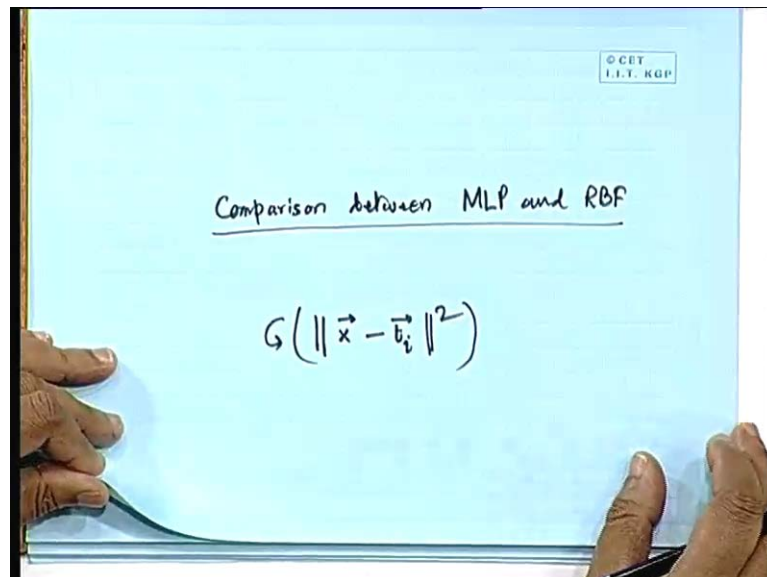


Neural Network and Applications
Prof. S. Sengupta
Department Of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 30
Comparison between MLP and RBF

The title of today's topic is Comparison between the Multilayer Perceptrons and Radial Bases Function networks, which we have discussed over the past few classes.

(Refer Slide Time: 00:57)



Now, this comparison aspect is not going to take much of our time. But, we need to discuss some of the things pertaining to our last discussion only, because towards the end of the last class. We were discussing about the generalized solution to the regularization problem where in fact, we had presented, that in case, that we chose the number of RBF's to be $m < 1$. Instead of choosing it to be equal to N or the number of green's function or the number of RBF's, we chose to be less than that of N .

In that case, basically the G matrix that involves is not symmetric matrix and the G in fact becomes a rectangular matrix of size $m < 1$ by N . And we had to obtain the solution of w 's, in terms of pseudo inverse of that G matrix. So, this is what we discussed in the last class and in fact one thing, which you have noted in our last class discussion is that, the

argument that we are choosing for the green's function, that argument is always the Euclidean norm of the arguments, that what we have been taking.

And Euclidean norm, we are defining as the Euclidean norm of the vector, vector difference between the input and the center. So, if t_i vector happens to be the center of the green's function and if x vector is the input vector, that we are feeding. So, in that case, the norm of the Euclidean norm of x vector minus t_i , where t_i is the i th center to it. And the square norm of this, that is the square Euclidean norm of this, we were considering as an argument to the green's function.

Now, just we deviate slightly, in order to tell you that it is not that the square norm, the Euclidean square norm is always the thing that we are looking for. Euclidean norm is, but we often find that the elements of this x vector. That is x is actually composed of m_0 numbers of elements that, we know. And this m_0 elements, may belong to different patterns, may belong to a different class.

And in such kind of cases, what we need to do is, sometimes we are required to modify the Euclidean norm and what we obtained, after the modification of the Euclidean norm, we call it as a weighted norm. So, we can express the green's function instead of writing the argument as a Euclidean normal. We can write the green's functions argument in the form of such weighted norms. So, let us now see that, what kind of weighted norm, we are looking for.

(Refer Slide Time: 04:40)

Weighted norm.

$$\|\vec{x}\|^2 = \vec{x}^T \vec{x}$$

$$\|\vec{x}\|_C^2 = (\vec{C}\vec{x})^T (\vec{C}\vec{x}) = \vec{x}^T \vec{C}^T \vec{C} \vec{x}$$

\vec{x} : Dimension m_0

\vec{C} : Dimension $m_0 \times m_0$
Matrix : Weighting matrix.

$x_2 \uparrow$ (x_1, x_2)
 $x_1 \rightarrow$

So, we first have some discussions on the weighted norm, now what we define as a weighted norm is that, we define x is the vector that we are considering. And the weighted norm, the weighted squared norm of the vector x , this is also a squared norm, different from Euclidean, weighted and weighted by what. Here, we are making use of a matrix, which is of size $m \times m$, so our x vector is of dimension m ; that we know.

So, we chose a matrix C of dimension $m \times m$, so C is going to be a matrix, which we will be using in the weighted norm expression. And what we do is that, we use this matrix as a weighting matrix, so we will call this, as the weighting matrix. And using this weighting matrix, the weighted squared norm could be expressed as $C^T x$, $C^T x$ transpose times $C x$.

