GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

Department of Computer Science and Engineering

HANDOUT

on

MACHINE LEARNING

Vision

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.

Mission

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

Program Educational Objectives

- Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.

- Manage software projects with significant technical, legal, ethical, social, environmental and economic considerations.

- Demonstrate commitment and progress in lifelong learning, professional development, leadership and Communicate effectively with professional clients and the public.

HANDOUT ON MACHINE LEARNING

Class & Sem : IV B.Tech – II Semester Year : 2019-20

Branch : CSE Credits : 3

1. Brief History & Scope of the Subject

A machine that is intellectually capable as much as humans has always fired the imagination of writers and also the early computer scientist who were excited about artificial intelligence and machine learning, but the first machine learning system was developed in the 1950s. In 1952, Arthur Samuel was at IBM. He developed a program for playing Checkers. In 1957, Rosenblatt proposed the Perceptron. However, the work along these lines suffered a setback when Minsky in 1969 came up with the limitations of perceptron. In 1986, J.R.Quinlan came up with decision tree learning, specifically the ID3 algorithm.

In the 90s, machine learning embraced statistics to a large extent. It was during this time, that support vector machines were proposed. It was a machine learning breakthrough and the support vector machines was proposed by Vapnik and Cortesin 1995 and S.V. Hemhad very strong theoretical standing and empirical results. Another strong machine learning model was proposed by Freund and Schapirein 1997, which was part of what we called ensembles or boosting and they came up with an algorithm called Adaboost by which they could create a strong classifier from an ensemble of weak classifiers. During 2001, Bayes net learning was also proposed. The rise of neural network began roughly in 2005 with the conjunction of many different discoveries for people by Hinton, LeCun, Bengio, Andrew and other researchers.

Some of the applications of machine learning are : In 1994, the first self driving car made a road test; in 1997, Deep Blue beat the world champion Gary Kasparov in the game of chess; in 2009 we have Google building self driving cars; in 2011,Watson, again from IBM, won the popular game of Jeopardy;

2014, human vision surpassed by ML systems. In 2014-15, machine translation systems driven by neural networks are very good and they are better than the other statistical machine translation systems where certain concepts and certain technology.

Now, in machine learning we have GPU's, which are enabling the use of machine learning and deep neural networks. There is the cloud, there is availability of big data and the field of machine learning is very exciting now.

2. Prerequisites

- Mathematics & statistics – calculus, differential equations, probability theory, graph theory
- Programming Experience – Python, R

3. Course Objectives

- To familiarize with supervised and unsupervised learning.
- To get acquainted with various machine learning algorithms.

4. Course Outcomes: Students will be able to

CO1: Define the basic concepts and applications of Machine learning

CO2:Demonstrate Version Space Find-S and Candidate elimination algorithms.

CO3: Design a decision tree

CO4: Apply appropriate supervised machine learning algorithm for an application.

CO5: Apply appropriate unsupervised machine learning algorithm for an application.

CO6: Compare supervised learning algorithms with unsupervised machine learning algorithms.

Program Outcomes:

Computer Science and Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

5. **Mapping of Course Outcomes with Program Outcomes:**

	a	b	c	d	e	f	g	h	i	j	k	l
CO1	H											M
CO2			H									
CO3												
CO4					M		M					
CO5					M		M					
CO6	M											

6. Prescribed Text Books:

1. Tom M. Mitchell, Machine Learning, Mc Graw Hill Education.

7. Reference Books:

1. