Maharishi University of Management

FINAL SOLN

Course Title and Code: *CS 582 - Machine Learning*
Instructor: *Dr. Emdad Khan*
Date: *Thursday 05/24/18*
Duration: *10am – 12.00 pm*

Student Name:

Student ID:

Total Mark
– – – –
30 points

1. This is a closed book exam. Do not use any notes or books!
2. Show your work. Partial credit will be given. Grading will be based on correctness, clarity and neatness.
3. We suggest you to read the whole exam before beginning to work on any problem.
4. There are 4 questions worth a total of 30 points, plus a Bonus question worth 3 on 9 pages (including this one)
5. You can use all back pages for scratch paper (may use for answers if you have used up the designated space for answer).

Question 1: 8 points (2 + 2 + 2 + 2)

a. What is the purpose of the neighborhood function in the SOM? How does it change the learning mechanism / strategy?

Ans. Neighborhood function is used to maintain topography. The training is changed to include a learning for the neighborhood neurons using a neighborhood function with different learning rate. Neighborhood function is adjusted at every epoch.

Learning strategy is changed as in addition to adjusting regular weights to learn the inputs (i.e. to form clusters), the neighboring neurons also do some learning to preserve the topography i.e. to maintain the spatial feature (nearby neurons correspond to similar input patterns: known as feature mapping – nearby neurons correspond to similar input patterns).

b. What is the learning equation for Competitive learning in k-means Neural Network? Why normalization of weights is needed in Competitive learning? Clearly explain your answer.

Ans. The learning equation for competitive learning is

Delta Wij = eta (Xj – Wij).

So, the network basically learns weights as input vectors i.e. for a winning neuron input vector matches the weights. This means that these two variables must have same range. This is why we use weight space when we use competitive learning. To ensure that weights are in the same range as inputs, we need to normalize them using a Unit Hyper Sphere so that weight values are less than or equal to unity.
In case a normalization was not done, some weights might have very high value and a neuron other than the winning neuron may get the maximum output which would be wrong. Hence we a need a normalization. The following example makes this more clear:

[OPTIONAL]
Suppose that the weights of all the neurons are small (maybe less than 1) except for those to one particular neuron. We'll make those weights be 10 for the example. If an input vector with values (0.2, 0.2, −0.1) is presented, and it happens to be an exact match for one of the neurons, then the activation of that neuron will be 0.2 × 0.2 + 0.2 × 0.2 + −0.1 × −0.1 = 0.09. The other neurons are not perfect matches, so their activations should all be less. However, consider the neuron with large weights. Its activation will be 10 × 0.2 + 10 × 0.2 + 10 × −0.1 = 3, and so it will be the winner. Thus, we can only compare activations if we know that the weights for all of the neurons are the same size. We do this by insisting that the weight vector is normalized so that the distance between the vector and the origin (the point (0, 0,… 0)) is one. This means that all of the neurons are positioned on the unit hypersphere, which

we described in Section 2.1.2 when we talked about the curse of dimensionality: it is the set of all points that are distance one from the origin, so it is a circle in 2D, a sphere in 3D (as shown in Figure 14.4), and a hypersphere in higher dimensions.

c. Briefly explain why $h_{SLD}(n)$ = straight-line distance based heuristic search is not optimal but A* search is optimal.

Ans. $h_{SLD}(n)$ just uses straight-line distance i.e. sort of just exploratory in a greedy way. But A* uses an h(n) and g(n), where g(n) is the cost to reach the node n and h(n) is the estimated cost of the cheapest path from n to the goal. Thus as A* advances, the total cost until the current node becomes optimal. Thus, when A* reaches the goal, the total cost become optimal and complete. It uses both exploitation and exploration. $h_{SLD}(n)$ is a greedy approach and hence does not guarantee an optimal result.

d. What does the initial population represent in GA? What does the final population represent? How a solution is found in GA?

Ans.