Very similar to the Euclidean norm, because the normal Euclidean norm, what we do, Euclidean norm of x , becomes we know equal to $x^T x$. So, here what we are getting is, that instead of $x^T x$, we are getting $C^T x$ transpose $C x$. So, it is only that, x is modified by application of this C , dimensionally; it remains the same. Because, x is of dimension m and C is of dimension $m \times m$. So, $C x$ also becomes a vector of dimension m only.

So, this is m vector transpose multiplied by m vector, so in a sense that, this again gives us a scalar quantity for this one, in fact we are talking as the weighted norm. In fact, you can look at it, the application of this C , so C is a matrix that modifies x . Very similar to the situation, what we get, when we take the affined transformations of points in the Euclidean space.

What we mean by affined transformation, that supposing in the Euclidean space, say we have got a point, let us say we have got one point, which has got, two element vectors. So, x_1, x_2 , this is the coordinate of that point, where x_1 is this direction and x_2 is this direction. So, x_1, x_2 forms a vector and now, if we rotate the axis by an angle θ or if we scale any application of translation of the axis, translation, rotation or scaling, that together we call as the application of an affined transformation matrix.

So, in effect what you do is that by applying the affined transformation matrix, transforming the point x into another transformed point, the affined transformed point. And here, you are doing a very similar thing, that from the x space, you are going into a $C x$ space, which is essentially some kind of weighted space; that it belongs to and the resulting norm; that we get is the weighted norm.

So, using the weighted norm, actually the generalized form of the equation that we had obtained in the last class that gets modified.

(Refer Slide Time: 08:57)

© C E T
I. I. T. K G P

$$F^*(\vec{x}) = \sum_{i=1}^{m_1} \omega_i G(\|\vec{x} - \vec{t}_i\|_C)$$

For the case of Gaussian

$$G(\|\vec{x} - \vec{t}_i\|_C) = \exp\left[-(\vec{x} - \vec{t}_i)^T \underline{C} \vec{C}^T (\vec{x} - \vec{t}_i)\right]$$

$$= \exp\left[-\frac{1}{2} (\vec{x} - \vec{t}_i)^T \underline{\Sigma}^{-1} (\vec{x} - \vec{t}_i)\right]$$

where, the inverse matrix $\underline{\Sigma}^{-1}$ is defined by

$$\frac{1}{2} \underline{\Sigma}^{-1} = \underline{C}^T \underline{C}$$

and $\underline{\Sigma}$ is the covariance matrix.

So, you remember that we had obtained last time, that F^* star x and why are we calling it as F^* star, because F^* star is different from that of F , because F is the exact solution, F^* is the actual solution for which we actually require N number of green's function where, N is the number of patterns and each of those green's function will be having as it is centre the individual patterns that we are feeding.

But, as we are saying that instead of having of N number of centers we will be restricting our self to m_1 number of centers. So, having m_1 number of centers our solution becomes approximate. So, that is why we are writing F^* star not as F and last time what we had obtained is that F^* star of x was written as summation i is equal to 1 to m_1 , w_i , G , whose argument was the Euclidean norm was x vector minus t_i vector.

Now, instead of writing the Euclidean norm, in this case what we are going to see is, that we will write it with this suffix C . Just like the way, we did here; that means to say, that instead of taking Euclidean norm, we are taking the weighted norm. So, here the argument of the G function will be written in terms of the weighted norm, where actually taking the case of Gaussian function, using Gaussian as a green's function.

So, I think we write down here, that for the case of Gaussian, for the case of Gaussian, we can write G of x vector minus t_i vector, taking it in the sense C , that is taking the weighted norm of x vector minus t_i vector. That will be expressed as what, that is that

will be simply expressed as $\mathbf{x}^T \mathbf{C}^{-1} \mathbf{x}$. In fact, one thing which I should have done over here is that I should have broken up this expression.

This weighted norm expression, if you simply break it up, then what it becomes $\mathbf{x}^T \mathbf{C}^{-1} \mathbf{x}$ and here it becomes, $\mathbf{C}^{-1} \mathbf{x}$, because here the whole thing has been taken as transpose. So, it is $\mathbf{x}^T \mathbf{C}^{-1} \mathbf{x}$, $\mathbf{C}^{-1} \mathbf{x}$, this is what we are getting.

So, in a very similar way, since we are breaking up the weighted norm, which exist in the argument of the Gaussian function, what we will be getting as the argument of the Gaussian function or within the exponential term. What we will be getting is $\mathbf{x} - \mathbf{t}$, then \mathbf{C}^{-1} . Because, instead of \mathbf{x} as we had used over here, here we are getting $\mathbf{x} - \mathbf{t}$, where \mathbf{t} is the centre of the Gaussian function.

