[Music]

Welcome to lecture seven of the six week of the course in machine learning and in this lecture we will talk about some examples of associative memory hopfield networks and Boltzmann machine and a few other approaches. As you can see from the map of material this week we are now here so the coming lecture will then after this we will turn to something called convolutional neural networks before we finish this week. The Hopfield network is one approach to the realization of associative memory. It is an instance of an the subcategory of Association memory called a auto association memory which if you remember is a situation where you look at certain objects you have a partial description of that kind of object you are interested in, what you do is you as like in a query fashion good in this partial description and the system is enable to provide you through the call the full description. So in a way this kind of system completes primarily completes partial descriptions. It's not likely to give you anything more, anything wider, anything outside the actual domain. So hopefully that work is also considered to be a recurrent neural network even if it's not able to handle temporal sequences in the sense that in every iteration, in every training for a hopfield network you in a way define statically the inputs to the networks based on a particular data item. In RNN you're able to have a sequence of items that constitute one single date item and being able to handle that in sequence, this is not the case with Hopfield network. However in Hopfield network has states and it's also having cycles in the network so because of that it's counted as a recurred neural network. However it's more natural I would say two primary classify of your networks within the class of associative memories. It's a one layer neural network in the sense that all units are input/output, so there are no hidden units and there is no distinction actually between input and output. So you input values and then you calculate new values which becomes the output. It's fully connected which means that all nodes all neurons here neurons are connected with each other and the ways are symmetric, so going from A to B has one weight and going from B to A has the same weight. The units are modeled inspired by the Mcculloch and Pitt's neural model which is very simple with output values minus one and plus one. There are two modes so either you can update if the state synchronously at the same time or asynchronously in some sequences. The weight updating is inspired by the Hebbian learning so it's clearly in the tradition of associative networks. It also has an energy concept which is very important and that ensures the convergence towards the stationary state, and so primarily when you search for stationary state you use the value of the energy variable as your

guideline and the models enables fixpoint stable attractors, which means which corresponds actually to the local minima. And hopfield networks were invented by Hopfield in 1982.

Let us now look at the updating of a unit in a Hopfield network. So in a Hopfield network first of all we have what we call one could call one-shot learning, that we in a way decide how many paid items we would consider, let's say N of those which corresponds to one can say with a memory terminology how many memories do we want to store in our associative memory. And having decided that we will construct a matrix X with these data item vectors as rows and then the weight matrix will be the transposition of that matrix multiplied by the matrix itself divided by N. So I can say it's the normalized value of this matrix multiplication. Okay so that's the kind of storage part and then we will look at a particular test case and in that test case, based on the test case we will calculate a new value for each of the states and that is done according to the formula but the new value of the stage is plus one, is the sum of all the inputs states from the other nodes in the network times the weight on the connections from those. If that's bigger than the threshold then we got a plus one otherwise we get a minus one. So the output series in the traditional McCollum some pits (07:38) And so the values of neurons Y i and j will converge if the weight between them is positive similar they will diverge is the way it is negative. So in that sense we can say that this way of doing it is in the Hebbian tradition. So finally another aspect of this is when you do the updates you can do them in two different ways, you can do it in asynchronous way where you update one unit at a time and then you can pick them at random or you can have a specific order and actually. And then there is the synchronous mode where all units are updated at the same time and it somehow requires a central clock to the system in order to maintain, so some people thinks this is less realistic but it because in nature and the corresponding systems in humans and animals and so on, there is actually no global clock of that kind but actually we are building a diffusion system, so maybe that's not a strong criticism.

Hopfield networks has one very important feature and that's the energy value. So it's this is a scalar value associated with each state of the network and that is defined on this slide so you can see it's the total sum of all combinations of all states and all the weights between them, divided by minus 2, plus the sum of the states multiplied by the threshold. So I will not prove to the inference of this formula, I will simply say that the setup of this formula called energy is that it should ensure that this formula always will decrease over time the through the updates. So

repeated updating will eventually converge to a state which is a local minimum in an energy function. So if the state is a local minimum for the any functions is a stable state for the network. So actually what you can possibly do is that you build a practical hopfield network system and then you can store a number of cases defining the network and then when you for every test case you test, you can look at the energy value and then you can see how the energy value behaves and of course the idea is that you want to minimize the energy value, the lower the energy value the closer you are to a stable state.