The initial population represent a space of possible initial solutions. The final population represent a space of potential answers. A solution is found by doing a search using Genetic algorithm i.e. selection using fitness, mating to reproduce, using fitness function to make the new population (i.e. find more fit population) and repeating the process to find a population with most fit.

Question 2: 7 points (2 + 1 +4)

a. Consider an Agent that needs to recognize 5 unique shapes using a binary image of 12 x 12 pixel. How many different states will there be from 12 x 12 pixels? How functions there will be to recognize 5 shapes from the image?

Ans. There will be $2^{(12 \times 12)} = 2^{144}$ states using binary values for each pixel.

Since we need to recognize 5 unique shapes, we would need $5^{2^{\wedge}144}$

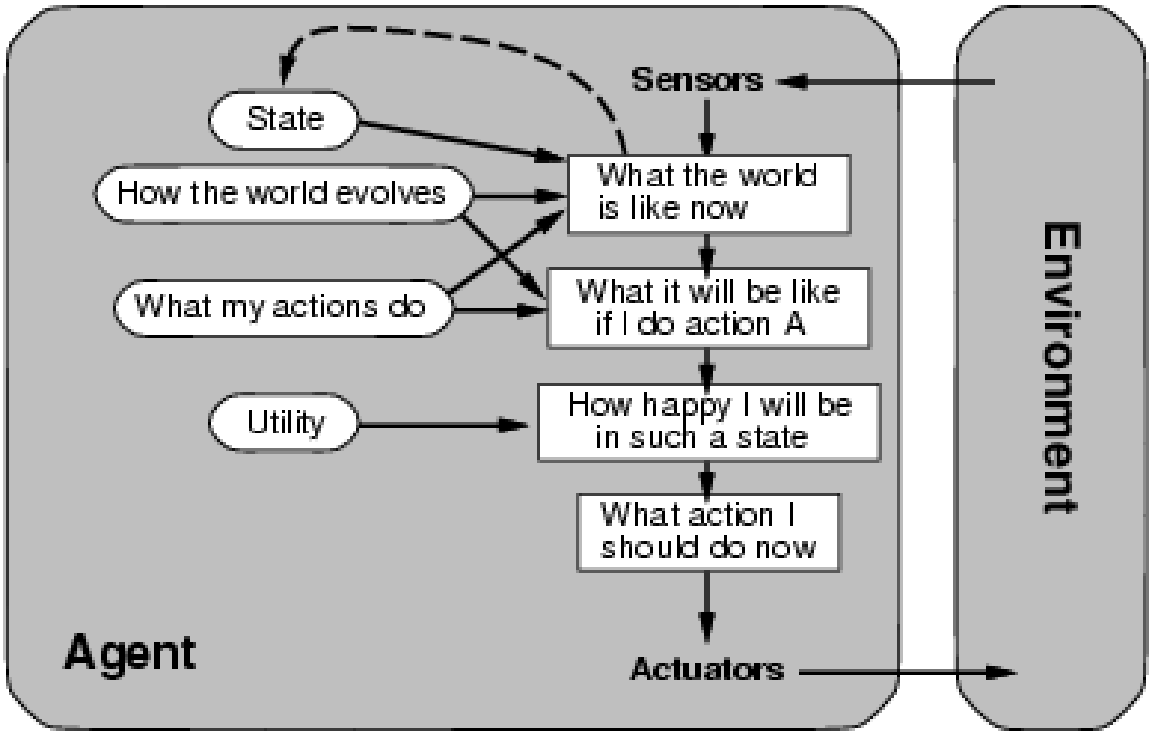
functions.

b. Is it practical for the Agent to use function based approach to recognize the shapes? If not, what would be a practical approach to recognize the shapes?