So, this is $\mathbf{x} - \mathbf{t}$, \mathbf{C}^{-1} and then what, tell me, do not keep quiet, \mathbf{C} and then $\mathbf{x} - \mathbf{t}$ that is correct, so $\mathbf{x} - \mathbf{t}$. So, this becomes the argument of the exponential and in fact, we can write down this entire thing as by some representation, we can write it as $\frac{1}{2} (\mathbf{x} - \mathbf{t})^T \Sigma^{-1} (\mathbf{x} - \mathbf{t})$. And we are introducing a new matrix, which we are calling as a capital sigma matrix, so this is a matrix.

And we are taking the inverse of sigma matrix, so I will be defining, what this means and this term. So, this term remains as this and this term that is $\mathbf{x} - \mathbf{t}$ vector also remains as it is. In between, what we have done is that, instead of writing it as \mathbf{C}^{-1} , we are writing it as half of sigma inverse, where actually the significance of this sigma is that, it is the covariance matrix.

So, where in this case, the inverse matrix, the inverse matrix that we obtain, the inverse matrix sigma inverse is defined by, it is just the waiting, so it is half sigma inverse. What we have written out here that is a substitution for \mathbf{C}^{-1} , what we had in the earlier step. So, and in this case, sigma, the original sigma matrix is the covariance matrix.

Now, we will be, there is nothing new that, we talked off; we are continuing the same old discussion of the last class only. Only thing is that, our idea was to show that, yes, Euclidean norm is not always the restriction, which you have to follow. The argument of the green's function could be a weighted norm and in terms of the weighted norm, things can be express like for the case of Gaussian.

We could be expressing the weighted norm form like this, were this could be expressed in terms of sigma inverse, where sigma inverse is $C^T C$ and sigma happens to be the covariance matrix. Anywhere, we will be, the reason why, I introduced this point is, because we will be need this thing, shortly. Because, the next topic, which I am going to discuss is some amount of comparison between the multilayered perceptron and the radial bases function networks.

Now, we know that both radial bases function as well as multilayer perceptron, they are essentially feed forward network, they are multilayer feed forward network. One, we are calling as MLP multilayered perceptron, were we know that it is having a definite structures. Where, we know that instead of, we are not restricted to one hidden layer, we could be having multiple hidden layers, but the neurons there look very similar.

Whereas, we are having the radial bases function networks, where we are restricting ourselves to only one hidden layer, but the beauty there is that the hidden layers composition looks remarkably for end. From those of the neurons, which exist in the output layer, because the neurons in the output layer of an RBF network looks as good as old neurons. We were simply, linearly adding those things up, linearly adding the responses up.

So, in output, we did not have any thing revolutionary concept, whereas in the hidden layer activations, the hidden layer activations were remarkably different from the kind of activations that we have used in the case of multilayered perceptrons. Because, for the case of multilayered perceptron, what we had used was that, the activation was essentially based upon the dot product between the weight matrix, the weight vector and the input vector.

So, we were having $w^T x$, which was deciding our activation, in this case what decides our activation is x vector minus t_i vector, t_i being the centre of the function. So, the computational mode is remarkably different, but one thing can be said always, that if for solving a problem, you have got some problem to solve. And there, you have decided to use a multilayered perceptron, following the back propagation model.

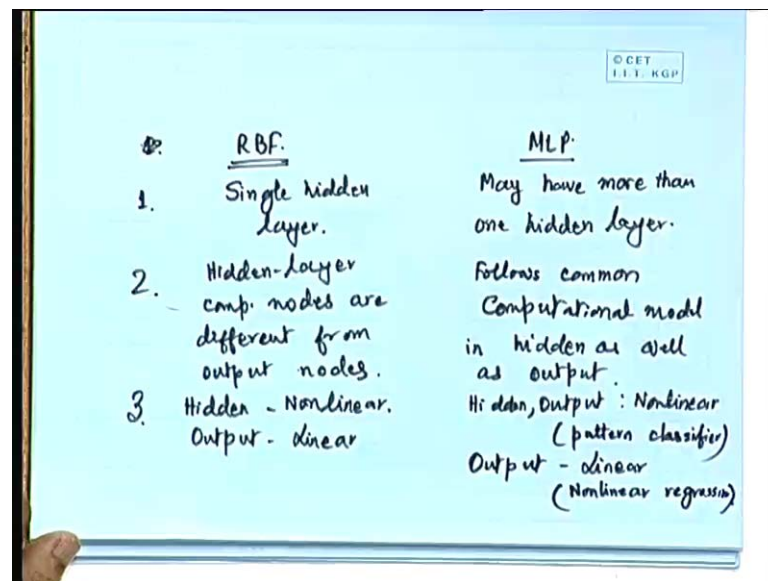
Now, whatever multilayered perceptron can do, the same mimicking can be done using the radial bases function network also. And it is vice versa; that means to say that if somebody is using a radial bases function network to solve a problem, the same problem can be mimicked using a multilayered perceptron. So, this mimicking ability is there;

that means to say that, they have got a good deal of similarity in there structures and everything.

