FDP on Data Analytics in Healthcare

Traditional Learning Algorithms for Healthcare

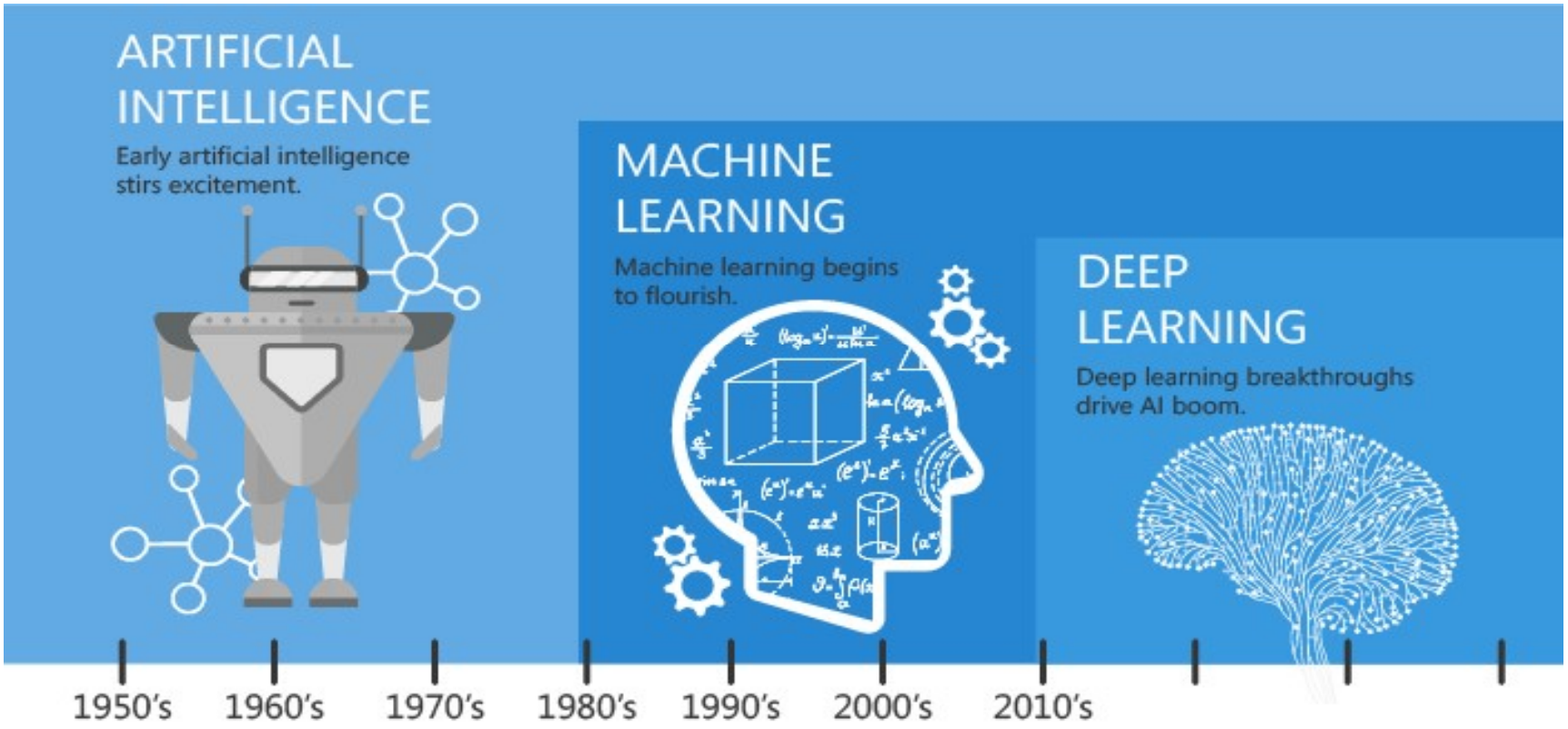
Presented by

Dr. S. Kavitha

Associate Professor

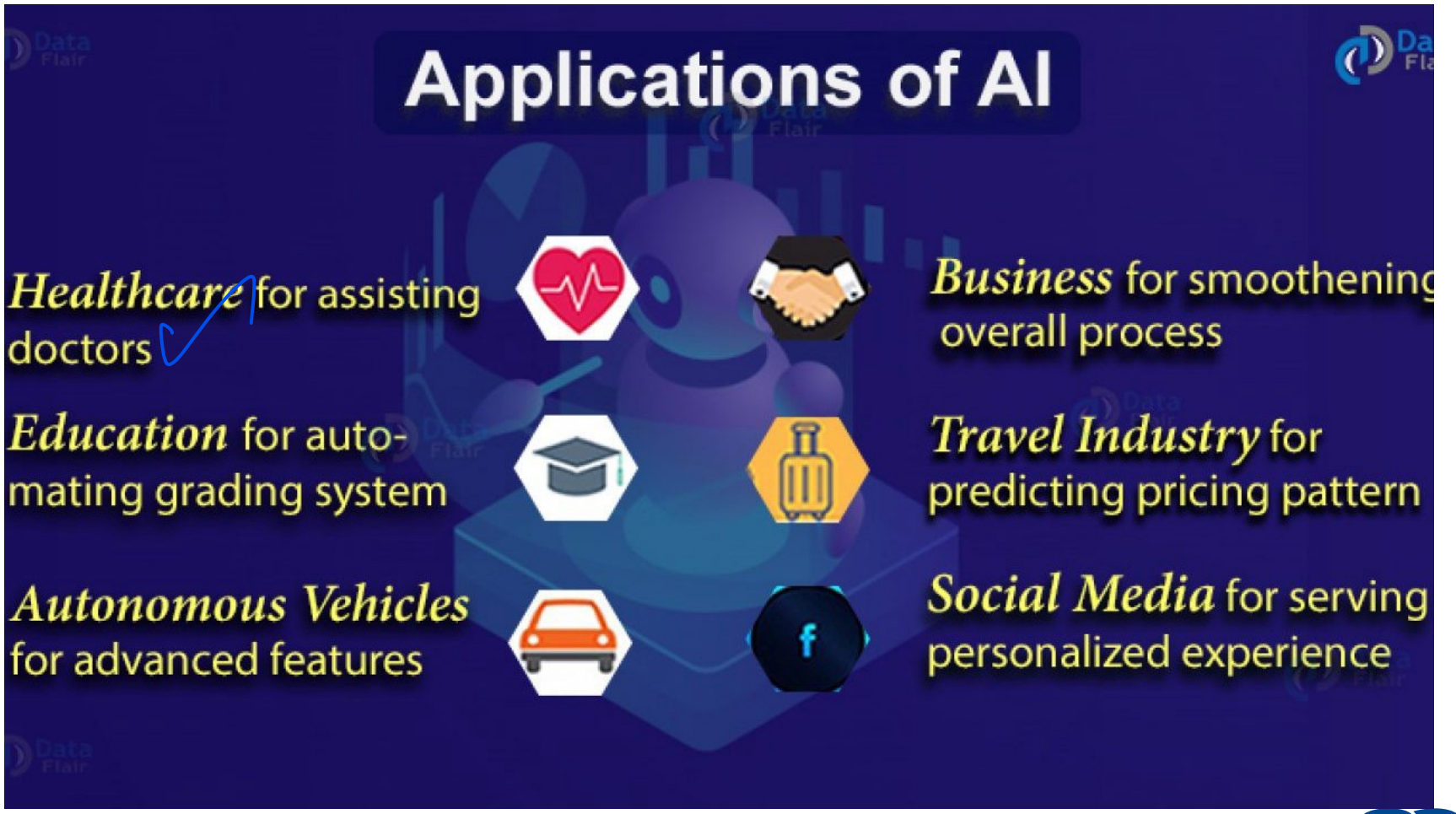
Department of CSE, SSN CE

AI-ML-DL



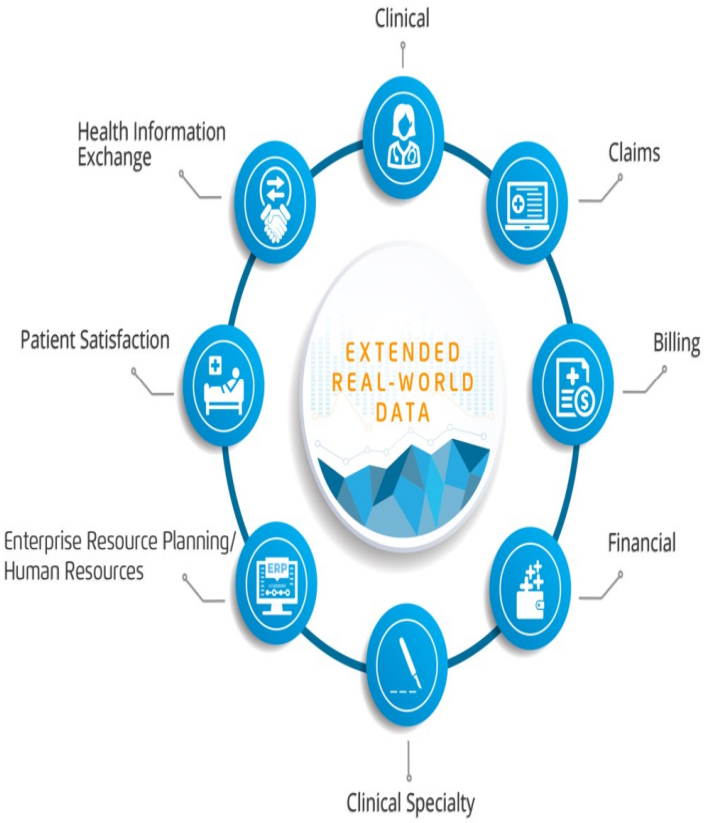
Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

Applications of AI



Healthcare Dimensions

- Data promises different dimensions in health care - Prediction of Chronic Diseases, Personalized Medicine, Precision Medicine, Drug Discovery, Remote Health Monitoring, Genomic Analysis, Patient's Data Management – Fraudulent Medical Claims.
- Medical imaging - improves medical decision making and reduces unnecessary medical procedures



Data Sources in Healthcare

Numerical
Images
Text
Signals
Applicat

Handshake of AI with Healthcare

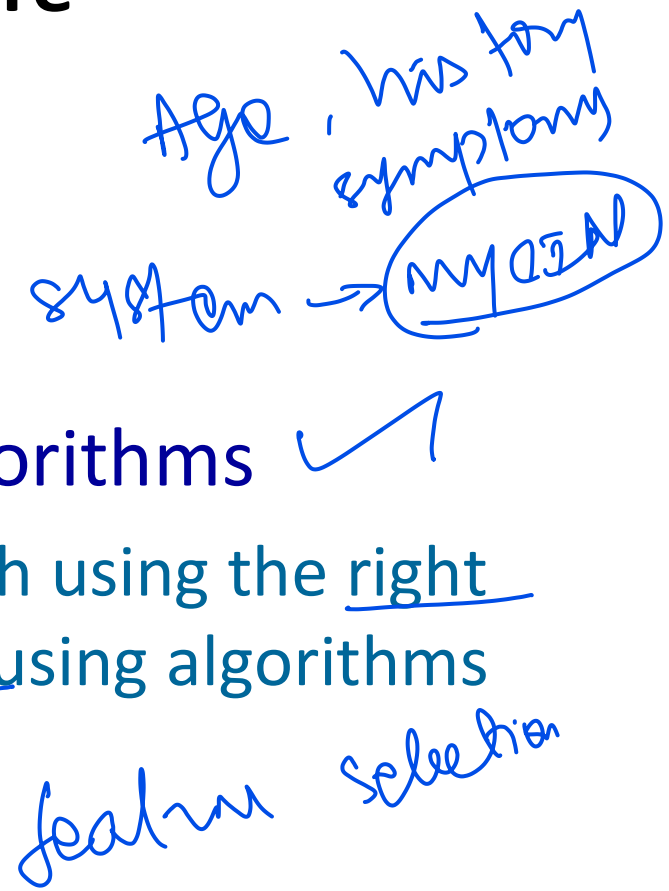
- Healthcare faces a overflow of rich data that is enables to operate with greater insight and effectiveness, and deliver better service.
- Advanced AI techniques coupled with the explosion of data in healthcare to uncover
  - leading clinical practices,
  - minimize research discovery time,
  - offer new personalized engagement paradigms at an industrial scale that align people's decisions and actions in ways that improve outcomes.