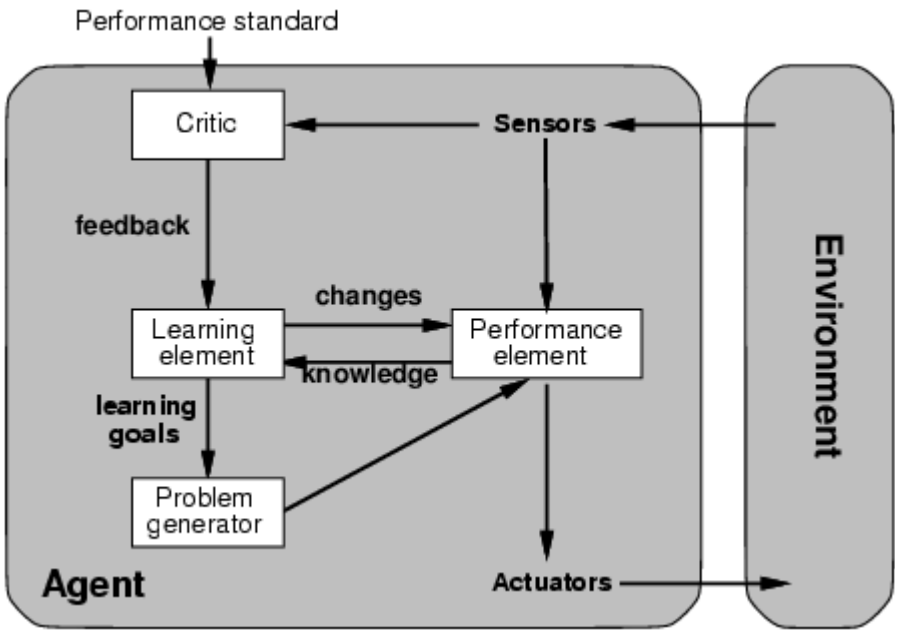
Ans: No, is it NOT practical for the Agent to use function based approach to recognize the shapes as there are too many states. Best way to handle this is to use some key features like edge, arc, semicircle etc. That cane make the intended unique shapes.

a. Consider a Natural Language based **Customer Support Softbot Agent**.
Draw its architecture along with the environment. Mention and explain important goals and utility functions.
Ans.

Customer Support Softbot should be a Utility as well as Learning Agent.