It looks quite similar, both are essentially non-linear mapping, because we are even the multilayered perceptrons starting point was also that we have patterns, which are non-linearly separable in the input space. And we could realize that, using the hidden layer, where in the hidden layer, we had separated out the different linearly partitionable spaces and then added them together in the subsequent output layers.

Whereas, in this case, what we are doing is that, we have just taken a different approach and we looked upon the whole problem as a function approximation problem or to say in a more modest way the surface pitting problem. So, coming to the comparison of the radial bases function network and the multilayered perceptron there are some points that can come to our mind.

(Refer Slide Time: 18:59)



Number 1 is that, when you talk of RBF, I think, we can list out the comparisons like this, we have RBF, this side an MLP this side. RBF is having single hidden layer, that is in it is very basic form, not that people have not experimented with multilayered RBF. But, really speaking multilayered RBF does not serve much of a purpose, because this side of a mapping is possible. It is possible to realize this mapping using single layer only.

Only thing is that, yes, you decide that, how many numbers of RBF's you use as the hidden layer. And what you chose as the centers, those are the some of the very crucial

decisions, but in its basic forms single hidden layer is good enough. Whereas, MLP can have it may have more than one hidden layer. Then, the second point is that, the RBF has got hidden layer computation nodes, different from that of it is an output nodes.

So, hidden layer computation nodes are different from output a computational node that is very much obvious, because in hidden layer we are using the RBF activation functions. Whereas, in the outputs, we are using the linear activations and in the case of multilayered perceptrons, we are using, it follows a common computational model. So, it follows common computation model in hidden as well as output.

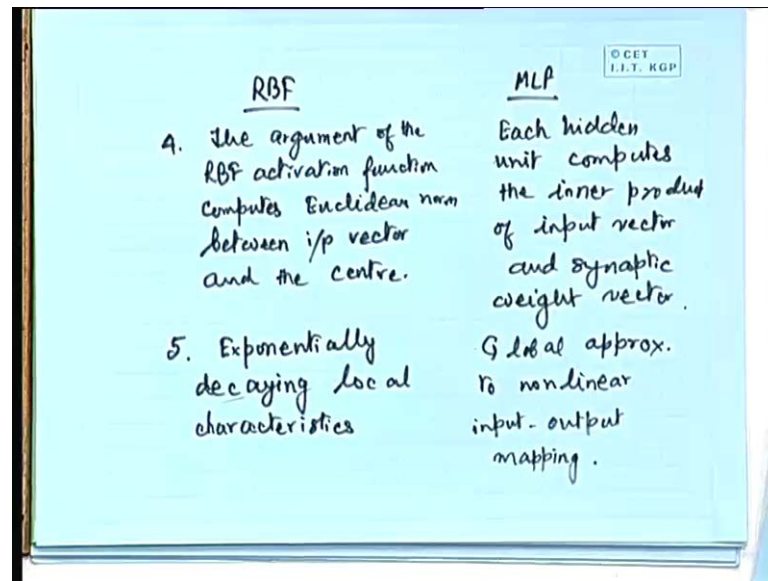
The third point, about the comparison is that, in the case of RBF, the hidden layer is non-linear. And output is what, linear, output is linear, we had seen that, because whatever non-linear mapping have to be done, that was done in the RBF layer itself, in the hidden layer itself we did the non-linear mapping. And then, once the non-linear mapping as been carried out, the rest could be linear.

Whereas, in the case of multilayered perceptron, what happens hidden is non-linear output also non-linear, we are using the sigmoidal activation functions typically, so hidden and output, both are non-linear. Although, in the non-linear regression, so this is actually true, when we are using the MLP as pattern classifier, we will be using both hidden layer and output layer as non-linear.

Whereas, for the case of non-linear regressions, because you know, MLP can be used for pattern classification problems, where the purpose is to just divide that, the patterns into a number of known classes. And the other way of looking at it is that, where we solve a regression analysis problem, more of a generalization problem, where we can use the output layer, there the output layer could be linear.

In the case of non linear regression, so these are up to the third points, so these three points are quite obvious. And now the fourth point says, about the argument of the functions, which we have already talked off.

(Refer Slide Time: 23:49)



The argument of the, so here RBF and here MLP, the argument of the activating function of the RBF activation functions, computes the Euclidean norm or weighed norm whatever between input vector in the centre. Whereas, in the case of multilayered perceptron, each hidden unit computes the inner product, inner product of what, input vector and synaptic weights.