Let us now quickly look at a few graphical examples. So first here you see in two neuron hopfield network characterized on by two stable states, you can see an graphical descriptions of the Basins for and the attractors for those stable states. If we move to three states you can also see some graphical depiction of that and you can see in this particular case given the details it turns out that this example have two stable states and finally if we move upwards it's not so easily to depict you can have a fine neuron hopfield network with a more complex weight matrix but of course you can also imagine a landscape actually defined by the error function in this case. So of course the more complex the network is the more valuable the energy function becomes because it's a simple scalar measure calculated from the system. So even if the system is complex to view this color the energy value will always tell you something.

Let us now look at was more practical example. So we have a small board with four items on the board or places on the board, we call them one two three four and so there are two stable and so for each position there is either minus 1 or plus 1. So the orange or yellow here are positive and the red one is negative and so if all positions on the board are in one of these state we can say it's stable because those are the states that can be generated actually by the updating function. Of course we can introduce externally other states and these are depicted us as blue here, so you can see down here that these items are blue which means that they are neither minus 1 or plus 1. So in this example if you can depict in a various ways you can depict the graph of course it's very important to understand what the weights are and has been said earlier to the weight updating in one shot so you look at the three patterns we have seen patterns about as you see and actually the in the Hebbian tradition and the updating is based of the average correlations across the three patterns. So if you locate for p1 and in in the column below, you can see that if you compare state 1 we stayed two it's the same so it's a postitive, so you get a 1, if you compare it with the

element 3 it's also positive but if you compare it with element four, it's the opposite so that for it's negative. So this kind of procedure you can go through for all these combinations of the first pattern and now you do the same for the second pattern now you do the same third pattern and then you get for each weight and you get a row and then you take the average of the elements of the figures for the various pattern and then you get a column with averages and actually these average are the initial weights we will use. One can also say that we'll see that soon a weight matrix and that way it works is because it's a definition hopfield network at that everything is symmetric, everything is connected, everything is symmetric, so this means and that that matrix need to be mirrored in the diagonal and them and the figures we see here in this column will constitute one of the house and the other half of the matrix will be the same and it's not rather interesting what's in the diagonal bit because it's actually the relation from a node to node and according to the model hopfield model there is no right cycle it's only in direct cycles in a hopfield model. So this means that we can then calculate constructors in the matrix as I described and then we can have a test case, so this means that we multiply this matrix with this vector and when we do that we get the resulting matrix but the result according to the formula is not just the multiplication of that matrix with the test case because it's also have to go through a threshold function and in this case we have some make it simple and have the threshold 0. So doing that threshold operation we will end up in a new value of one or minus one, that can then be depicted as you see below with the stable state consisting of three positive and one negative in the bottom right corner.

If we repeat the same calculation for the same example using the matrix notation we originally introduced you can see that if we form a matrix with the original with the input vectors we considered as rows, if we transpose that vector then we get actually our weight vector by multiplying the transpose vector by the vector and dividing by N. So you can see here is that why we form the matrix, the transpose matrix, we multiply them and we divide by N so we end up with matrix to the to the right, and then we can use that matrix, so essentially when we want to calculate an updated state we simply can multiply the new test case with that matrix to get the result, by that we got a more schematic better scheme to easily calculate new states based on the weight matrix.

Finally you can see here the alternative way of updating so in last in case we treated more in detail we looked at this asynchronous case where all the rows were updated in in parallel of his providing that there were some mechanisms of synchronized time. So here you cannot see an example of the synchronous case where we have to take it stepwise. So first we update one now we look at the result to be obtained another and so on and eventually that also will lead in the bottom right to a stable State. So by this we take a little break and come back in part two in another video to the same thing thank you