Name: Uddhav P Gautam

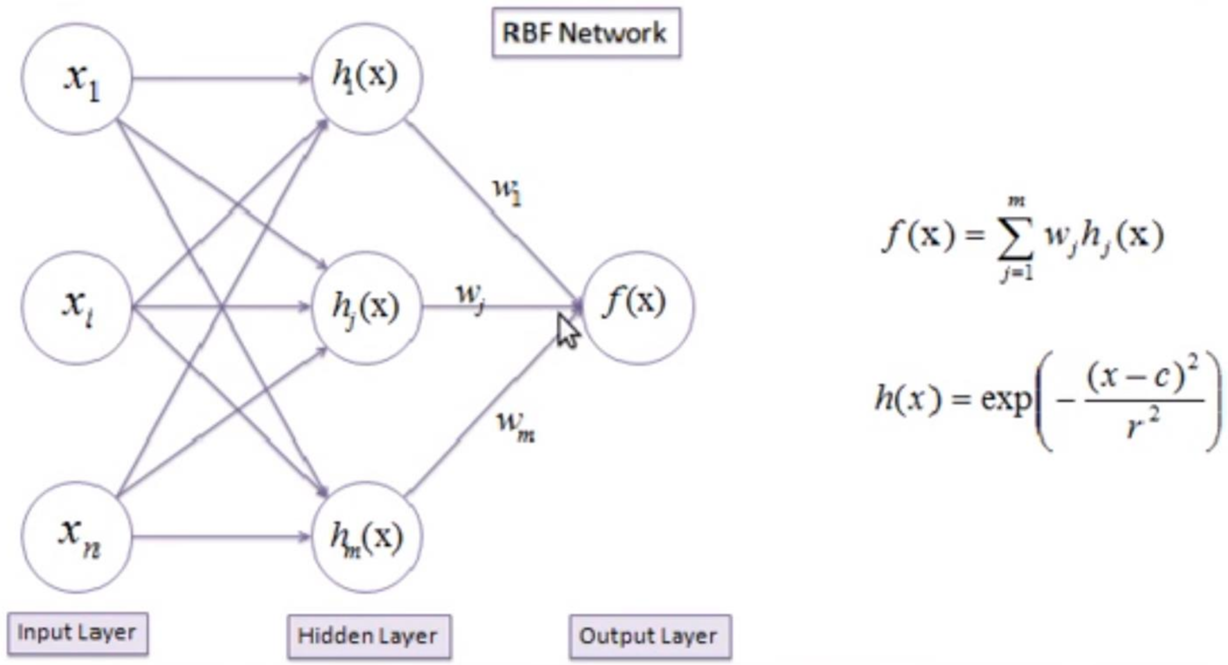
Exam 1 **Due Date: March 9, 2020**

Machine Learning

Task 1: Create a Radial Basis Functions (RBF) **network** that solves the XOR function (20 points)

Soln:

Background

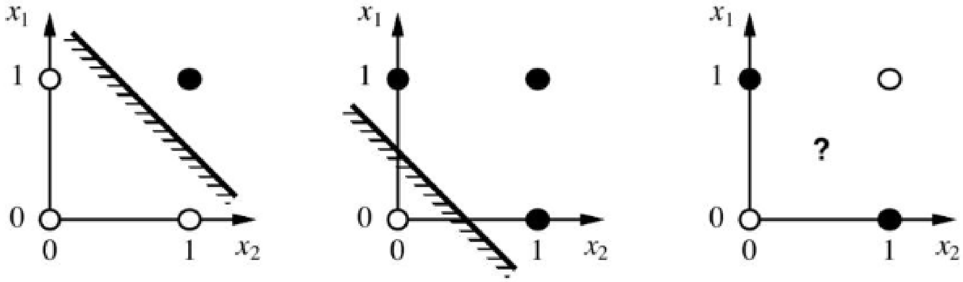


$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j h_j(\mathbf{x})$$

$$h(x) = \exp\left(- \frac{(x-c)^2}{r^2} \right)$$

As shown above, for hidden layer we need to find corresponding weight, c, and r for each RFG gaussian function. The r is the activation function with parameter (radius or standard deviation), c (center or average taken from the input space).