The last point is that, in that RBF can we call it as a local or is it a global function, it is definitely a local function, because we had see, that we are employing the centers for those RBF's. And definitely, the response depends upon those centers and as you go away from the center, the response decreases. So, it is actually an exponentially decaying local characteristics.

Whereas, as in the case of multilayered perceptron, it is doing global approximation to non-linear input, output mapping, so I think this points are pretty obvious. In fact, here, one can always say that structurally, although capability wise, both RBF and MLP can do the same thing. That is to say, solving non-linearly separable problems, both RBF and MLP's are capable of doing that.

RBF looks little complicated in the sense that, the type of functions that, it is using is not as simple as the kind of functions, that we are used in the multilayered perceptron, because the computation of the sigmoidal functions are pretty simple. In fact, hardware units are also available, nowadays, where this sigmoidal can be computed in hardware itself.

So, that way, computationally, yes, computing so many different RBF functions is could be the only involved thing, but otherwise, structurally RBF looks pretty simple. And in fact, one of the points, which we should note at this stage, is that coming to the training of the network. In the case of multilayered perceptron, we had seen that the training is quite involved in the sense that, we are feeding the inputs.

And then, we are getting the outputs, the actual outputs different from that of the desired outputs. And then, the error that we are having, that error is in turn guiding us to adjust the synaptic weights from the output layer back up to the input layer. So, we proceed from back to the front and using a back propagation of the error, we were making it learn.

So, we had to adjust so many different synaptic weights, but coming to the training, if one asks us, about the training of an RBF, well where is the training involved. As far as the synaptic weights are concerned, the only place where we are having synaptic weights is where we combine the RBF outputs into the output. All the RBF outputs are taken together and then, we are deciding upon an output and there only we are having synaptic weights.

So, learning is only that much, that what we thing or we may right in saying, that only those are the synaptic weights. But, coming to the point, is it only learning that is involved in the RBF, what the answer, yes or no. Do you think that, only learning the weights of those RBF output to the actual output. Only leaning those weights are enough for a learning mechanism in RBF, many people said that the answer is no and why and then, let me invite that discussing also. Why do you think that, there is some more learning that is involved?

Student: ((Refer Time: 30:13))

For each node, the radial bases function ((Refer Time: 30:21)), but one thing that, we are choosing a particular type of radial bases function let us say Gaussian. So, once we chose Gaussian, then all the nodes are at least Gaussian functions, the sentence are varying. Now, the kind of discussions, which we had so far, there we have telling that, as if to say that, we have decided a priory, that there are $m + 1$ number of different radial bases functions existing.

And we are feeding the input vector and the input vector will be going to those RBF computational nodes, which will be computing the Euclidean norm and then, finding out responses of all this different RBF's. So, as if to say that a fixed structure of RBF's, if you chose, then actually speaking there is no learning that is involved. But, if I ask you one think, that who says, that my choices of centers are the best.

You see, as long as the problem is that, I am having N number of different patterns and I chose N different centers, each centre corresponding to the patterns, which are actually existing in our training space. Well and fine, it is understood, that we are used the training patterns, themselves as the pattern centers and we have N such training patterns, we constitute that N such centers.

But, the question comes that, when we are saying that N , the number of patterns is too large a number to construct a radial bases function network and we have decided to reduce the dimensionality from N to $m - 1$. Then, the immediate next question is that, how would you chose this $m - 1$ out of N . And even, if you happen to chose $m - 1$ out of N , some arbitrary way, some random way where is the guarantee, that this is the best type of an arrangement or could there be some adaptability, that depending upon the patterns, which actually exist in the input space. We may be prepared ourselves to adoptively adjust the centers from one iteration to the other? If we adoptively adjust the centers of those radial bases functions, in that case, you see, we are bringing in another degree of learning mechanism in the radial bases function network.

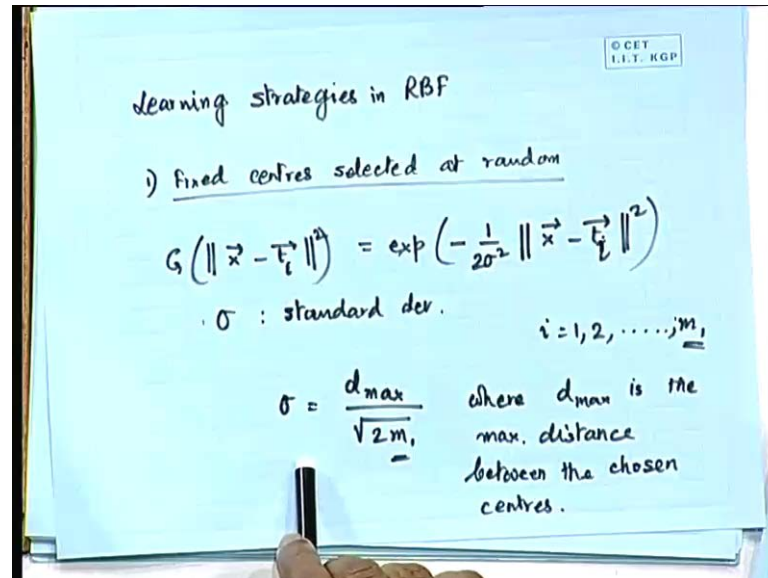
It learns what, it learns the positions of those centers, we can start with some arbitrary positions of the radial bases function centers, make that network learn. And then, we can adoptively adjust the positions of their centers in accordance with whatever input vector that we are actually feeding to the system. So, if we do like that, then the learning in the RBF is to be looked upon as a two step strategy.

