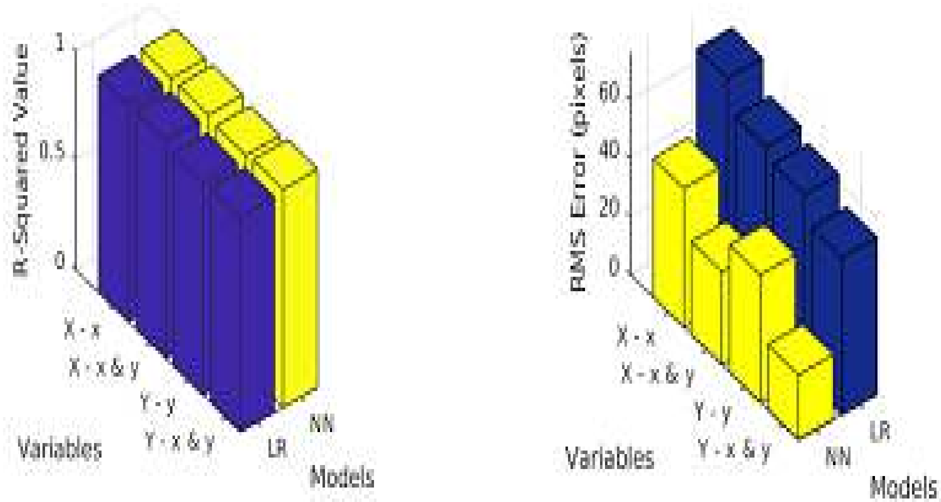
Case Study 2 – Gaze Controlled HUD / HMD

Challenge for ML

- Existing eye trackers are developed for desktop computing environment where
  - Tracker is attached below display
  - Display is a flat screen
- We used eye tracker to track eyes on windshield
- Display was away from eye tracker
- Display surface was not flat like a computer screen

Exploration

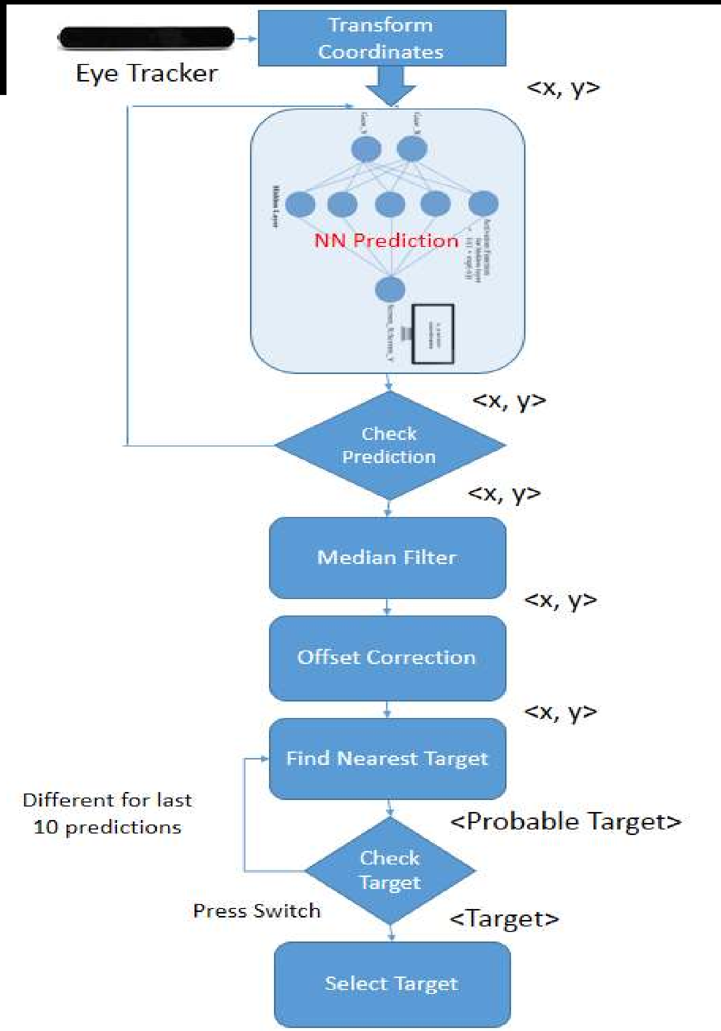
- Compared ML systems to convert eye gaze coordinates to screen coordinates on windshield
- Set up Linear Regression and Backpropagation Neural Network Models for
  - Predicting x-coordinate in screen from x coordinate recorded by gaze tracker
  - Predicting x-coordinate in screen from x and y coordinates recorded by gaze tracker
  - Predicting y-coordinate in screen from y coordinate recorded by gaze tracker
  - Predicting y-coordinate in screen from x and y coordinates recorded by gaze tracker
- Compared R^2 and RMS error
- Neural Network model worked better than Linear Regression