Objectives

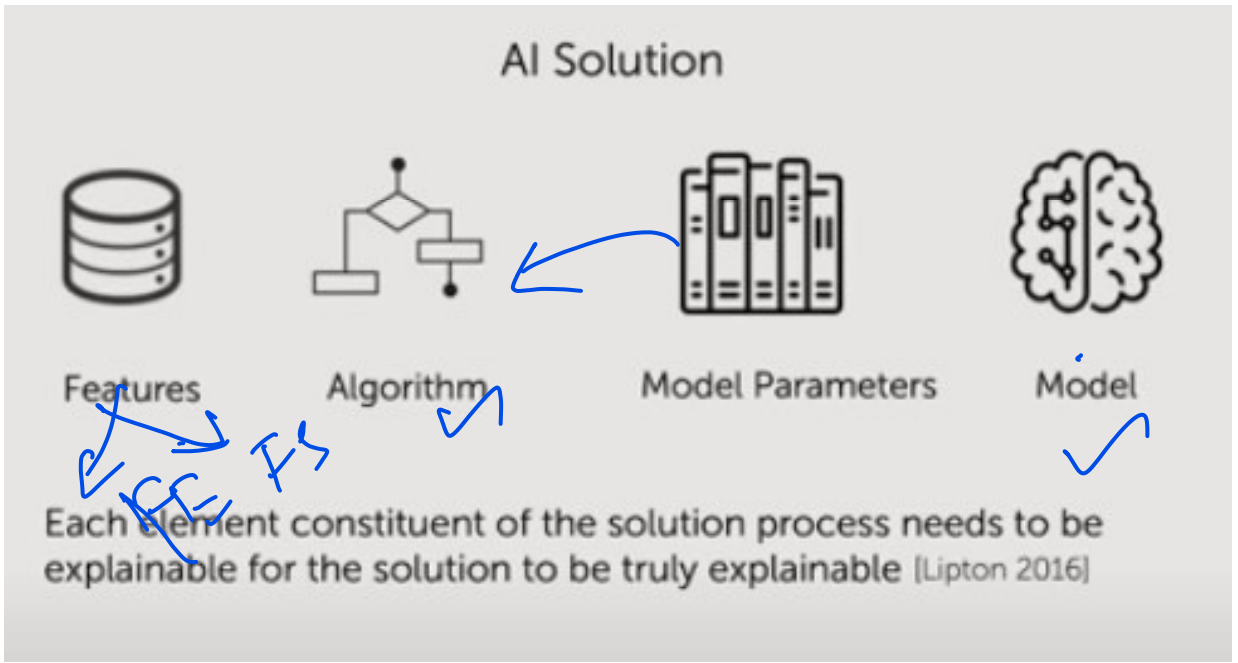
- Why?
  - AI and ML needs to provide explanation in healthcare
- What?
  - Type of explanation do we need in healthcare
- How?
  - To select the right ML algorithm for explanation
- Where?
  - Settings and different types of interpretable ML models
- When?
  - The past, present and future of explainable AI in healthcare

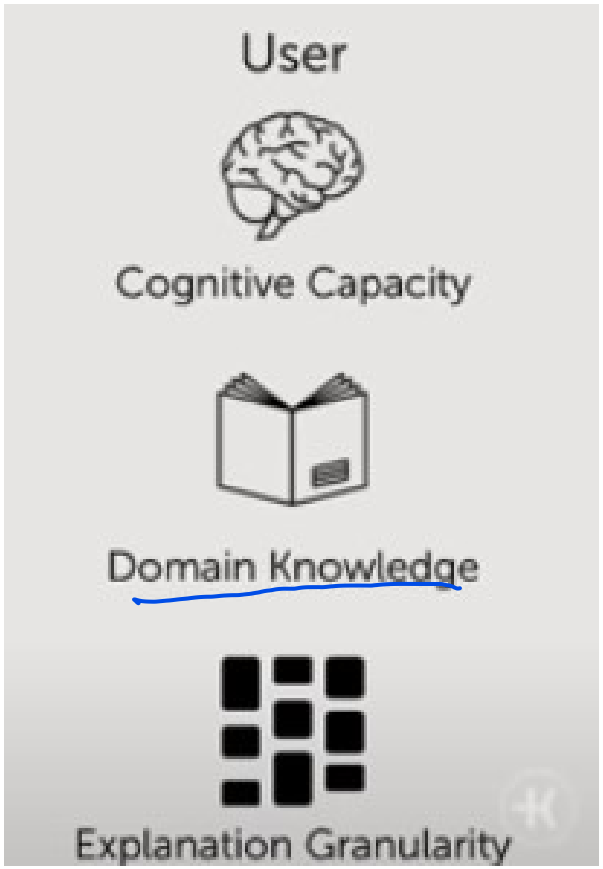
Decisions in Healthcare

- Heuristics
- Rule based systems
- ML based systems – learning algorithms
  - Machine learning is concerned with using the right features to build the right models using algorithms that achieve the right task

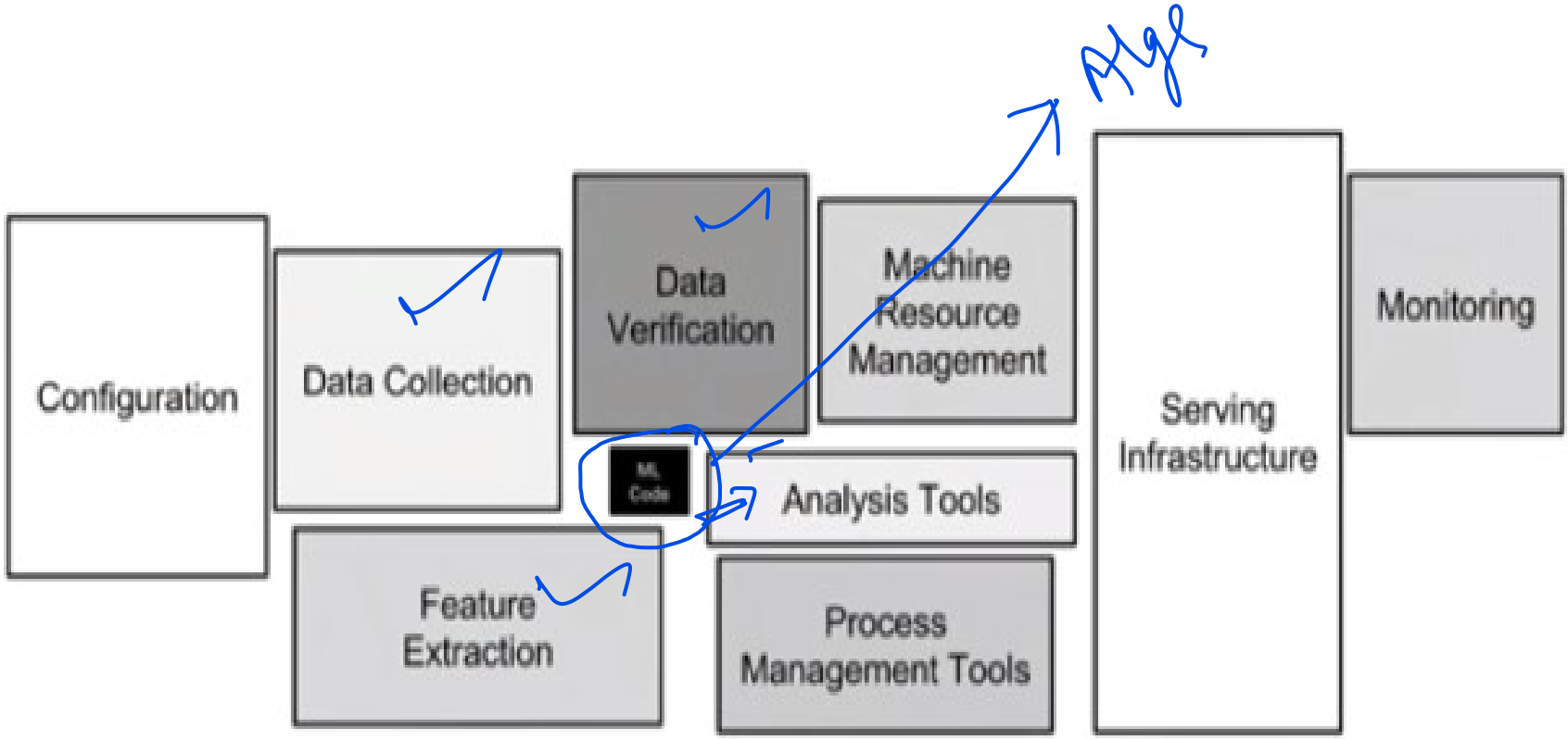


AI Solution


AI Solution
Features
Algorithm
Model Parameters
Model
Each element constituent of the solution process needs to be explainable for the solution to be truly explainable [Lipton 2016]


User
Cognitive Capacity
Domain Knowledge
Explanation Granularity

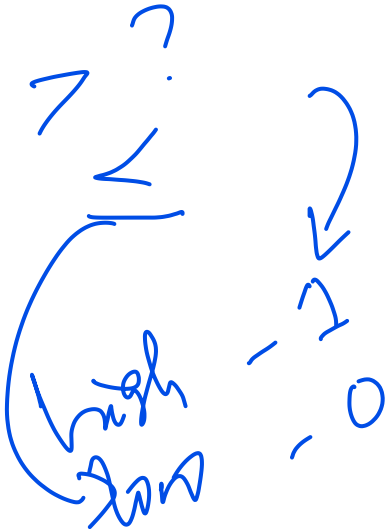
Implementation of AI in Healthcare



Only a small fraction of real-world machine learning systems actually constitutes machine learning code [Sculley 2015].