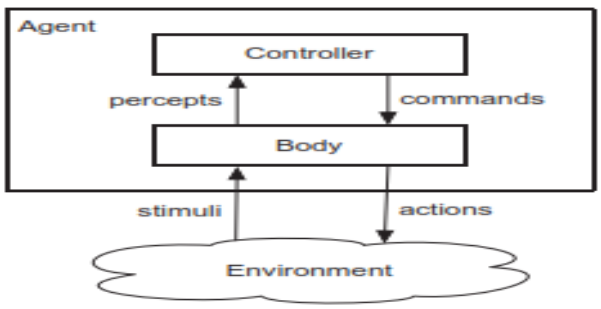


+

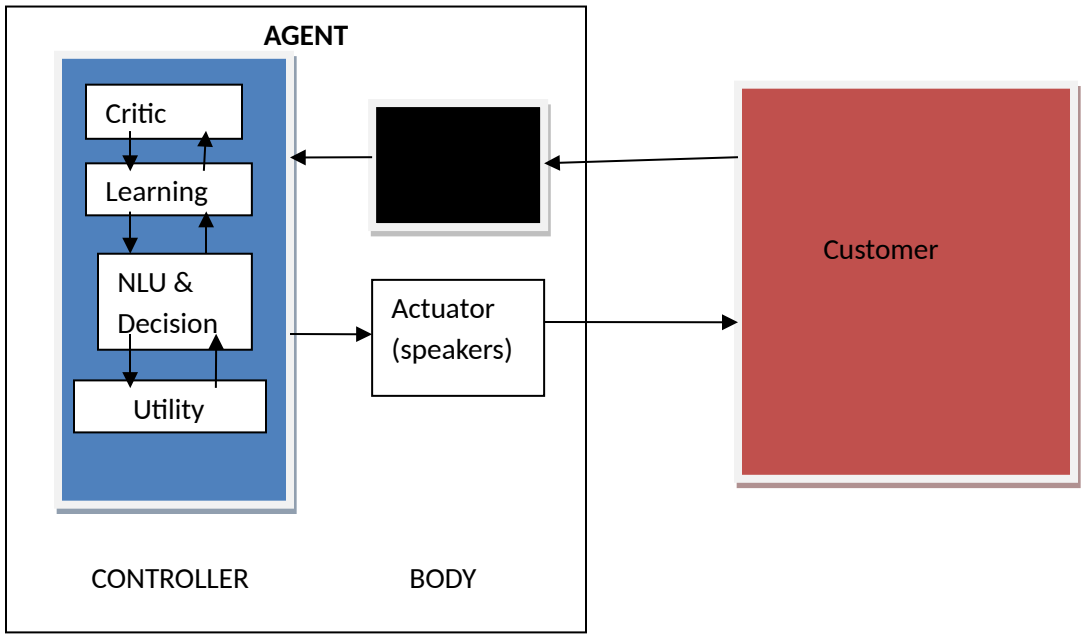


Goal: Get good answers to the questions

Another Way to Represent an Agent (David Poole)



Agent Diagram for NLU Based Customer Support Softbot



Utility: Get very good answers / directions with minimal number of questions, minimal time, minimal additional efforts (i.e. Agent should not suggest customer to look into a website for more answers that the Agent can provide easily etc).

Learning inputs: not correct answer, customer hanging up too soon, communication not good, not a good answer to "Agent says – have I answered your questions well?" ….

KEY ISSUES:

- **Understanding the percept i.e. the "semantics" of the sentences ("this is not what I was looking for" "Or I meant this and not that.." or Agent not understanding the question and just request to say it again and again..) are dependent on this, a major issue.**
- **Using the facts, rules etc and coming up with a good answer**
- **Muti-level goal states e.g. understanding "semantics", a goal not easily measurable; question is answered – another goal not easily measurable.**
- **Learning**
- **Uncertainties in various information, decisions, results**

Question 3: 8 points (2.5 + 5.5)

a) Consider the following features in determining whether you will be able to play tennis or not: Outlook, temperature, humidity and wind. Outlook attribute has 3 values, namely, Sunny, Overcast and Rain. There 14 training example in the training set, S as shown below. The Entropy for the training set is 0.94.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Calculate the Information Gain for Outlook and Humidity attributes. The information Gain for Wind and Temperature are 0.048 and 0.029.

Ans.

The entropy of the training set S is 0.94 (given) = -(P+log(p+) + p-log(p-)) = -9/14log(9/14) – 5/14log(5/14).

Now, we need to calculate the Information gains :

First we need, S(Sunny) <- [2+, 3-]

S(Overcast) <- [4+, 0-]

S(Rain) <- [3+, 2-]

Information Gain (S, Outlook) = Entropy(S) – (5/14) S(Sunny) – 4/14 S(Overcast) – 5/14S(Rain) = 0.94 – 5/14[-(2/5)log2/5 – (3/5)log3/5] – 4/14[-(4/4)log4/4 -0] – 5/14[-(3/5)log3/5 – (2/5)log2/5]

= 0.94 -5/14[-(2/5)x-1.32 –(3/5)x-0.74] - 4/14[-0-0] – 5/14[-(3/5)x-0.74 – (2/5)x-1.32]

=0.94-0.34-0.34 = 0.25

Similarly, Information Gain (S, Humidity) = Entropy(S) – (7/14) S(High) – 7/14 S(Normal) = 0.151

We will use the Highest value of the Information Gain as the root of the tree as it has the maxmimum Information. Hence the root of the tree is « Outlook ».