R^2 and RMS error for screen mounted tracker

Implementation

- Transform raw gaze coordinates geometrically for inverted image
- Run calibration program to train neural net
- Filter predicted gaze coordinates
- Correct offset based on initial calibration
- Activate target nearest to predicted gaze location

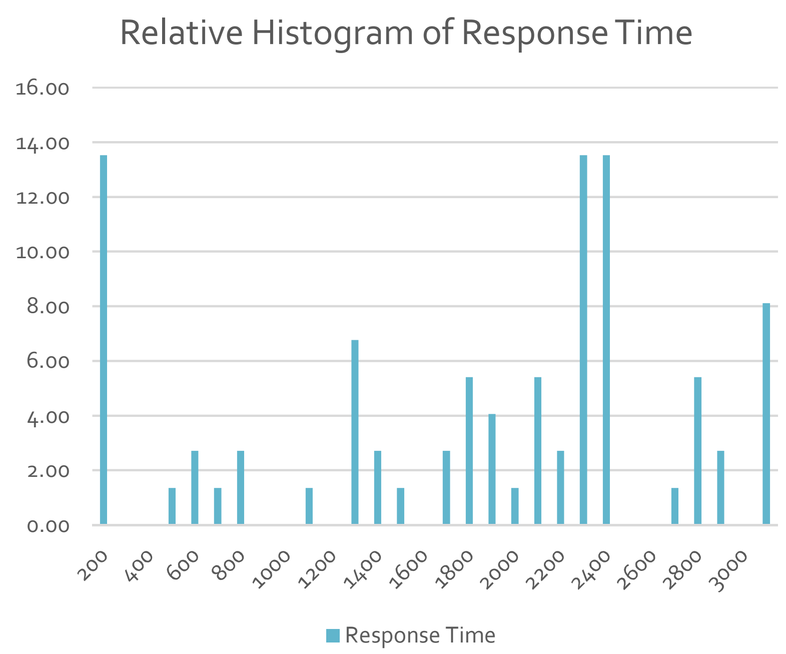


User Study

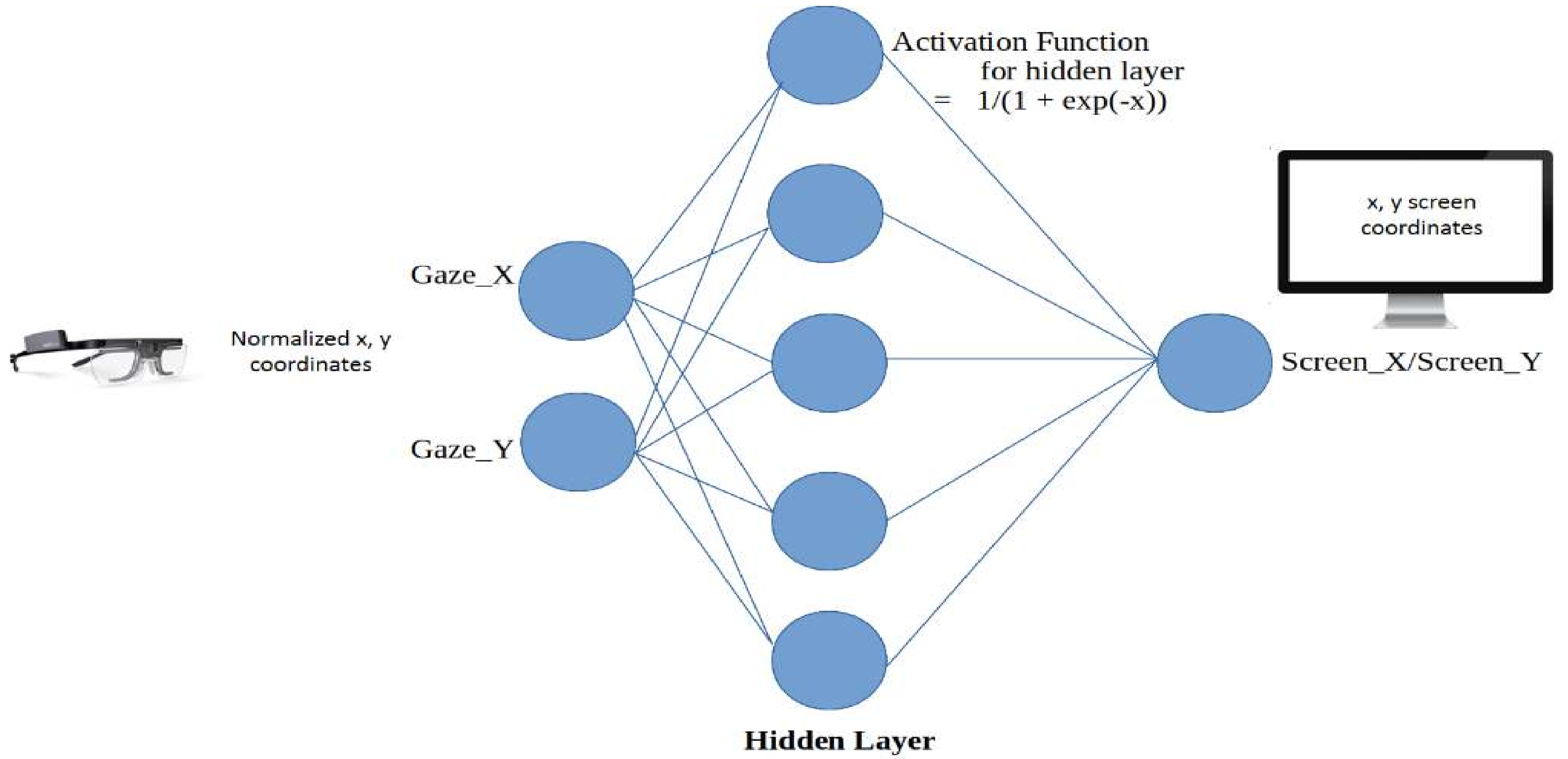
- Set up HUD in a Toyota Etios Car
- Collected data from 9 users
- Undertook standard pointing and selection task following ISO 9241 standard
- Collected 81 pointing tasks