Complexity of Data

- Syntactic correctness
  - Male, Female encoded as 1 and 0
- Morphological correctness
  - Blood pressure value is 500?
- Semantic correctness
  - Range of blood pressure value encoded as high or low, depends on Child or Adult(Age)

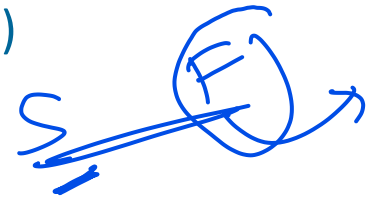


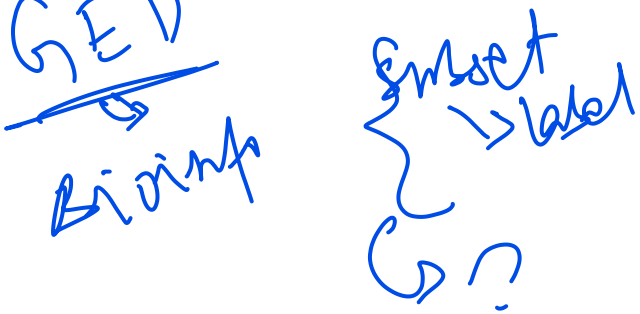
Learning Algorithms

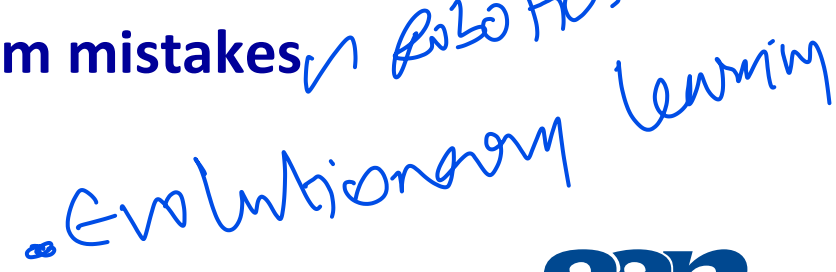
- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.

Classification of Algorithms

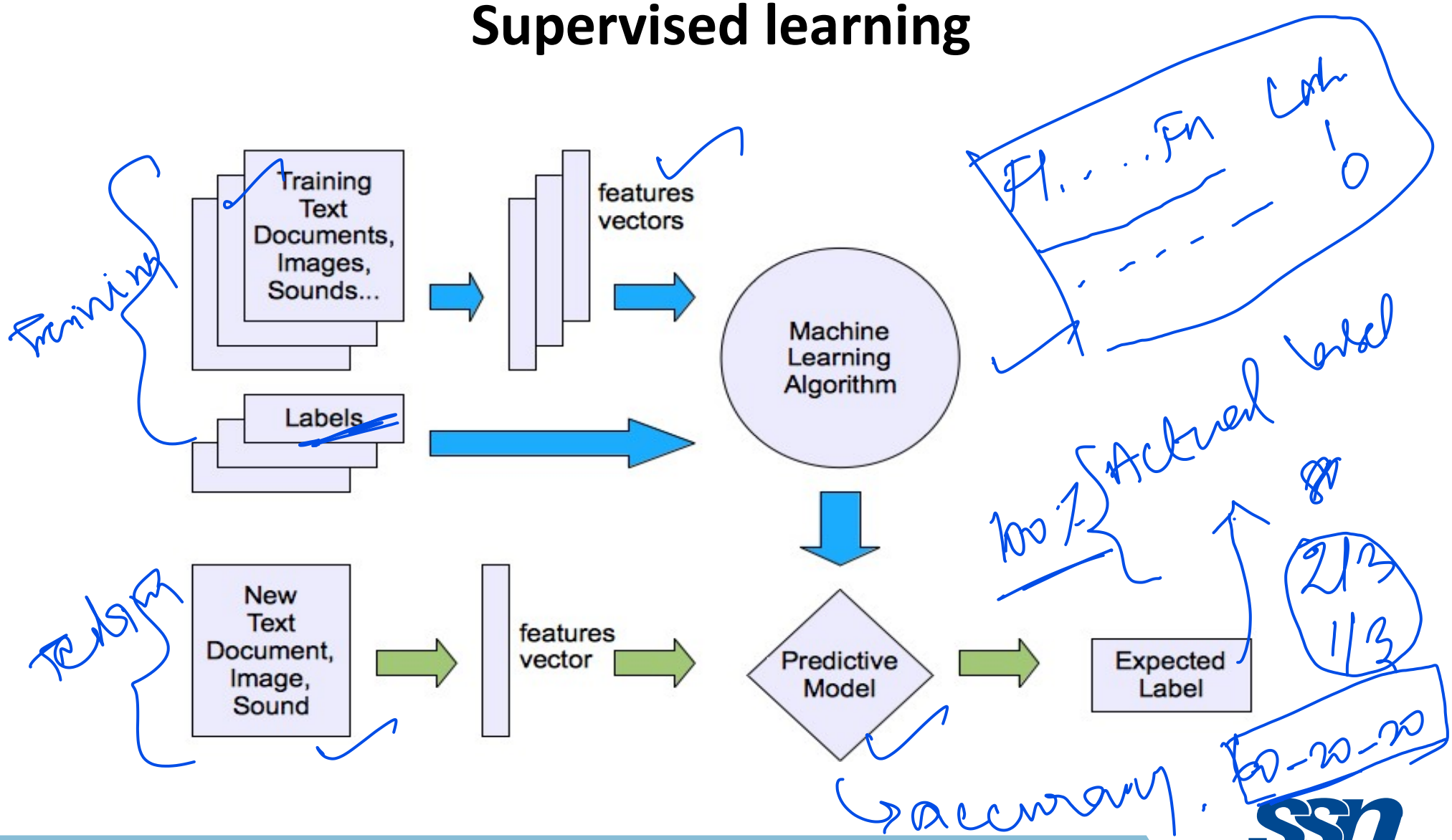
- **Supervised learning – task driven**
  - Prediction $\{x_n \in R^d, y_n \in R\}_{n=1}^N$
  - Classification (discrete labels), Regression (real values)
- **Unsupervised learning – data driven**
  - Clustering $\{x_n \in R^d\}_{n=1}^N$
  - Probability distribution estimation
  - Finding association (in features)
  - Dimension reduction
- **Semi-supervised learning**
- **Reinforcement learning – learning from mistakes**
  - Decision making (robot, chess machine)