Problem Defn.



x_1 and x_2 x_1 or x_2 x_1 xor x_2

In above figure, black dot means the value is 1, white dot means value is 0. We have linear separability between 1 and 0 in or and and. But there is no linear separability between o and 1 in xor. This is a probleem.

Soln:

RBFN has at least one hidden layer, which is used to change the into 4 dimensions (4-d feature vectors) in order to make inputs linearly separable.

4 receptors t1 (0,0), t2 (0,1), t3 (1,0), and t4 (1,1). 4 corresponding RBF functions are ϕ1, ϕ2, ϕ3, and ϕ4. The corresponding spread (or st. deviation) are σ1, σ2, σ3, and σ4.

Using K-nearest cluster, where K=2. From each receptor ti, two are at a distance of 1 and 1 is at distance of √2. We used -1, 1, 1, -1 corresponding weights for t1, t2, t3, and t4. The output decision is if y=f(x) <0 then 0 otherwise 1

So, root mean square distance between any 2-nearest cluster is 1. There each σi = 1. The RBF function is ϕ1 is $e^{-||x-t1||/2\sigma^2}$

Putting all these values in table,

Inputs (x)	ϕ1 [note: t1=0,0] w1=-1	Φ2 w2=1	Φ3 w3=1	Φ4 w4=-1	y=f(x) = $\sum w_i \varphi_i$	Output decision
0, 0	1	0.6	0.6	0.4	-0.2	0
0, 1	0.6	1	0.4	0.6	0.2	1
1, 0	0.6	0.4	1	0.6	0.2	1
1, 1	0.4	0.6	0.6	1	-0.2	0

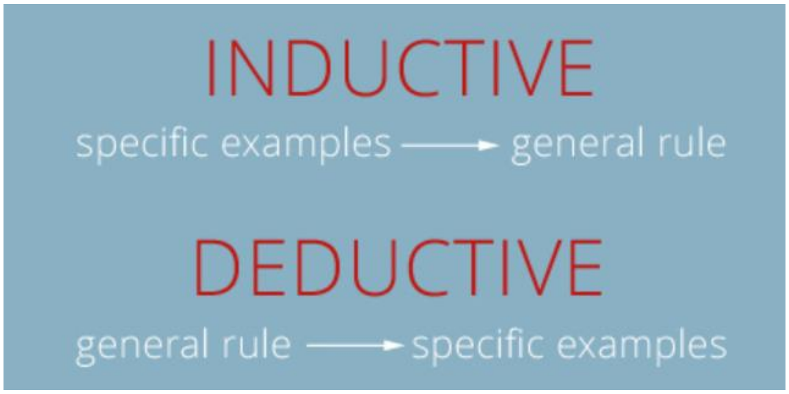
Task 2: What is meant by the term overfitting in the context of inductive learning?

What can cause overfitting when training an artificial neural network on training data? How can one avoid overfitting? (20 points)

Soln:

We estimate a function from some **input-output pairs** with no (or very little) knowledge. This learning is Inductive learning. This function is a general rule from a set of observed instances. Over the time, the function improves for all observed instances. There is always **inductive bias**. It's a type of supervised learning.

In supervised learning, we provide the **correct function** for **particular inputs**, and based on the **feedback** (learning) the supervised learning function improves itself to better match the output.



INDUCTIVE

specific examples → general rule

DEDUCTIVE

general rule → specific examples

Overfitting in inductive learning: When inductive bias. is too tailored for some specific dataset then that's a overfitting in inductive leanring. So, this inductive learning (rules or functions) cann't be applied over datasets.

The causes of overfitting

1) Too many hypothesis for a model for the training data.
2) If the input (training) data is too small (with large trainig data, there is no overfitting)
3) Model favors simple hypothesis (biased hypothesis) and the overfitting occurs
4) When values of hyperparameters are too large or small.
5) High learning rate generaalizes large data (similar to data augmentation), thus minimizes the overfitting
6) Using **regularization** to reduce the complexity of the model. The regularization gives the best regression line that can generalize the large data set thus prevents the overfitting

Variance should dominates the **estimation error**, **not the bias**. We have to tune learning rate, the adjusted weights/capacities, hypermeters, tune the system based on bias vs. variance etc. to avoid overfitting in ANN.