Results

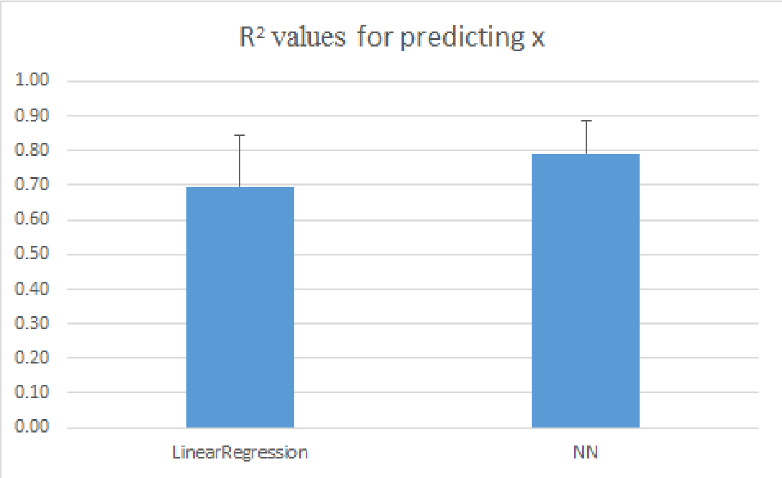
- Median pointing and selection time 2.1 secs
- Average selection time was 1.8 secs and standard deviation was 1.1 secs