Number 1 is the learning, that we in cooperate in the RBF functions itself, choosing the centers of the functions, where the first degree of learning is involved. And only, when this learning is over, then we can afford to go in for the second level of learning, which is the learning of the synaptic weights from the hidden layer to the output layer. So, learning as to be looked upon as a two step strategy in the case of RBF.

So, I thought that, before we introduce the next topic in our program that is the principle component analysis, which I said. In fact in the last class, that I will introduce that, but I

thought that, may be some discussions are very much needed to give you treatment of the learning mechanism in the RBF's. So, that, what I intent to cover in the remaining part of today's lecture.

(Refer Slide Time: 35:14)



So, let me tell you about, in terms of the leaning strategies in RBF, we can list out several learning strategies. The first one is where we have fixed centers selected at random. So, we can choose the centers, randomly, if we adopt this strategy fixed centers, selected at random. So, there what we do is that, we chose the function G of as argument, we take x minus t i vector this squared. It can be expressed as, if we happen to choose the Gaussian as the radial bases as function.

In that case, the argument of this exponential will be minus sigma square, that what minus one upon sigma square times; we can have this, that, it will be one upon sigma square or one upon two sigma square. That is right, one upon two sigma square into norm of x minus t i square, is it correct. So, were sigma happens to be the standard deviation of the Gaussian function.

And in fact, so this is the standard deviation and how many such t i's are we choosing, let us say that we are choosing m 1 such centers. So, we choose i is equal 1, 2 up to m 1. So, what we do is that, we have to determine the standard deviation, this is very important, because what happens is that, if we chose the standard deviating to be too small; that means to what, we are choosing very narrow Gaussian functions.

And very narrow Gaussian function will be having a disadvantage, that it will be, highly local in nature, highly local in nature mean response will be only centered around that and as we deviate from the centre, the response of that will be very poor. So, there will be a number of patterns, which will not be getting any response. If we consider the total input space, then in the input space, there can be existence of a number of patterns, which will be having small response.

So, that why sigma should not be too small, again the problem of sigma choice as very large, if we chose sigma to be very large, in that case, we are avoiding that problem, no doubt. But, then, we are inviting another problem, that is large sigma means what, that we are not choosing this t i alone, we are choosing m - 1 such t i's. So, there will be m - 1 such different center and the responses of those RBF's will be overlapping with each other.

So, there are two things in it is, so that why the sigma is to be very judiciously chosen. In fact, one of the norms, which people have followed in choosing this sigma is that, they decided to chose sigma as $d_{max} / \sqrt{2(m-1)}$, where d_{max} happens to be the maximum distance, between the chosen centers.

(Refer Slide Time: 40:28)

OCET
I.I.T. KGP

$$G(\|\vec{x} - \vec{t}_i\|^2) = \exp\left(-\frac{m_i}{d_{max}^2} \|\vec{x} - \vec{t}_i\|^2\right)$$

$$\vec{w} = \vec{G}^T \vec{d}$$

$$G = \{g_{ji}\}$$

where,

$$g_{ji} = \exp\left(-\frac{m_i}{d^2} \|\vec{x}_j - \vec{t}_i\|^2\right)$$

$j=1,2,\dots,N$
 $i=1,2,\dots,m$

So, if we do that, if we apply this definition of sigma, then the Gaussian function that we use, can be expressed as G of norm x minus t_i whole square is equal to exponential to the power minus $m - 1$ by d_{max} square. Euclidean normal of x minus t_i whole square $m - 1$ of s t is the total number of such centers, that you are using. So, here what, so that means

to say that, we are choosing uniform sigma's, for all the centers, if we chose the sigma's like this.

Then, for all the centers, for all the Gaussian functions that we are using as RBF's, we are having the same amount of standard deviations. So, that means to say that, they are equally spaced, but one thing, which one should note is that, when we have a pattern space, when in the input space, we have got some training patterns. We may find that, the patterns are very much clustered in some areas and the patterns are quite sparse in some other areas, it can happen.