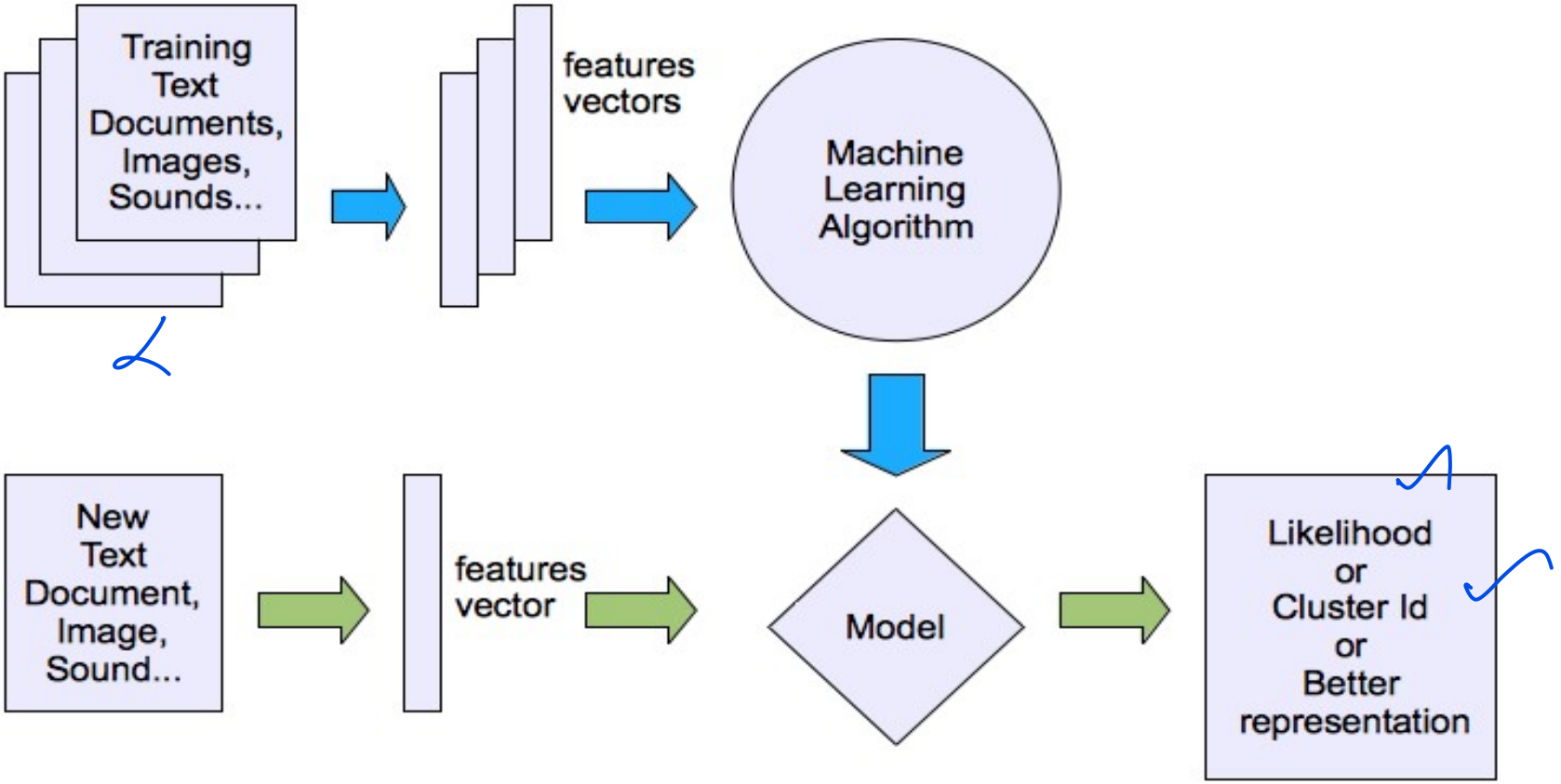




Supervised learning



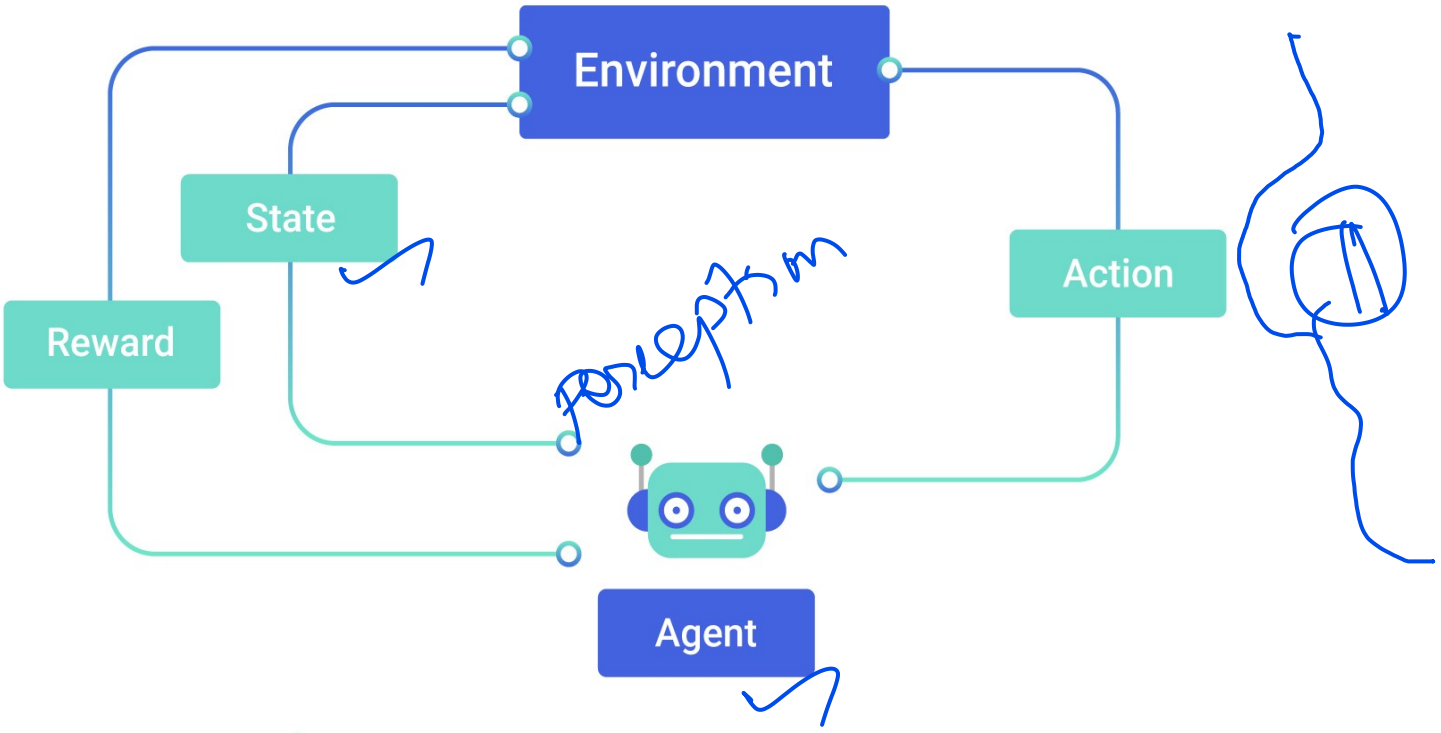
Unsupervised learning



Semi-supervised learning

- Training data includes a few desired outputs.
- Data is cheap, but labelled data is expensive.
- Use small labelled training set to build an initial model, which is then refined using the unlabelled data

Reinforcement learning



What are we seeking?

- Supervised: Low E-out or maximize probabilistic terms

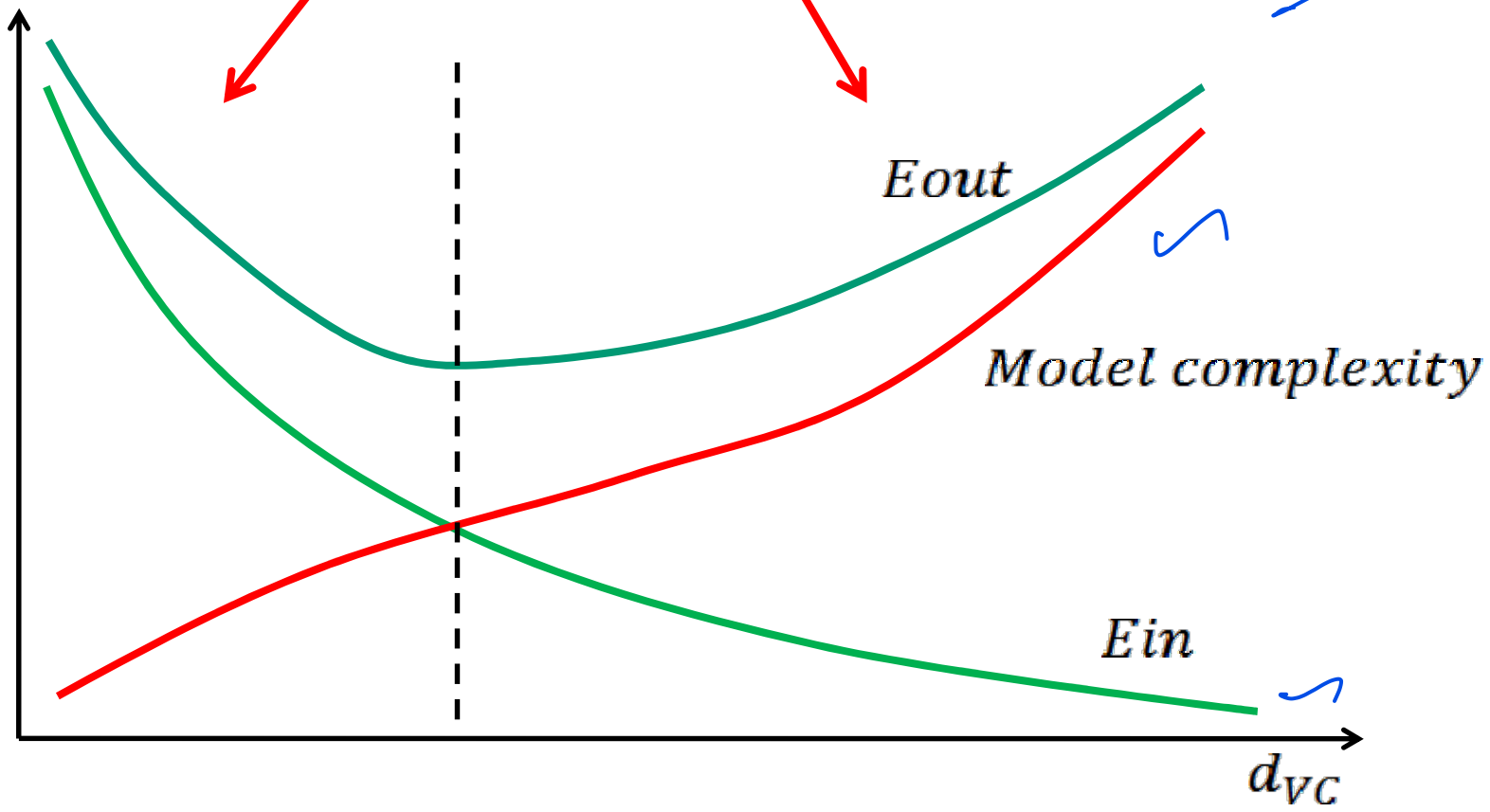
$$error = \frac{1}{N}\sum_{n=1}^{N}[y_n \neq g(x_n)]$$

E-in: for training set
E-out: for testing set
E-out(g)<=E-in(g)

- Unsupervised: Minimum quantization error, Minimum distance, MAP, MLE(maximum likelihood estimation)

What are we seeking?

Under-fitting VS. Over-fitting (fixed *N*)



Generalization

Learning techniques

- Supervised learning categories and techniques
  - **Linear classifier** (numerical functions)
    - Perceptron, logistic regression, support vector machines (SVM), neural networks
  - **Parametric** (Probabilistic functions)
    - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
  - **Non-parametric** (Instance-based functions)
    - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression
  - **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
  - **Aggregation**
    - Bagging (bootstrap + aggregation), Adaboost, Random forest

Learning techniques

- Unsupervised learning categories and techniques
  - **Clustering**
    - K-means clustering
    - Spectral clustering
  - **Density estimation**
    - Gaussian mixture model (GMM)
    - Graphical models
  - **Dimensionality reduction**
    - Principal component analysis (PCA)
    - Factor analysis

ICA

ML Problems

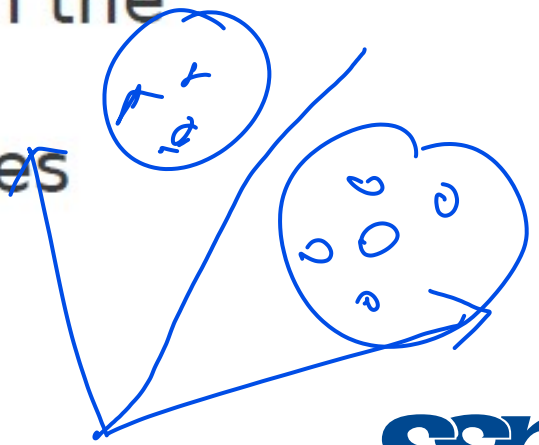
	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Linear vs Non-linear techniques

- Linear classification techniques
  - Perceptron
  - Logistic regression
  - Linear SVM
  - Naïve Bayes
- Non-linear classification techniques
  - *k*-nearest neighbors
  - Non-linear SVM
  - Neural networks (MLP)
  - Decision trees
  - Random forest

Linearly Separable

- Any pattern classification problem that can be solved by finding a linear decision boundary is called as linearly separable
- We need an algorithm to learn the weights, and thus a decision boundary, from given examples

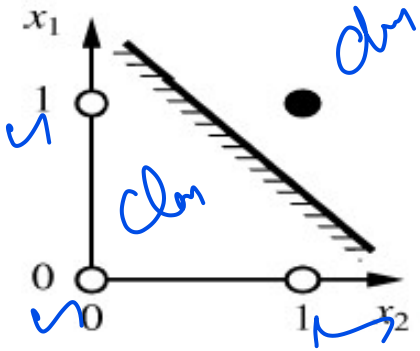


Example – Logic gates

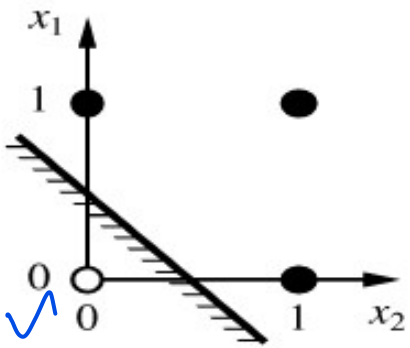
- Thus, single layer network represents a linear separator (line, plane, hyperplane) in the input space

  $\sum_j W_j x_j = 0$ or $\mathbf{W} \cdot \mathbf{x} = 0$

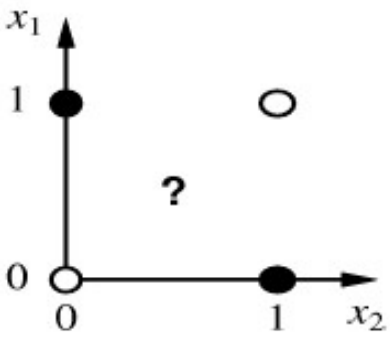
- This is not adequate for many pattern recognition problems!