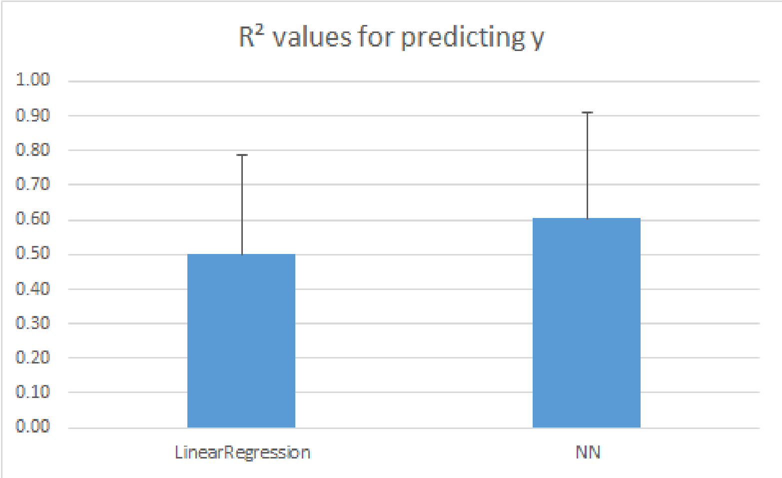


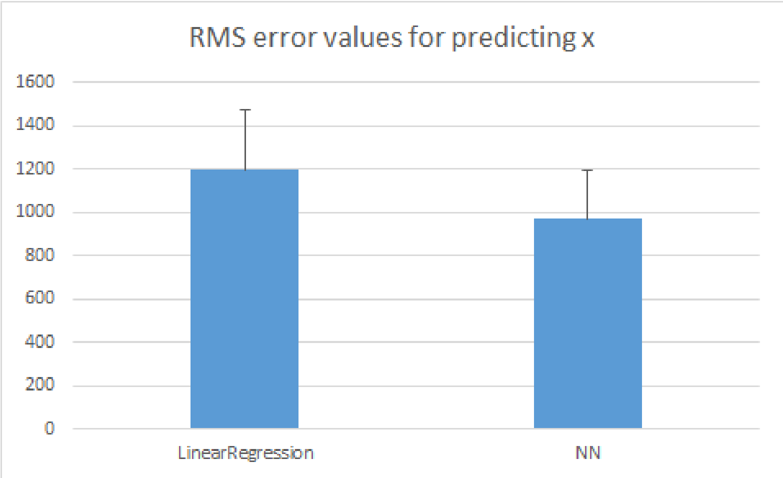
HMD

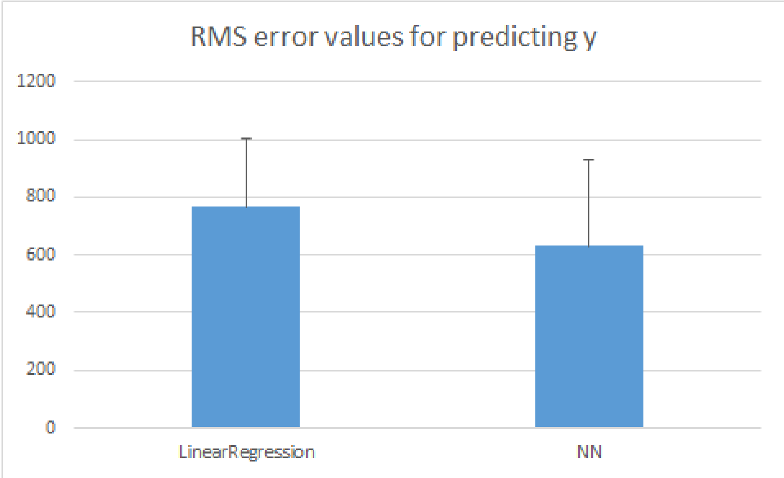

Normalized x, y coordinates
Gaze_X
Gaze_Y
Activation Function for hidden layer = 1/(1 + exp(-x))
x, y screen coordinates
Screen_X/Screen_Y
Hidden Layer

Comarison


R² values for predicting x
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
LinearRegression
NN

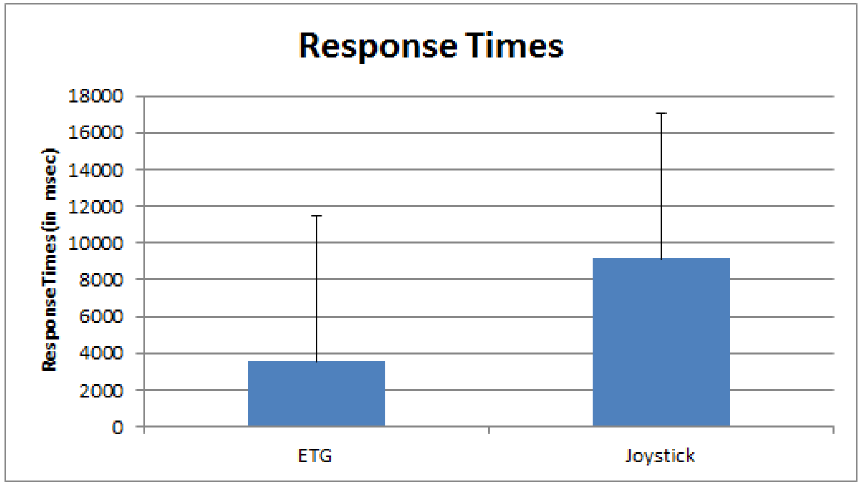

R² values for predicting y
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
LinearRegression
NN


RMS error values for predicting x
1600
1400
1200
1000
800
600
400
200
0
LinearRegression
NN

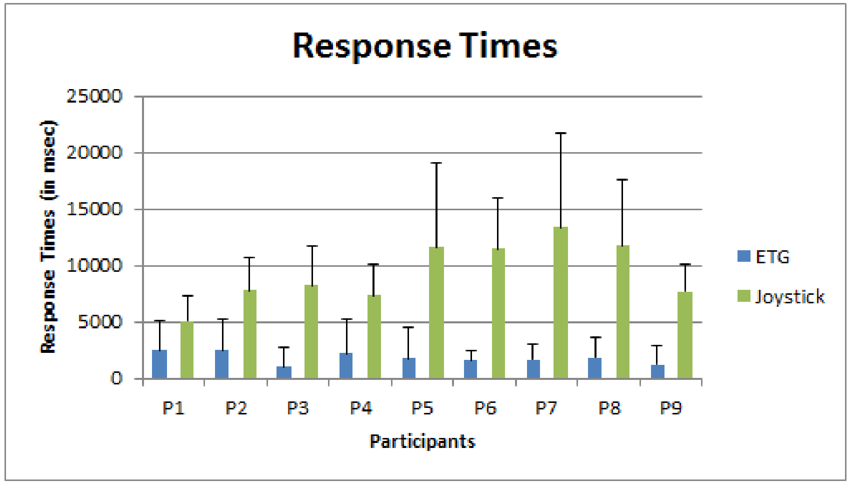

RMS error values for predicting y
1200
1000
800
600
400
200
0
LinearRegression
NN