You can use performance estimates (e.g., cross-validation or train-validation-test) to determine the number of epochs correctly. If there are too few hidden layers then it will bring the overfitting problem.

Task 3: In which situations you would recommend Leave-One-Out method for validation of data mining results? (20 points)

Soln:

Background: Leave-one-out cross-validation (LOOCV) is a particular case of leave-p-out cross-validation with p = 1. Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the **function** approximator is trained **on all the data except for one point** and a prediction is made for that point. Leave-one-out cross-validation is less unbiased (means error estimates are also unbiased), because the difference in size between the training set used in each fold and the entire dataset is only a single pattern (or single point).

This brings consistent results. Because of N-1 training sets out of N inputs are used in LOOCV in each iteration, there will be higher variance because there is a lot of overlap between training sets. This makes test error estimates are highly correlated, which means mean of test error estimates have higher variance.

In what situations:

- When we require computationally inexpensive models (e.g., linear regression, nearest-neighbors' classifiers etc.) then we **don't use** LOOCV. Because LOOCV model has to process n-1 training set.

- If test data is an outlier, then the calculated then difference of **y-y**$^{\wedge}$ will be high. It means test error is not consistent. Outliers are any data which has higher variance. We have to tune data to remove outliers and then do cross-validation.

- When dataset is small. Although, LOOCV is computationally expensive, for small dataset, this works. Because for the small number of data sets, the whole computation finishes faster, and also, we can't make many bigger K-folds cross validation training data sets.

- When we need consistent test estimates LOOCV is used. Because there is no randomness in the training and testing data. Doing LOOCV multiple times means the same and consistent result.

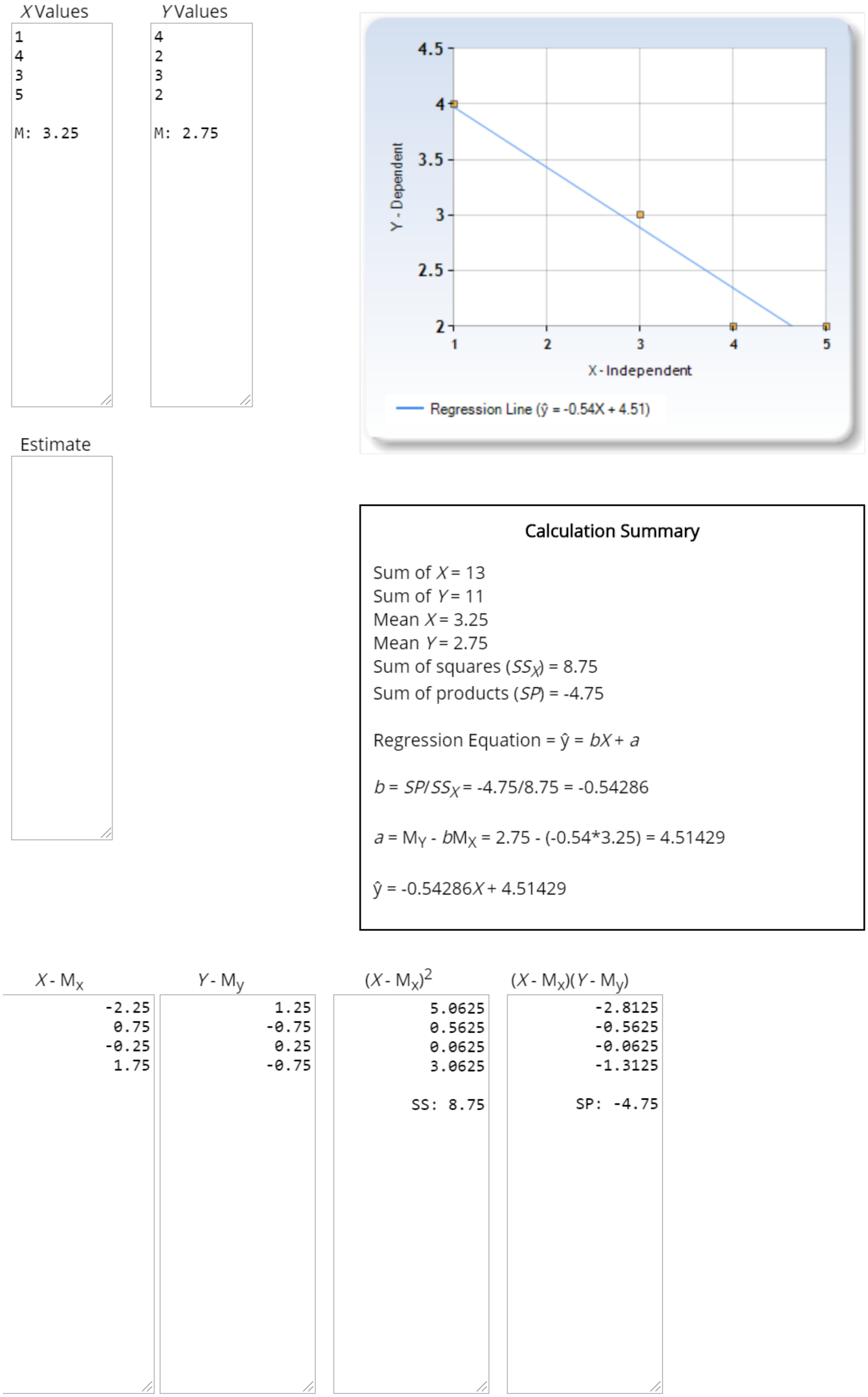
- **Task 4 :Given the data set with two dimensions X and Y:**