(a) x_1 **and** x_2



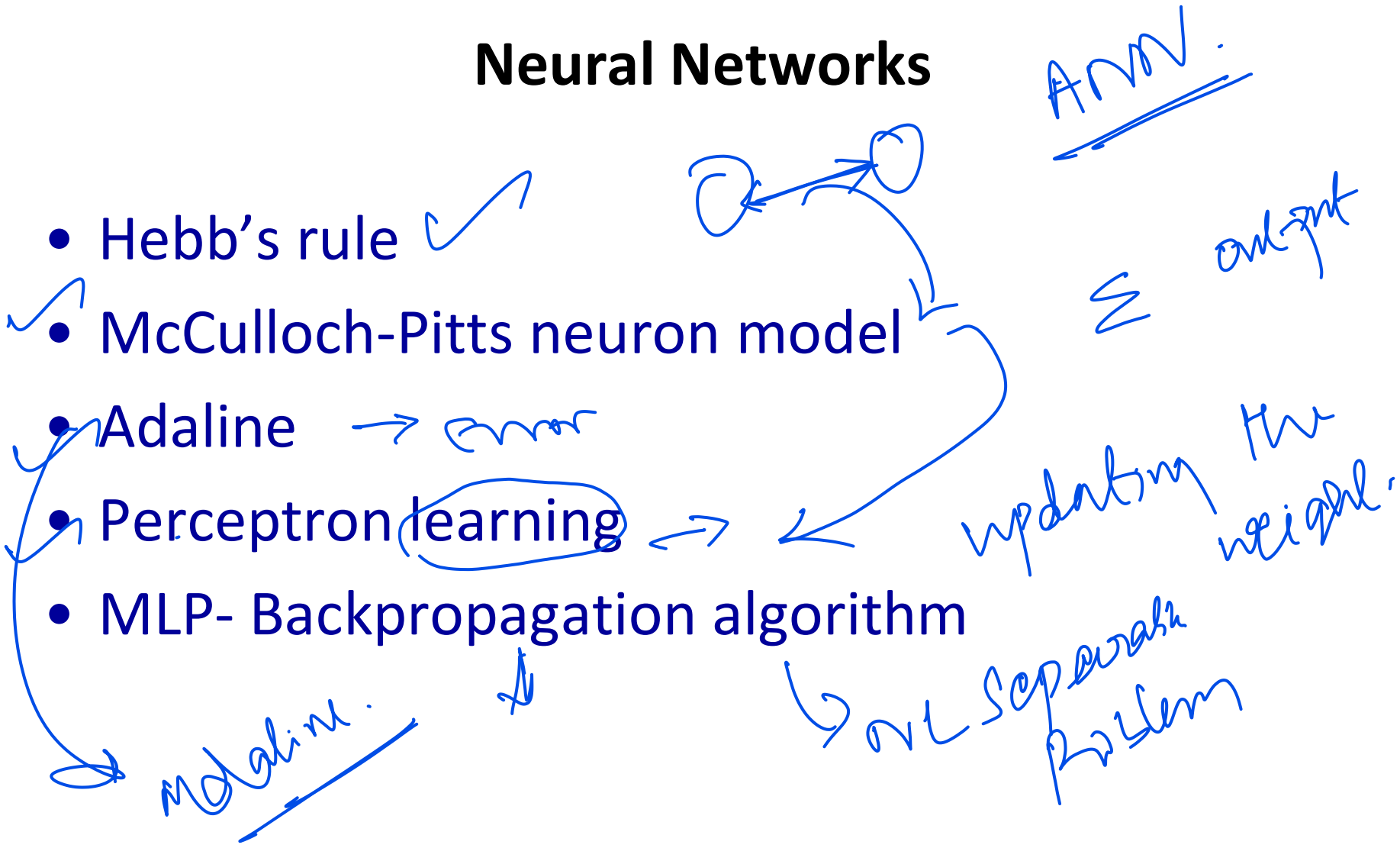
(b) x_1 **or** x_2



(c) x_1 **xor** x_2

Neural Networks

- Hebb's rule
- McCulloch-Pitts neuron model
- Adaline
- Perceptron learning
- MLP- Backpropagation algorithm



McCulloch-Pitts neuron model

$$h = \sum_{i=1}^{m} w_i x_i,$$

$$o = g(h) = \begin{cases} 1 \text{ if } h > \theta \\ 0 \text{ if } h \leq \theta. \end{cases}$$

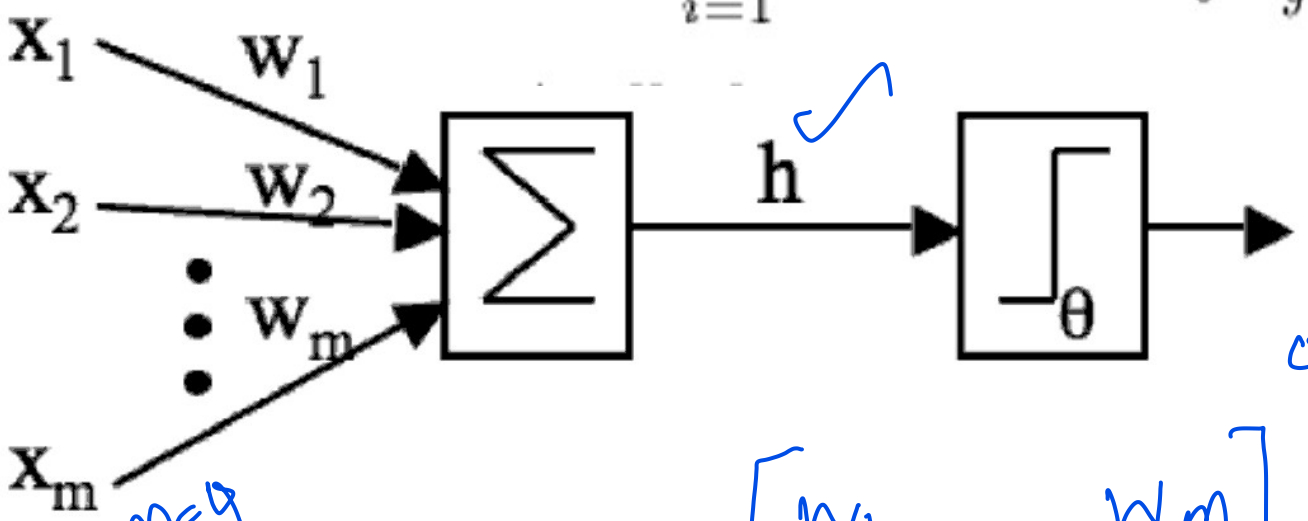


FIGURE 1.6: A picture of McCulloch and Pitt's mathematical model of a neuron. The inputs x_i are multiplied by the weights w_i, and the neurons sum their values. If this sum is greater than the threshold θ then the neuron fires, otherwise it does not.

Perceptron – learning algorithm

- The value of the learning rate decides how fast the network learns, typically $0.1 < \eta < 0.4$
- learning rate is a crucial parameter for convergence
- Weight updation: wij: $w_{ij} \leftarrow w_{ij} + \eta(t_j - y_j)x_i$.
- Bias: fixed (usually the value of -± is chosen)

Perceptron – learning algorithm

- **Initialisation**
  - set all of the weights w_{ij} to small (positive and negative) random numbers
- **Training**
  - for T iterations or until all the outputs are correct:
    - for each input vector:
      - compute the activation of each neuron j using activation function g:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} 1 & \text{if } \sum_{i=0}^{m} w_{ij}x_i > 0 \\ 0 & \text{if } \sum_{i=0}^{m} w_{ij}x_i \leq 0 \end{cases} \quad (3.4)$$

      - update each of the weights individually using:

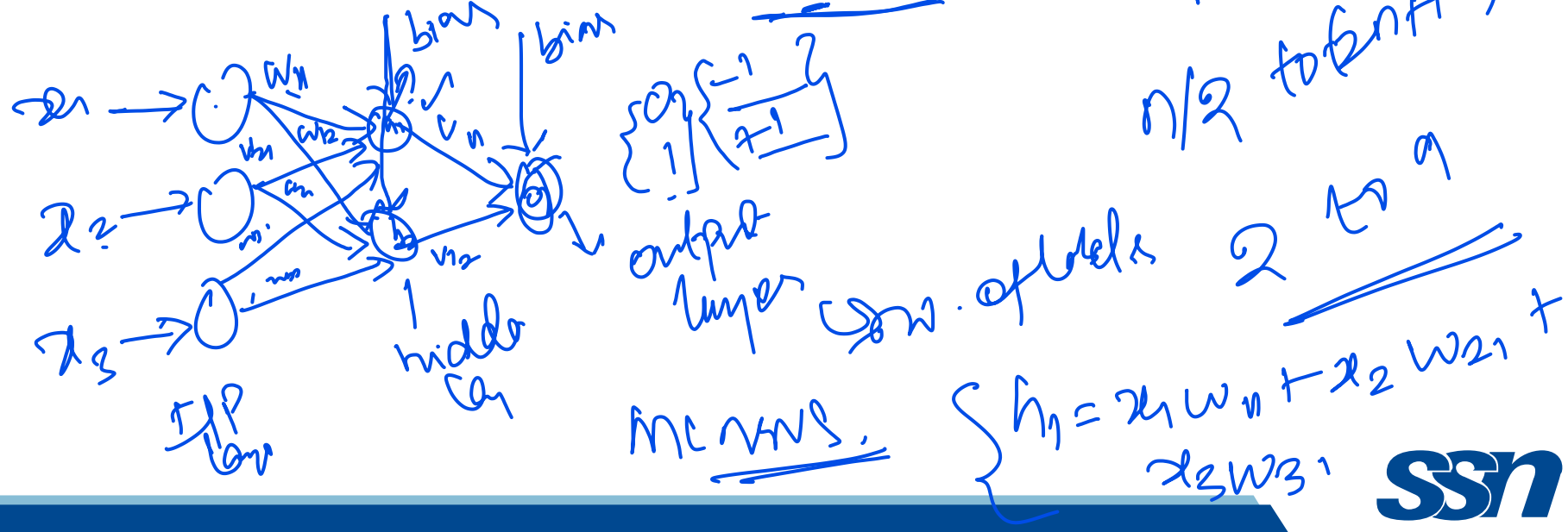
$$w_{ij} \leftarrow w_{ij} - \eta(y_j - t_j) \cdot x_i \quad (3.5)$$

- **Recall**
  - compute the activation of each neuron j using:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} 1 & \text{if } w_{ij}x_i > 0 \\ 0 & \text{if } w_{ij}x_i \leq 0 \end{cases} \quad (3.6)$$

Complexity

- The recall phase loops over the neurons, and within that loops over the inputs, so its complexity is O(mn).
- The training part does this same thing, but does it for T iterations, so costs O(Tmn).