Video

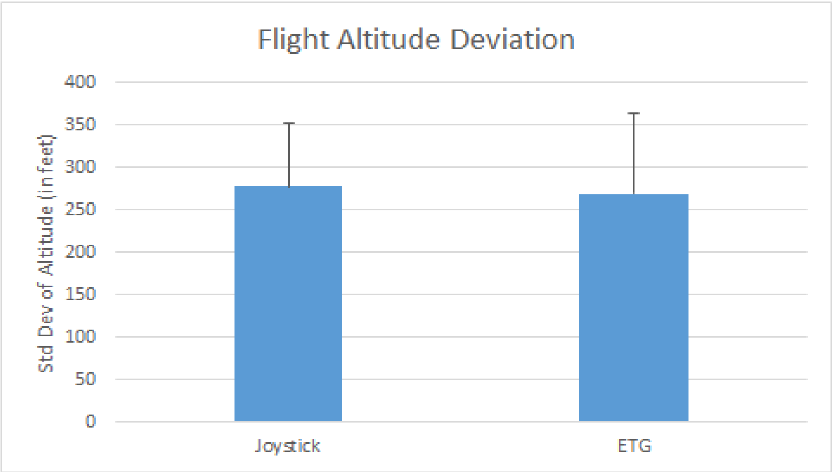
Results



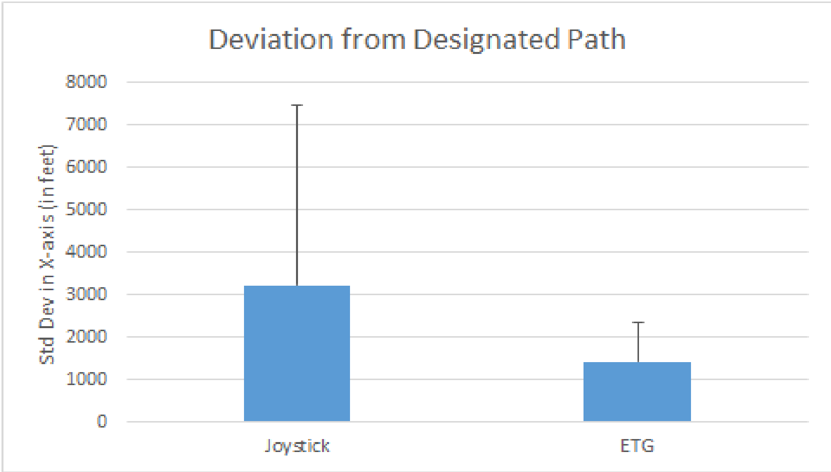
Response Times
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
ResponseTimes(in msec)
ETG
Joystick



Response Times
25000
20000
15000
10000
5000
0
Response Times (in msec)
P1
P2
P3
P4
P5
P6
P7
P8
P9
Participants
ETG
Joystick



Flight Altitude Deviation
400
350
300
250
200
150
100
50
0
Std Dev of Altitude (in feet)
Joystick
ETG