Patterns, quite sparsely populated in some regions, in the m dimensional spaces only. Some regions, it is spares, some region, it is less, but if we happen to choose the centers, fixed centers and fixed locations, uniformly apart. In that case, the disadvantage that we are getting is that, there will be some centers, which we are keeping unnecessarily, because within it is region of activation, there is not many patterns to accept from.

So, there responses will be poor, so we are just waisting as if to say, waisting that RBF node, where the population is not dense, near to it is center. Whereas, in some regions, we may be having large population of patterns, but there may be just one center. Where, we may find difficult to actually discriminate between the patterns, because pattern classification is our main motto of using this RBF. So, where we will be unable to distinguish between the patterns, where the patterns are highly clustered and close.

So in fact, this it suggests, that instead of choosing fixed centers, we should be allowed to play with the centers where, if you find that somewhere, the region of the, concentration of the input patterns is more shift more centers towards that, try to design the RBF with having centers in those populated areas. And having less number of RBF's in unpopulated or sparse areas something like that you can design. So, this it suggest that, there is a scope of adaptability in the space, in the centre allocation.

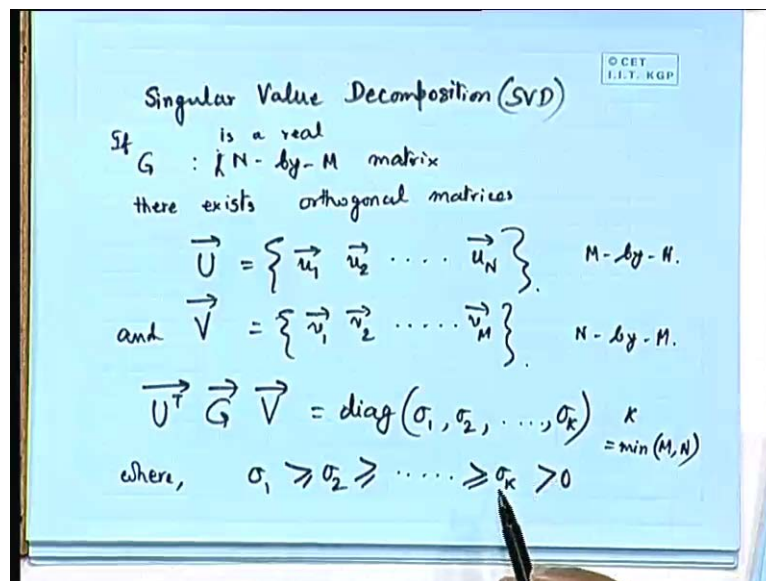
In fact, one thing that leads to is that, if you used fixed centers, for example that, we use fixed centers, assuming that the input patterns are uniformly distributed in the m dimensional space and we can afford to use fix centers. Then, the only learning for fix center case the only learning, that is involved is the adjustment of the synoptic weights. And synoptic weights, also we know, because last time in the last class, we had obtained a solution of the weight vector. Remember, we had obtained it as G plus, which by G plus is the pseudo inverse of the G matrix d vector where, d vector is the desired

response vector and G plus is the pseudo inverse of matrix G . And our matrix G 's definition was that, G was consisting of the elements g_{ji} where, in the case of Gaussian functions used as RBF's, g_{ji} was expressed as exponential to the power minus $m-1$ by d square, norm of x_j minus t_i square of that. This was our g_{ji} , where j varies from 1 to N , that is correct, why N , because N is the total number of input patterns.

So, x is index is i , so it corresponds to, how many patterns are we having, so j can go up to N and what is the index of i , that is very correct 1 to $m-1$, because there can be $m-1$ number of centers. So, i is the index of the center and j is the index of the pattern, j vary from 1 to N and i varying from 1 to $m-1$. So, now the whole problem for us in the computation of this synoptic weight is computation of this g inverse or computation of this pseudo inverse of g .

In fact, pseudo inverse computation, good techniques are as available to computerized G plus. In fact, very fast and efficient algorithms, computer algorithms are available, were by one can obtain the pseudo inverse of matrixes.

(Refer Slide Time: 46:29)



Now, just one very popular technique of that is by the method of singular value decomposition. Those who are having good background about matrix algebra, may be already knowing that, but for the benefit of all. Let me, give you some idea about this singular value decomposition. Actually, in the case of singular value decomposition, what we say is that, G our matrix, G that we are taking is actually and N by M matrix, because N is the number of patterns, M is the number of the RBF's that we are using.

So, it is N by M matrix, we were taking $m-1$ that by taken m and for a G , there exist two orthogonal matrix. So, if G is a real N by M matrix, then there exist two orthogonal matrices, which we are defining as a U matrix. U matrix in fact is consisting of N number of vectors u_1, u_2 up to u_N . So, we have a matrix consisting of N number of vectors and another matrix V , which is consisting of M number of vectors. So, this is v_1, v_2 up to v_M .