Date:

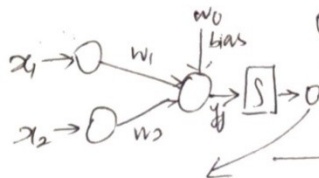
OR:

x_1	x_2	Target
0	0	0
0	1	1
1	0	1
1	1	1

(x_0) bias $= 0$; $w_0 = w_1 = w_2 = 0$

Theta = 1

$\eta = 1$



$$0 = g(y_j) = \begin{cases} 1 & y_j \geq 1 \\ 0 & y_j < 1 \end{cases}$$

(actual = 0)

$$y_j = w_0 \times \text{bias} + x_1 \times w_1 + x_2 \times w_2$$

$$w_i = w_i + \eta (t_j - y_j) \times x_i$$

Epoch

I $x_1 = 0 \; x_2 = 0$ $\quad y_1 = 0 < 1$ $\quad 0 = 0$

$w_0 = 0 + 1(0-0) \times 0 = 0$

$w_1 = 0 \quad w_2 = 0$

$[0\ 0\ 0]$

$x_1 = 0 \; x_2 = 1$

$y_2 = 0 < 1 \therefore$ $\quad 0 = 0$

$w_0 = 0 + 1(1-0) \times 0 = 0$

$w_1 = 0 + 1(1-0) \times 0 = 0$

$w_2 = 0 + 1(1-0) \times 1 = 1$

$[0\ 0\ 1]$

$x_1 = 1 \; x_2 = 0$

$y_3 = 0 < 1 \therefore$ output $= 0$

$w_0 = 0 + 1(1-0) \times 0 = 0$

$w_1 = 0 + 1(1-0) \times 1 = 1$

$w_2 = 0 + 1(1-0) \times 0 = 1$ $\quad [0\ 1\ 1]$

$x_1 = 1 \; x_2 = 1$ $\quad y_4 = 0 + 1 \times 1 + 1 \times 1 = 2 \geq 1$

$\therefore$ output $= 1$

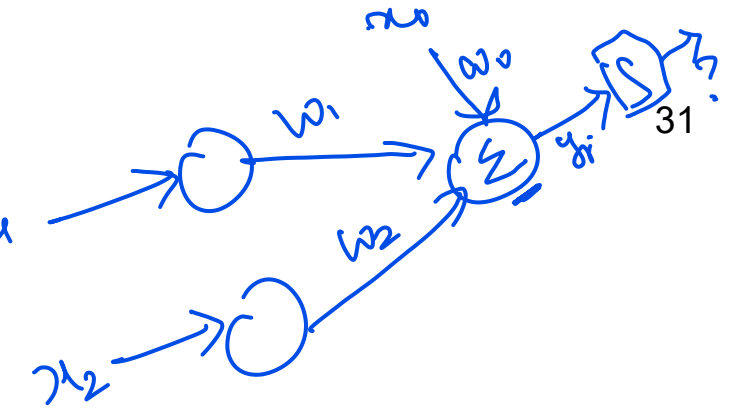
$w_0 = 0 + 1(1-1) \times 0 = 0$

$w_1 = 1 + 1(1-1) \times 1 = 1$

$w_2 = 1 + 1(1-1) \times 1 = 1$ $\quad [0\ 1\ 1]$

$[w_0\ w_1\ w_2]$

Target	Actual
0	0
0	0
1	1
1	1



Epoch (II)

$x_1 = 0, x_2 = 0$ $y_1 = 0 < 1$ output = 0 ✓ → weight → activation

$w_0 = 0 + 1(0-0)*0 = 0$
$w_1 = 1 + 1(0-0)*0 = 1$
$w_2 = 1 + 1(0-0)*0 = 1$ [0 1 1]

$x_1 = 0, x_2 = 1$ $y_2 = 0+0+1 = 1$ ∴ output = 1

$w_0 = 0$
$w_1 = 1 + 1(1-1)*0 = 1$
$w_2 = 1 + 1(1-1)*1 = 1$ [0 1 1]

$x_1 = 1, x_2 = 0$ $y_3 = 0+1+0 = 1$ ∴ output = 1

$w_0 = 0+1(1-1)*0 = 0$
$w_1 = 1 + 1(1-1)*1 = 1$
$w_2 = 1 + 1(1-1)*0 = 1$ [0 1 1]

$x_1 = 1, x_2 = 1$ $y_4 = 0+1+1 = 2$ ∴ output = 1

$w_0 = 0 + 1(1-1)*0 = 0$
$w_1 = 1 + 1(1-1)*1 = 1$
$w_2 = 1 + 1(1-1)*1 = 1$ [0 1 1]

Epoch.

3 ✓
1

x_1	x_2	Target.	actual	w_0	w_1	w_2

Pros and Cons of NN

- Neural networks are good to model with nonlinear data with large number of inputs.
- Once trained, the predictions are pretty fast.
- Neural networks are flexible for both regression and classification problems.
- Any data which can be made numeric can be used in the model, as neural network is a mathematical model with approximation functions.

Pros and Cons of NN

- It is computationally very expensive and time consuming to train with traditional CPUs.
- Neural networks are black boxes, that is how each independent variable is influencing the dependent variables.
- Neural networks depend a lot on training data. This leads to the problem of over-fitting and generalization.

Decision tree

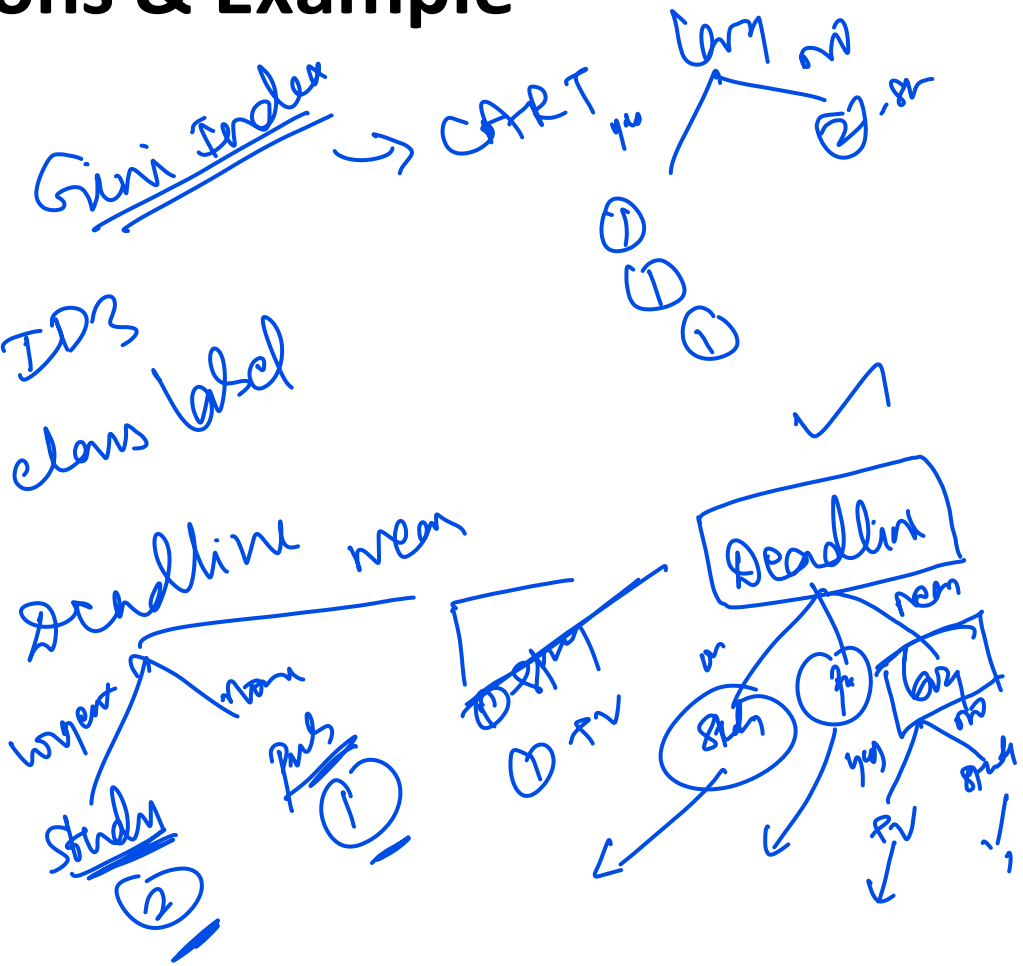
- A feature tree is a tree such that each internal node is labeled with a feature and each edge emanating from an internal node is labeled with a literal.
- The set of literals at a node is called a split.
- Each leaf of the tree represents a logical expression, which is the conjunction of literals on the path from root to the leaf.
- A leaf node represents a class label h(x) or class label distribution

Impurity functions & Example

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

	Deadline	Lazy	Activity type
1	Urgent	Yes	Study
2	None	Yes	Pub
3	Near	No	Study
4	Near	Yes	TV
5	Urgent	No	Study



$-P_i \log_2 P_i$

Date:

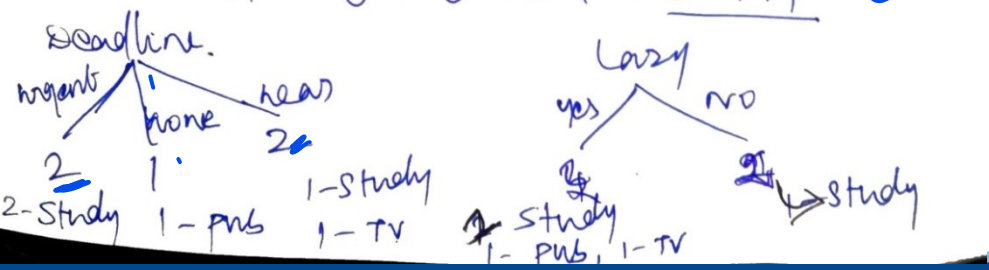
$$\text{Entropy}(S_1) = \frac{-3}{5}\log_2\frac{3}{5} - \frac{1}{5}\log_2\frac{1}{5} - \frac{1}{5}\log_2\frac{1}{5}$$

$$= 1.371.$$

$$\text{Gain}(S_1, \text{deadline}) = \text{Entropy}(S_1) - \frac{|S_{1\text{urgent}}|}{|S_1|}\text{Entropy}(S_{1\text{urgent}}) - \frac{|S_{1\text{none}}|}{|S_1|}\text{Entropy}(S_{1\text{none}}) - \frac{|S_{1\text{near}}|}{|S_1|}\text{Entropy}(S_{1\text{near}})$$

$$= 1.371 - \frac{2}{5}\left(-\frac{2}{2}\log_2\frac{2}{2}\right) - \frac{1}{5}\left(\frac{-1}{1}\log_2\frac{1}{1}\right) - \frac{2}{5}\left(\frac{-1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right)$$

$$= 1.371 - 0 - 0 - 0.4 = 0.971$$



Date:

$$\text{Gain}(S_1, \text{Lazy}) = \text{Entropy}(S_1) - \frac{|S_{1\,yes}|}{|S_1|}\,\text{entropy}(S_1) - \frac{|S_{1no}|}{|S_1|}\,\text{entropy}(S_1)$$

$$= 1.371 - \frac{3}{5}\left(\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) - \cancel{\frac{1}{4}\log_2\frac{1}{4}}\Big) - \frac{2}{5}\left(\frac{2}{2}\log_2\frac{2}{2}\right)$$

$$= 1.371 - (3/5) * 0.933 - 0 = \underline{0.811}$$

ID3(*X, T, Attrs)*
X: training examples: T: target attribute: Attrs: other attributes, initially all attributes