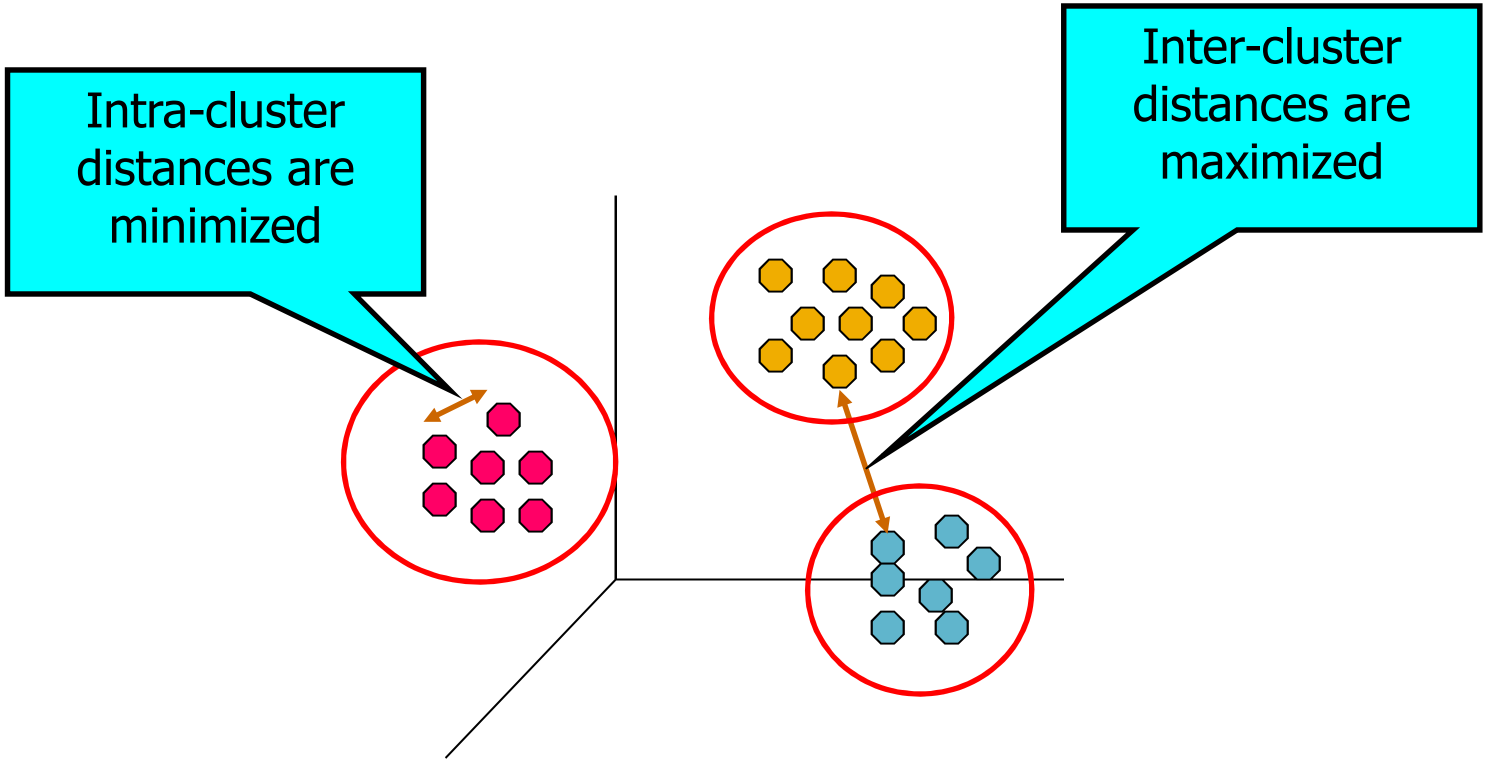


Deviation from Designated Path
8000
7000
6000
5000
4000
3000
2000
1000
0
Std Dev in X-axis (in feet)
Joystick
ETG

Cluster Analysis

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Clustering is used:
  - As a stand-alone tool to get insight into data distribution
    - Visualization of clusters may unveil important information
  - As a preprocessing step for other algorithms
    - Efficient indexing or compression often relies on clustering

Applications of Clustering

- Pattern Recognition
- Image Processing
  - cluster images based on their visual content
- Bio-informatics
- WWW and IR
  - document classification
  - cluster Weblog data to discover groups of similar access patterns

Similarity and Dissimilarity Between Objects

- Distance metrics are normally used to measure the similarity or dissimilarity between two data objects
- The most popular conform to *Minkowski distance*:

$$L_p(i,j)=\left(|x_{i1}-x_{j1}|^p+|x_{i2}-x_{j2}|^p+\ldots+|x_{in}-x_{jn}|^p\right)^{1/p}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ are two n-dimensional data objects, and p is a positive integer

- If $p = 1$, L_1 is the Manhattan (or city block) distance:

$$L_1(i,j)=|x_{i1}-x_{j1}|+|x_{i2}-x_{j2}|+\ldots+|x_{in}-x_{jn}|$$

Similarity and Dissimilarity Between Objects (Cont.)

- *If* $p = 2$, L_2 is the Euclidean distance:

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_n} - x_{j_n}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$
- Also one can use weighted distance:

$$d(i,j) = \sqrt{(w_1 |x_{i_1} - x_{j_1}|^2 + w_2 |x_{i_2} - x_{j_2}|^2 + \ldots + w_n |x_{i_n} - x_{j_n}|^2)}$$

Major Clustering Approaches

- Partitioning algorithms: Construct random partitions and then iteratively refine them by some criterion
- Hierarchical algorithms: Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Density-based: based on connectivity and density functions
- Grid-based: based on a multiple-level granularity structure
- Model-based: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

```
Select K points as the initial centroids.
repeat
    Form K clusters by assigning all points to the closest centroid.
    Recompute the centroid of each cluster.
until The centroids don't change
```