X	**Y**
1	**4**
4	**2**
3	**3**
5	**2**

Use a linear regression method to calculate the parameters α *and* β *where* $y = \alpha + \beta x$. (20 points)

Soln:

This is calculated as shown below.



X Values	*Y* Values
1	4
4	2
3	3
5	2
M: 3.25	M: 2.75

Estimate

Calculation Summary

Sum of *X* = 13
Sum of *Y* = 11
Mean *X* = 3.25
Mean *Y* = 2.75
Sum of squares (SS_X) = 8.75
Sum of products (*SP*) = -4.75

Regression Equation = ŷ = *bX* + *a*

$b = SP/SS_X$ = -4.75/8.75 = -0.54286

$a = M_Y - bM_X$ = 2.75 - (-0.54*3.25) = 4.51429

ŷ = -0.54286*X* + 4.51429

$X - M_x$	$Y - M_y$	$(X - M_x)^2$	$(X - M_x)(Y - M_y)$
-2.25	1.25	5.0625	-2.8125
0.75	-0.75	0.5625	-0.5625
-0.25	0.25	0.0625	-0.0625
1.75	-0.75	3.0625	-1.3125
		SS: 8.75	SP: -4.75

Task 5: Support Vector Machines (SVM). The Mercer kernel used to solve the XOR problem is given by polynomial kernel function, $k(x_i, x_j) = (1 + x_i^T x_j)^p$. What is the smallest positive integer p for which the XOR problem is solved? Show the kernel and XOR Problem solution using SVM (20 points)

Soln:

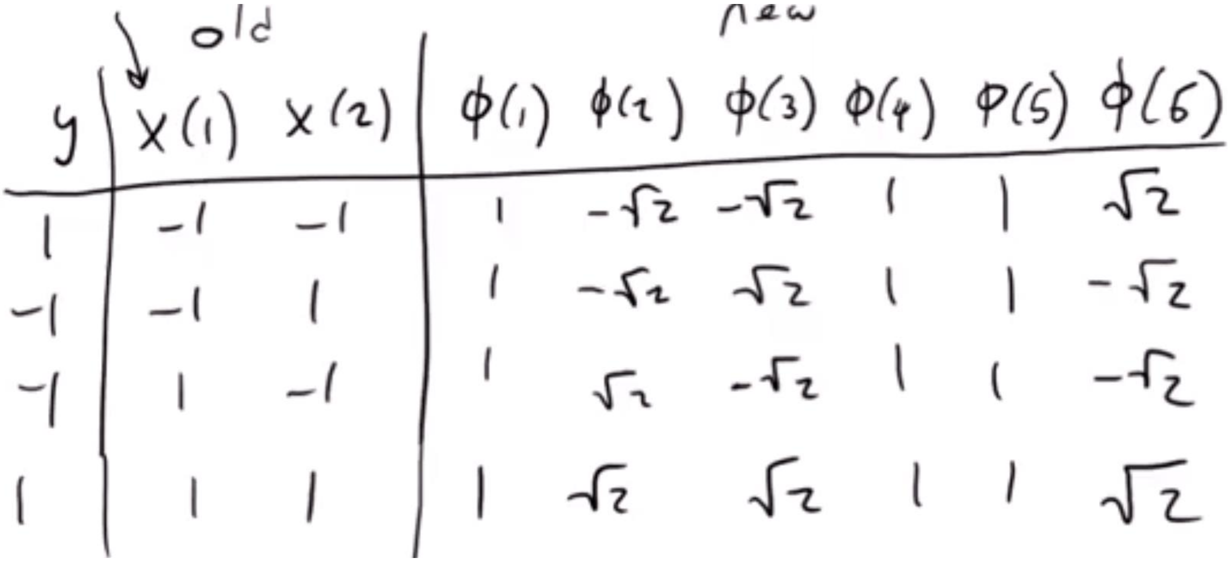
From Wikipedia: In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When p = 2 (quadratic case), the XOR problem can be solved. So, p=2 is the smallest positive integer for the XOR problem is solved. The $k(x_i, x_j) = (1 + x_i^T x_j)^p$ is a polynomial kernel function of order p.

Let's take one example of input sets as (x1, x2): (-1, -1), (-1, 1), (1, -1), and (1,1). This input sets is not 2-d linearly separable. As the defn of linearly separable is there exists a Weight verctor W such that

1) $W^T \Phi > 0$ if $\Phi \rightarrow y=1$
2) $W^T \Phi < 0$ if $\Phi \rightarrow y=-1$

We transform 2D non-linearly separable features verctors to higher dimension but linearly seperable features vectors using kernle function $k(x_i, x_j) = (1 + x_i^T x_j)^p$. This transformation (map) results in 6D features vectors represent by new Φ as shown in below table.



	old		new					
y	x(1)	x(2)	φ(1)	φ(2)	φ(3)	φ(4)	φ(5)	φ(6)
1	-1	-1	1	$-\sqrt{2}$	$-\sqrt{2}$	1	1	$\sqrt{2}$
-1	-1	1	1	$-\sqrt{2}$	$\sqrt{2}$	1	1	$-\sqrt{2}$
-1	1	-1	1	$\sqrt{2}$	$-\sqrt{2}$	1	1	$-\sqrt{2}$
1	1	1	1	$\sqrt{2}$	$\sqrt{2}$	1	1	$\sqrt{2}$

W^T = (w1, w2, w3, w4, w5, w6)

$$i=1: \quad \underline{w}^T \underline{\phi}_i = w_1 - \sqrt{2} w_2 - \sqrt{2} w_3 + w_4 + w_5 + \sqrt{2} w_6 > 0$$

$$i=2: \quad \underline{w}^T \underline{\phi}_2 = w_1 - \sqrt{2} w_2 + \sqrt{2} w_3 + w_4 + w_5 - \sqrt{2} w_6 < 0$$

This way we get six total equations. These system of equations has infinite solutions. This means there are infinite number of weight vectors W that can linearly separate the data. Each system of equations has compared with value 0, this is Mercer's condition.

$$\begin{bmatrix} 1 & -\sqrt{2} & -\sqrt{2} & 1 & 1 & \sqrt{2} \\ -1 & \sqrt{2} & -\sqrt{2} & -1 & -1 & \sqrt{2} \\ -1 & -\sqrt{2} & \sqrt{2} & -1 & -1 & \sqrt{2} \\ 1 & \sqrt{2} & \sqrt{2} & 1 & 1 & \sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

My separability assumption is somethin like if y = f(x) > 0 then it belongs to cluster C1, else C2. Thus makes linearly separable.