- Create *Root* node
  If all X's are +, *return Root* with class +
  If all X's are –, *return Root* with class –
  If Attrs is empty *return Root* with class most common value of *T* in *X*
  else
  A ← best attribute; decision attribute for Root ← *A*
  For each possible value v_i of *A:*
    - add a new branch below *Root,* for test $A = v_i$
    - X_i ← subset of *X* with $A = v_i$
    - *If* X_i is empty *then* add a new leaf with class the most common value of *T* in *X*
      else add the subtree generated by ID3(X_i, *T, Attrs* ← {*A*})
  return Root

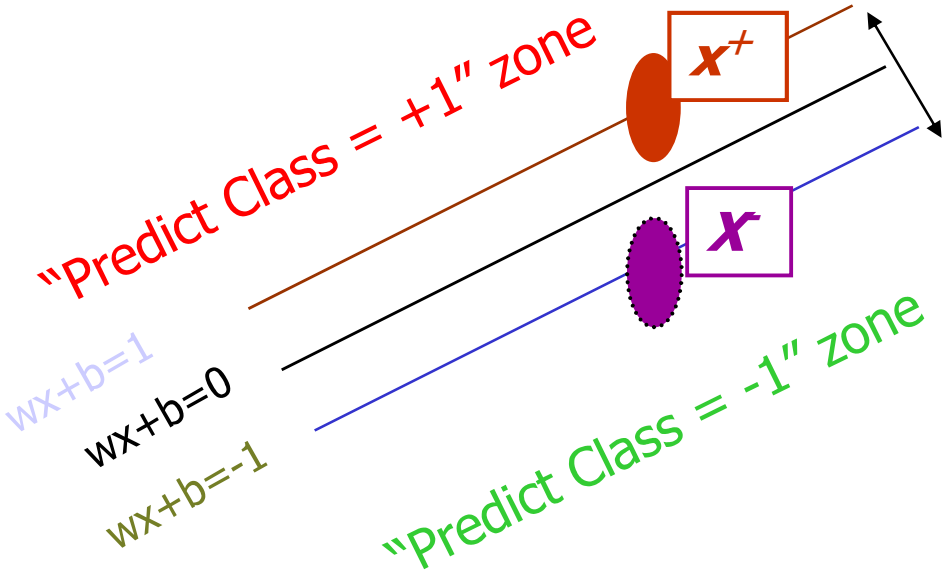
Strengths of decision tree

- Decision trees are able to generate understandable rules.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Weaknesses of decision tree

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.

Linear SVM Mathematically



M=Margin Width

What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^{+} + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^{-} + b = -1$
- $\boldsymbol{w} \cdot (\boldsymbol{x}^{+} - \boldsymbol{x}^{-}) = 2$

$$M = \frac{(x^{+} - x^{-}) \cdot w}{|w|} = \frac{2}{|w|}$$

We can formulate a quadratic Optimization Problem and solve for w and b, for all i

$$y_i (wx_i + b) \geq 1$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

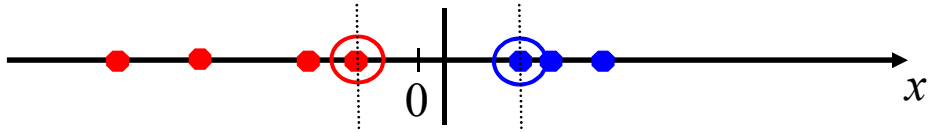
- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

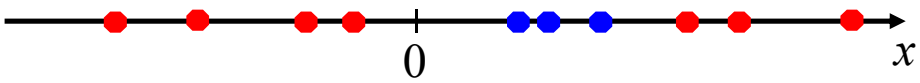
- **Parameter *C* can be viewed as a way to control overfitting.**

Non-linear SVMs

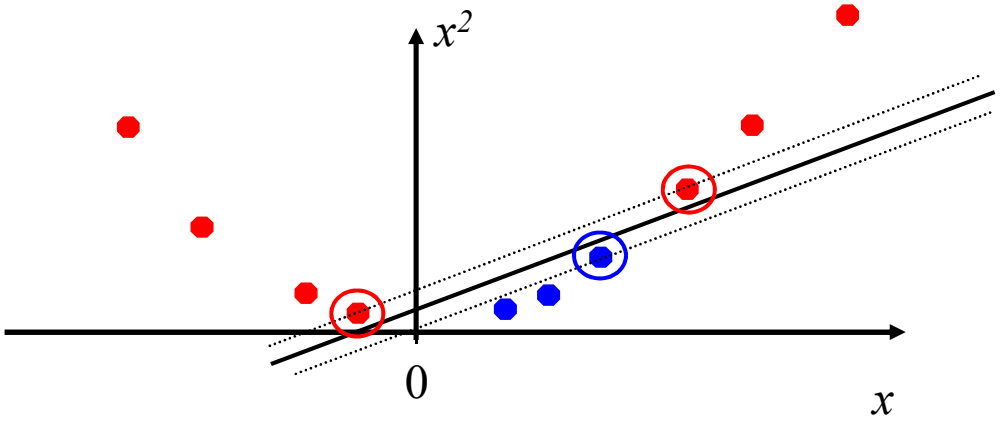
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

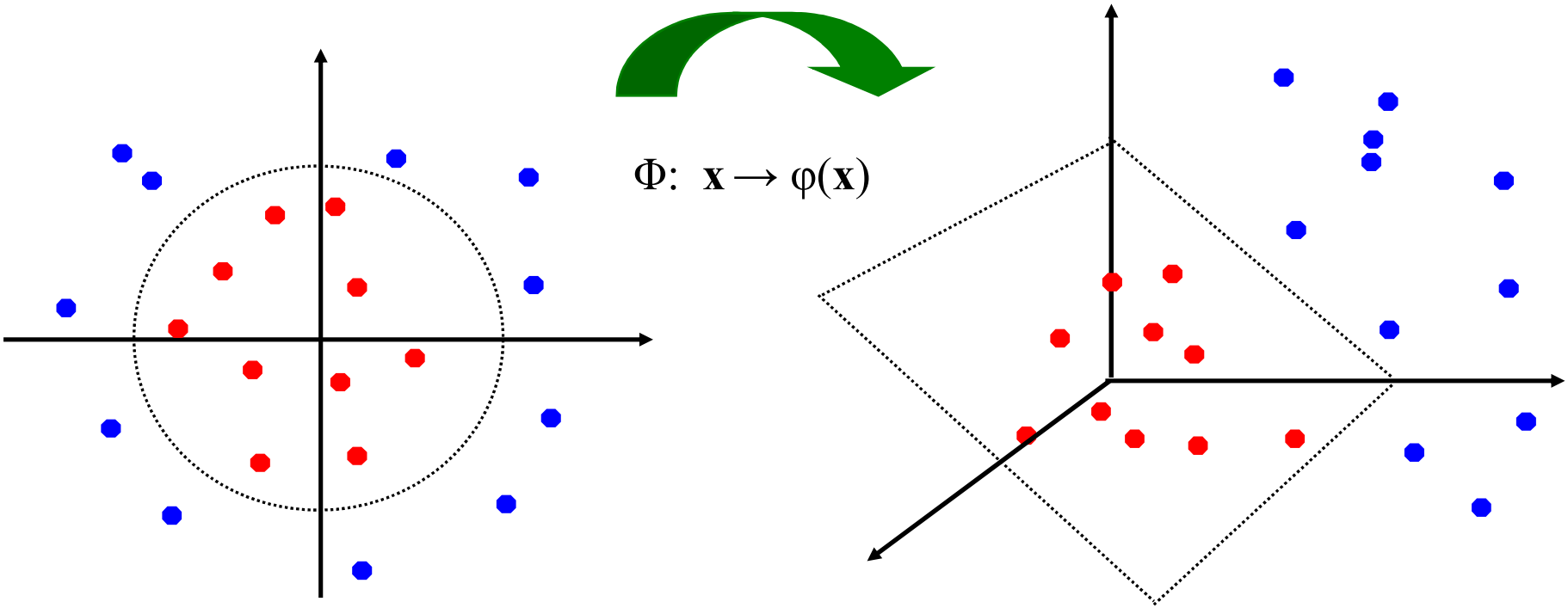


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Strengths and Weaknesses of SVM

Strengths

- Suitable for classification, regression and outliers detection.
- Effective in high dimensional spaces where the number of dimensions is greater than the number of samples.
- Uses support vectors

Weaknesses

- Sensitive to noise : small number of mislabeled examples can dramatically decrease the performance
- Considers two classes : how to do multi-class classification with SVM?