Limitations of K-means

- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-spherical shapes

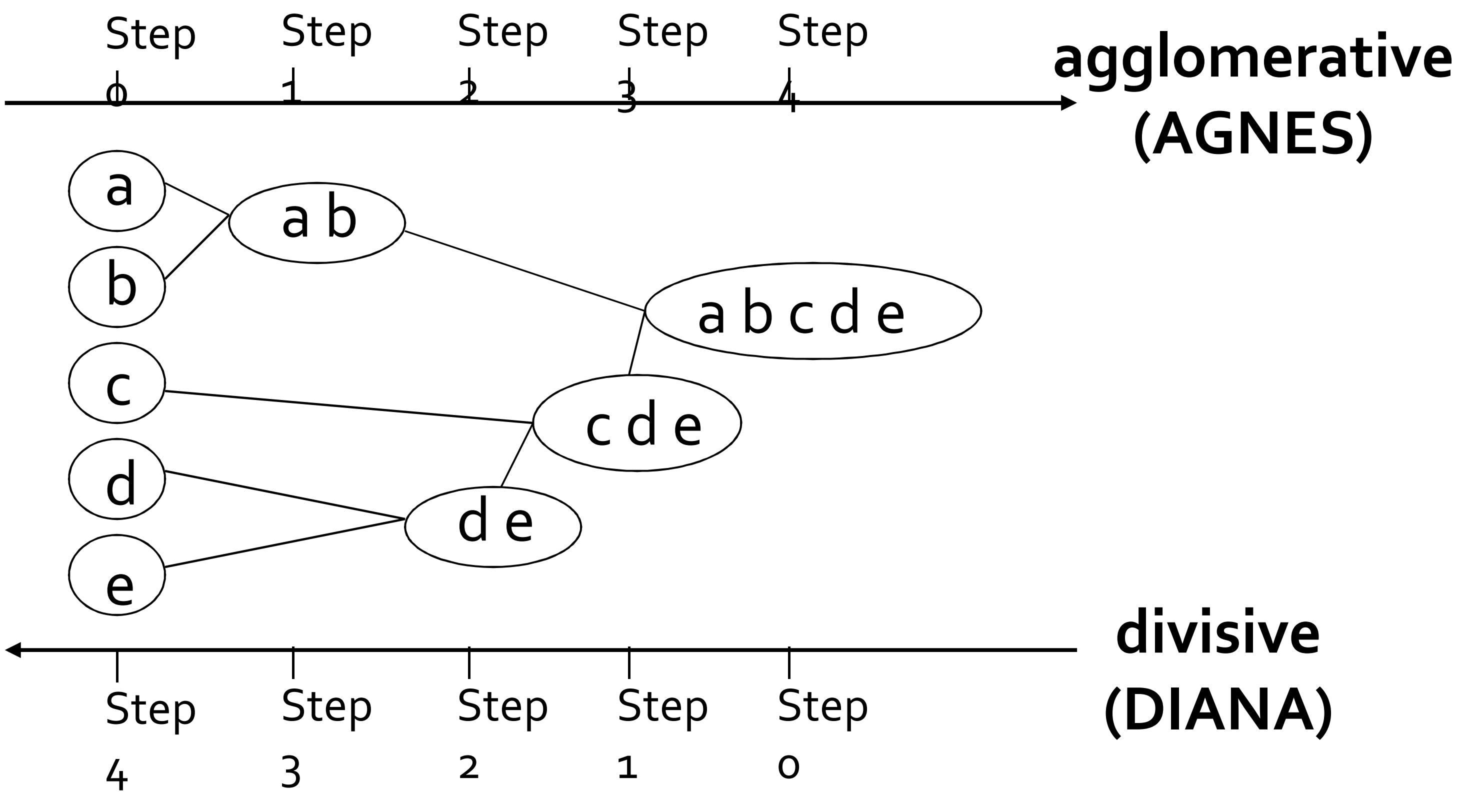
- K-means has problems when the data contains outliers. Why?