b) Consider to solve the following equation using GA (Genetic Algorithm):
a + 2b + 3c + 4d = 30. You are basically trying to find out the values for coefficients a, b, c and d using a search process. So, each chromosome has 4 numeric values for 4 coefficients. Assume a population size of 4. Assume values of the coefficients are between 0 and 30. You need to **show just 2 iterations** (thus you may not find a solution).
Determine a good fitness function. Selection is done by using the 2 top probabilities for the 2 parents that need to be mated for the next generation (so you would need to calculate appropriate probabilities). Then pick 2 population from 4 (after mating) using highest fitness values. From the new 4 population, select 2 parents for the 2nd iteration (using the criterion mentioned above) and complete the 2nd iteration. **Show all GA steps.**

Ans.: Here the goal is to minimize f = a + 2b + 3c + 4d – 30 (ideally to 0). The constraint is that a, b, c and d are between 0 and 30.

1st Iteration -

Initialization: Assume 4 chromosomes (population size of 4 as given) –

P1 - {1, 10, 20, 30}

P2 - {5, 6, 7, 21}

P3 - {10, 11, 21, 29}

P4 - {7, 8, 9, 10}

Fitness Function – Since we are trying to minimize f, a good fitness function will be 1/f so that the probability will be high i.e. fitness will be high etc.

Evaluation –

For P1 -> 1 + 2x10 + 3x 20 + 4x30 – 30 = 1 + 20 + 60 + 120 – 30 = 71 -> 1/f = .014

For P2 -> 5 + 6x10 + 7x 20 + 21x30 – 30 = 5 + 60 + 140 + 630 – 30 = 805 -> 1/f = .0012

For P3 -> 10 + 11x10 + 21x 20 + 29x30 – 30 = 10 + 110 + 420 + 870 – 30 = 1380 -> .000072

For P4 -> 7 + 8x10 + 9x 20 + 10x30 – 30 = 7 + 80 + 180 + 300 – 30 = 537-> 1/f = .0018

Probabilities – for P1 .014 (.014 + .0012 + .00074 + .0018) = .014 / .0177 = 0.79.

Selection:

Clearly, the 2 highest probabilities will be with P1 and P4. So, we take P1 and P4 as the parents to mate (these are the fittest).

P1 - {1, 10, 20, 30} P4 - {7, 8, 9, 10}. Use 3rd position as Crossover point. Hence, we have,

P1' – {1, 10, 9, 10} and P2' – {7, 8, 20, 30}.

Mutation – randomly select position 4 for P1' and position 2 for P2'

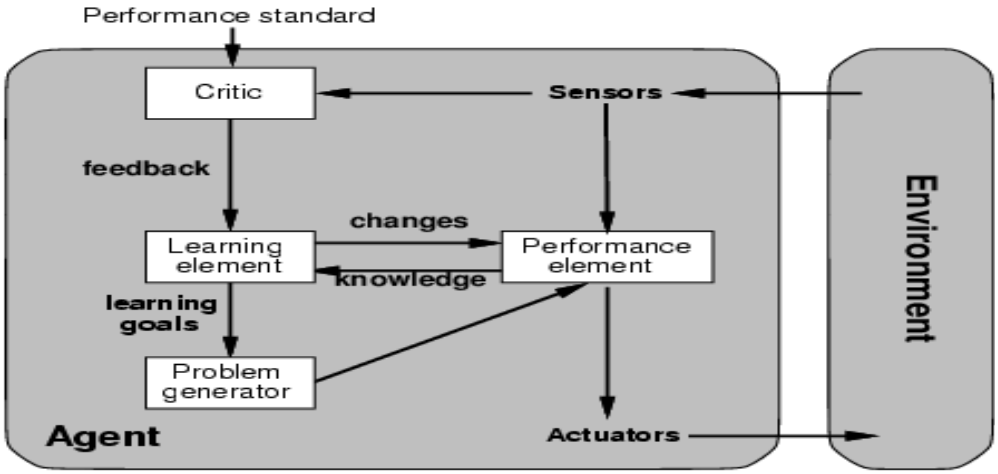
- ➔ {1, 10, 9, 20} and {7, 19, 20, 30}. We now need to calculate the fitness of all these 4 chromosomes and then select the 2 with the highest fitness values (call these Q1, Q2).