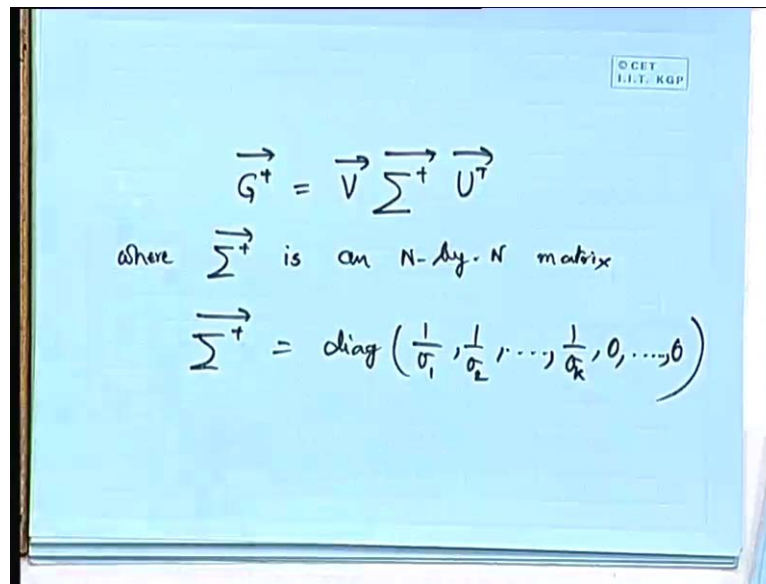
In fact, we can say that all this individual vectors u_1, u_2 all these vectors are of dimensionality m . So, U is actually an M by N matrix, whereas V , here there are M number of vectors, defined over here, where each of this vector is of dimensionality N . So, V is actually of dimensionality N by M , so this is this is to say M by N , whereas V is N by M .

So in fact, why are saying this U matrix and V matrix to be orthogonal, because it is possible to find this U matrix and V matrix. Such that U transposes G , V , this actually diagonalizes this G , by the pre multiplication by U transpose and the post multiplication by V , makes the diagonalization of G . And in fact, the diagonals that results, those diagonals are refer to as singular value.

So, that is why, by choice of this U matrix and V matrix, what we are essentially obtaining is singular value decomposition, so this is to be diagonalized. So, we get a diagonal matrix, whose elements, whose diagonal elements are σ_1, σ_2 up to σ_K , where, k is actually the minimum of M and N . And here the diagonalization will be such that, σ_1 is greater than or equal to σ_2 is greater than this will be ordered up to σ_K and this should be greater than 0.

In fact, all these σ 's are called as the singular values and this kind of technique, we are referring to as the singular value decomposition, where we are decomposing this G into such kind of singular values. Now, if G can be decomposed into singular values by findings this two matrices U and V , then it is possible us to get back the G pseudo inverse, the G plus matrix, in fact we can obtain by applying a transformation like this.

(Refer Slide Time: 51:04)


$$\vec{G}^+ = \vec{V} \vec{\Sigma}^+ \vec{U}^T$$

where $\vec{\Sigma}^+$ is an N -by- N matrix

$$\vec{\Sigma}^+ = \text{diag} \left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \dots, \frac{1}{\sigma_k}, 0, \dots, 0 \right)$$

So, G plus matrix can be recovered as V sigma plus, U transpose, where sigma plus matrix is an N by N matrix and sigma plus is actually defined as the diagonal of 1 by sigma 1 , 1 by sigma 2 up to 1 by sigma K and rest of the elements in that are 0 's. So, if we can form, since we are to the singular value decompositions, we have got sigma 1 , 2 sigma K 's.

We can now form a diagonal matrix with the elements of the 1 upon of sigma 1 , 1 upon sigma 2 up to 1 upon sigma K . And then, K is minimum of N and M , where as we have to go up to N , so if it is M , then we have to fill up there rest of the elements with 0 . So, this becomes the sigma plus matrix and making this sigma plus matrix and because we know the V matrix of the U^T , we can at the G inverse. So, this is the pseudo inverse computation, so pseudo inverse computation good techniques are available.

Now, this is the about the fix center case, where we find that the only learning, that involves in the output layer where, we have to learn the synoptic weights, in fact synoptic weights, there is direct solution. Because, if we are G plus, in that case, synaptic weights can be computed directly. In fact, there is no learning that no iterative kind of the learning, which is involved over here.

You simply compute G plus, you already have the d and you compute the w straight away. But, the next strategy is actually selection of self organized centers, where we are going to tell about some kind of learning mechanism in the adjustment of the centers,

that we will be talking in the next class. So, where the learning mechanism of RBF, we will continue for some time, before we introduce the next topic.

Thank you very much.