The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling

Hierarchical Clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition

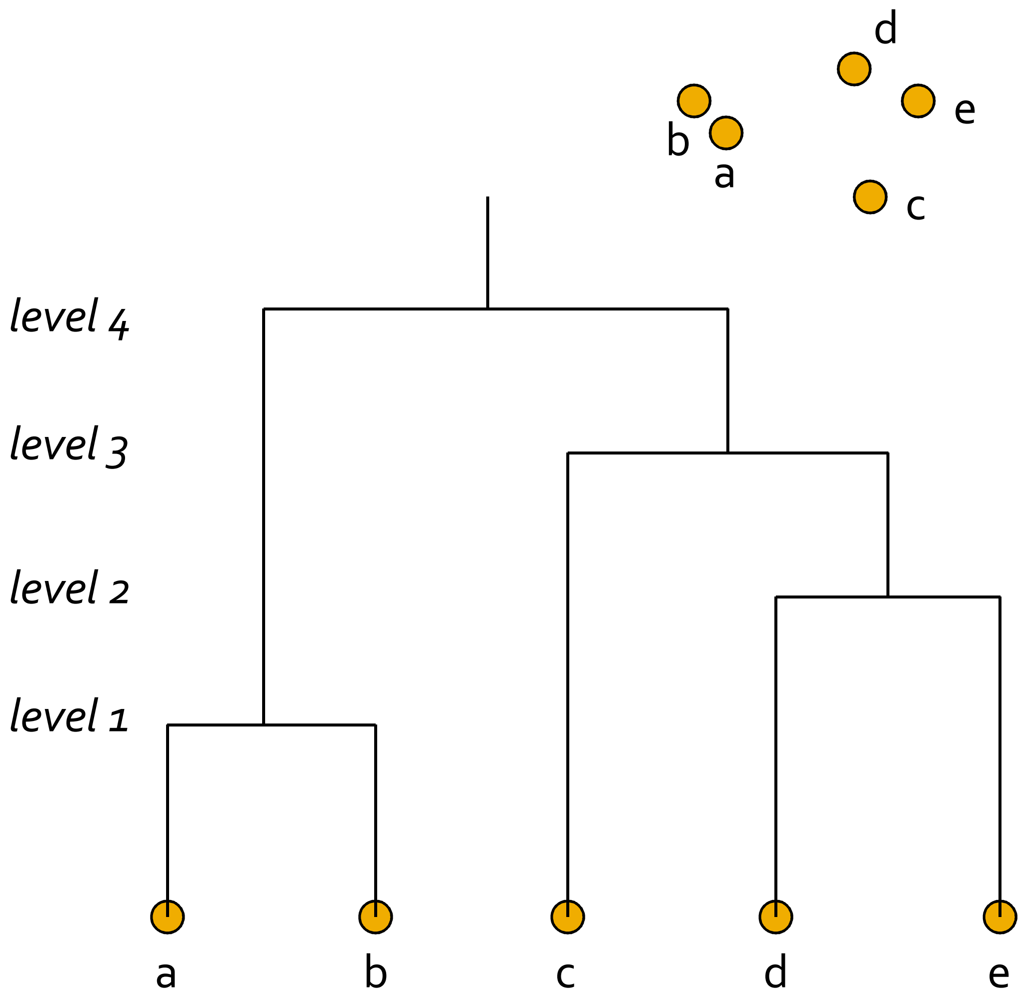


A *Dendrogram* Shows How the Clusters are Merged Hierarchically

Decompose data objects into a several levels of nested partitioning (<u>tree</u> of clusters), called a <u>dendrogram</u>.

A <u>clustering</u> of the data objects is obtained by <u>cutting</u> the dendrogram at the desired level, then each <u>connected component</u> forms a cluster.

E.g., level 1 gives 4 clusters: {a,b},{c},{d},{e},
level 2 gives 3 clusters: {a,b},{c},{d,e}
level 3 gives 2 clusters: {a,b},{c,d,e}, etc.



Soft Clustering

- What happens when we can not specify the optimum number of clusters beforehand
- Can we find the optimum number of clusters?
- Two methods can return overlapping clusters
  - Fuzzy c-means
  - EM Clustering algorithm

Fuzzy c-means

- Place a set of cluster centres
- Assign a fuzzy membership to each data point depending on distance
- Compute the new centre of each class
- Termination is based on an objective function
- Returns cluster centres and membership values of each data point to each cluster

EM Algorithm

- Assume data came from a set of Gaussian Distribution
- Assign data points to distributions and find Expected probability
- Update mean and std dev of distributions to Maximize probabilities

Expectation Maximization

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

1) E Step : Evaluate responsibilities using the current parameters values

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

2) M Step : Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

3) Evaluate the log likelihood

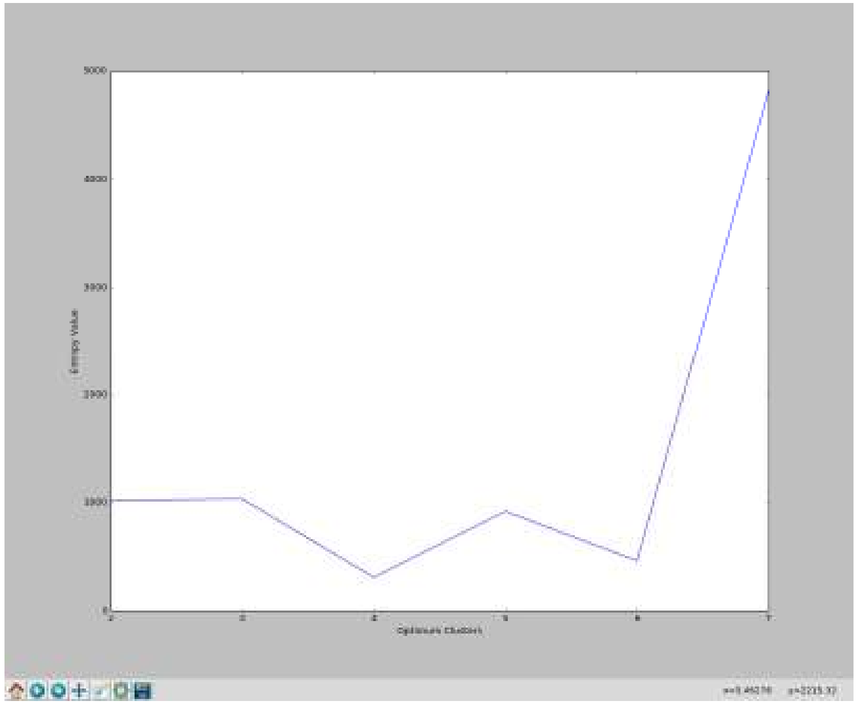
$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

XB Indexing

- Cluster validity indexes are used to evaluate the fitness of partitions produced by clustering algorithms
- Entropy values are also used to evaluate the fitness of partitions
- XB indexing is one type of validity function proposed by Xie and Beni
- Ratio between compactness measure and separation measure



Summary

- Classification and Clustering
  - Decision tree and neural network for classification
  - Linear Regression
  - Cross validation
  - Hierarchical & K-means clustering
  - Soft Clustering
  - Cluster Validation Index
  - Case studies on IUI