2nd iteration:

Now we have 4 new population: P2, P4, Q1 and Q2. Now repeat the same process again starting with calculating the fitness values to make selection for next parents.

Question 4: 7 points (2 + 2 + 3)

Consider a Learning Agent as shown below to be used for a **Reinforcement Learning** application.



a. Describe the function of **Problem Generator** unit and **Critic** unit.

Ans.

Critic -

The critic tells the learning element how well the agent is doing with respect to a fixed performance standard. The critic is necessary because the percepts themselves provide no indication of the agent's success. For example, a chess program could receive a percept indicating that it has checkmated its opponent, but it needs a performance standard to know that this is a good thing; the percept itself does not say so. It is important that the performance standard be fixed. Conceptually, one should think of it as being outside the agent altogether, because the agent must not modify it to fit its own behavior.

Problem Generator – it is responsible for exploration i.e.

for suggesting actions that will lead to new and informative experiences. The point is that if the performance element had its way, it would keep doing the actions that are best, given what it knows. But if the agent is willing to explore a little, and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run. The problem

Generator's job is to suggest these exploratory actions. This is what scientists do when they carry out experiments.

The Agent needs to do both exploitation and exploration.

b. What are the key steps that the learning agent uses to maximize its rewards?

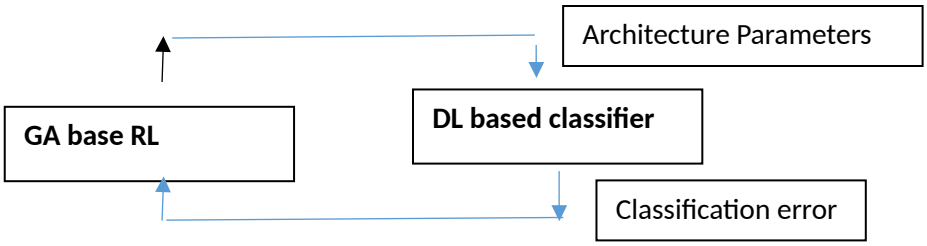
Ans.

Reinforcement learning agent tries to maximize its rewards over time using search over the state space of possible inputs and possible outputs. This learning is sort of trial and error learning (also known as Law of Effect). The agent would need to do **various steps** to achieve maximum reward which includes selecting actions with some strategy, discounting (as needs to optimize over time), measuring values, doing exploitation and exploration, refining policy as appropriate, measuring performance of each trial and repeating this process.

Learning element plays a key role in performing all these.

c. Consider a **Deep Learning** NN. You would like to **optimize its architecture** by determining number of sparsity hidden layers, number of neurons in each sparsity hidden layer, and also the type of activation functions (which also can be represented by a number) for the output / classification layer. You are considering to use a RL (similar to the one in Q4). The RL can only use **Genetic Algorithms**. Describe whether this can be achieved. If it cannot be achieved, explain why.

Note: DL uses regular NN input when doing its classification and there are classification errors until an optimal / near optimal architecture is determined.



Ans.:

Yes, GA can be used to determine the architecture of the DL. Let's say the key paprameters of the DL is a vector like

{# of sparsity hidden layers, # of neurons in each sparsity layer, type of activation} – all integers.

The out of the DL (including the classification stage) is a classification vector like

{0,1, 1,0, 0, 1}. This classification vector will have errors in general. This error (which is also a vector) can be used as an input the to the Critic in RL. The Critic then can convert this into an

equivalent information to the LE (learning element} which can perform a GA operation to find the next generation of the population. The process can be repeated until the result is optimal / near optimal. Note that DL runs its input in iteration of GA to do it classification so the error vector can be used by GA.