  1) with output classes m, learn m SVM's
  - SVM 1 learns "Output==1" vs "Output != 1"
  - SVM 2 learns "Output==2" vs "Output != 2"
  - :
  - SVM m learns "Output==m" vs "Output != m"

  2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Performance metrics – Overall

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Performance metrics - Example

		Patients with bowel cancer (as confirmed on endoscopy)			
		Condition positive	Condition negative	Prevalence = (TP + FN) / Total_Population = (20 + 10) / 2030 ≈ **1.48%**	Accuracy (ACC) = (TP + TN) / Total_Population = (20 + 1820) / 2030 ≈ **90.64%**
Fecal occult blood screen test outcome	Test outcome positive	**True positive** (TP) = 20 (2030 × 1.48% × 67%)	**False positive** (FP) = 180 (2030 × (100 − 1.48%) × (100 − 91%))	Positive predictive value (PPV), Precision = TP / (TP + FP) = 20 / (20 + 180) = **10%**	False discovery rate (FDR) = FP / (TP + FP) = 180 / (20 + 180) = **90.0%**
	Test outcome negative	**False negative** (FN) = 10 (2030 × 1.48% × (100 − 67%))	**True negative** (TN) = 1820 (2030 × (100 − 1.48%) × 91%)	False omission rate (FOR) = FN / (FN + TN) = 10 / (10 + 1820) ≈ **0.55%**	Negative predictive value (NPV) = TN / (FN + TN) = 1820 / (10 + 1820) ≈ **99.45%**
		TPR, Recall, **Sensitivity** = TP / (TP + FN) = 20 / (20 + 10) ≈ **66.7%**	False positive rate (FPR), Fall-out, probability of false alarm = FP / (FP + TN) = 180 / (180 + 1820) = **9.0%**	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$ = (20 / 30) / (180 / 2000) ≈ **7.41**	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$ ≈ **20.2**	F_1 score = $2 \times \frac{Precision \times Recall}{Precision + Recall}$ ≈ **0.174**
		False negative rate (FNR), Miss rate = FN / (TP + FN) = 10 / (20 + 10) ≈ **33.3%**	**Specificity**, Selectivity, True negative rate (TNR) = TN / (FP + TN) = 1820 / (180 + 1820) = **91%**	Negative likelihood ratio (LR−) = $\frac{FNR}{TNR}$ = (10 / 30) / (1820 / 2000) ≈ **0.366**		

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Many classifiers to choose from

- Neural networks
- Naïve Bayes classifier
- Decision tree
- SVM
- Linear / Logistic regression
- K-nearest neighbor
- Ensemble approach
  - Boosting
  - Bagging
  - Random Forest

Which is the best one?

Check your understanding

- Dataset without class labels are suitable for ……. learning.
- Learning through reward function is called ……..
- Machine learning is a sub domain of……
- Learning without feature extraction is called……
- Examples for probabilistic based learning algorithms are ……and.....
- SVM is more suitable for ……. Dataset
- Pruning concept is related with ……algorithm

Check your understanding

- Name any three impurity functions ?
- Misclassification is measured using ……. Variable
- Name any three kernel functions of SVM?
- Name three important activation functions of NN?
- Number of nodes in input layer of NN is equivalent to ……..
- Widrow-Hoff rule is also called as ……. and …….
- NN structure is derived from human……..
- The leaf nodes of decision tree are ……..

References

- Machine Learning: An Algorithmic Perspective, by Stephen Marsland, Third Edition, CRC Press

- Machine Learning: The Art and Science of Algorithms that Make Sense of Data Paperback, by Peter Flach, Cambridge University Press

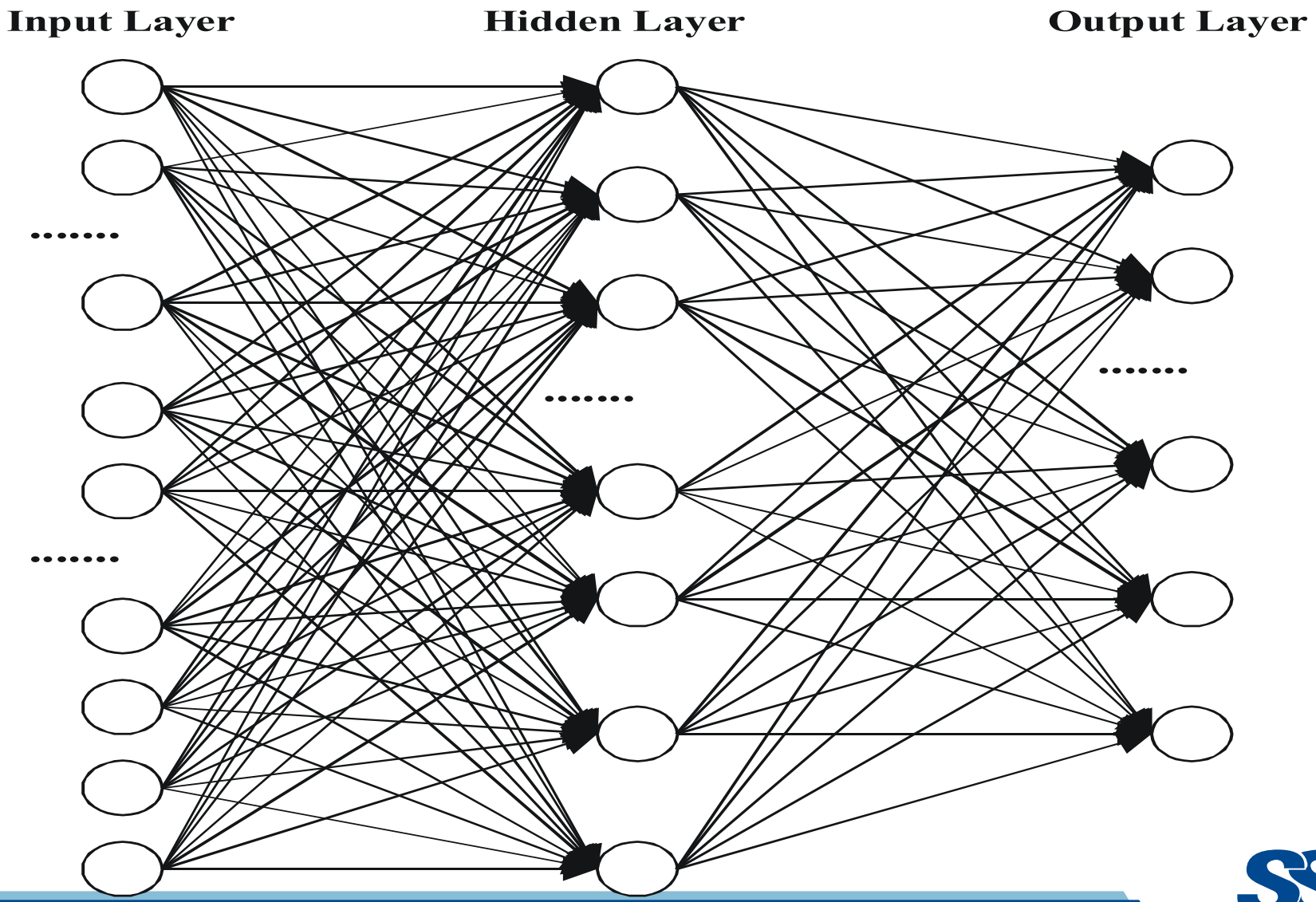
Thank You

Back propagation network

Training a network by back propagation involves 3 stages:

- Feed forward of the input training pattern
- Back propagation of the associated error
- Adjustment of weights

Multi Layer Network



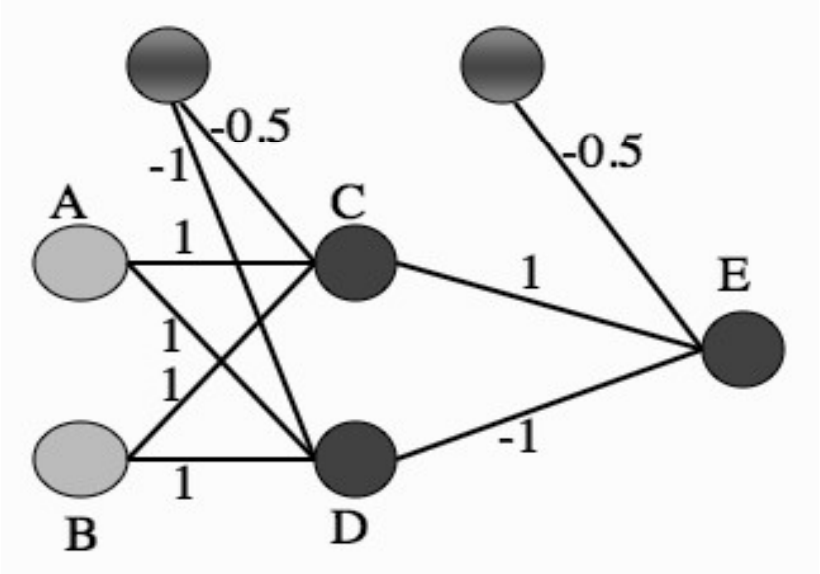
MLP training algorithm using back-propagation

- an input vector is put into the input nodes
- the inputs are fed *forward through the network*
- the inputs and the first-layer weights (here labelled as *v) are used to decide* whether the hidden nodes fire or not. The activation function *g(·) is the sigmoid* function
- the outputs of these neurons and the second-layer weights (labelled as *w) are* used to decide if the output neurons fire or not

Contd…

- the *error is computed as the sum-of-squares difference between the network outputs* and the targets
- this error is fed *backwards through the network in order to* first update the second-layer weights and then afterwards, the first-layer weights

MLP – XOR problem



- Multi-layer Perceptron network showing a set of weights that solve the XOR problem.

Multi-layer Perceptron Algorithm

- **Initialisation**
  - initialise all weights to small (positive and negative) random values
- **Training**
  - repeat:
    - for each input vector:

      Forwards phase:

      - compute the activation of each neuron j in the hidden layer(s) using:

$$h_\zeta = \sum_{i=0}^{L} x_i v_{i\zeta} \tag{4.4}$$

$$a_\zeta = g(h_\zeta) = \frac{1}{1+\exp(-\beta h_\zeta)} \tag{4.5}$$

      - work through the network until you get to the output layer neurons, which have activations (although see also Section 4.2.3):

$$h_\kappa = \sum_{j} a_j w_{j\kappa} \tag{4.6}$$

$$y_\kappa = g(h_\kappa) = \frac{1}{1+\exp(-\beta h_\kappa)} \tag{4.7}$$

Contd…

Backwards phase:

- compute the error at the output using:

$$\delta_o(\kappa) = (y_\kappa - t_\kappa)\, y_\kappa(1 - y_\kappa) \tag{4.8}$$

- compute the error in the hidden layer(s) using:

$$\delta_h(\zeta) = a_\zeta(1 - a_\zeta) \sum_{k=1}^{N} w_\zeta \delta_o(k) \tag{4.9}$$

- update the output layer weights using:

$$w_{\zeta\kappa} \leftarrow w_{\zeta\kappa} - \eta \delta_o(\kappa) a_\zeta^{\text{hidden}} \tag{4.10}$$

- update the hidden layer weights using:

$$v_\iota \leftarrow v_\iota - \eta \delta_h(\kappa) x_\iota \tag{4.11}$$

* (if using sequential updating) randomise the order of the input vectors so that you don't train in exactly the same order each iteration

– until learning stops (see Section 4.3.3)

- **Recall**
  - use the Forwards phase in the training section above

Practice Questions

4. b. Identify the suitable weights for the AND network for the following input and output.

X_1	X_2	T
1	1	1
-1	1	-1
1	-1	-1
-1	-1	-1

Where w0 = w1 = w2 = 0.5, eta = 0.1, bias = -1

Find the updated weight of each sample for one Epoch.

Practice Questions

- Construct a decision tree for “OR function” and write the rules for two inputs and one output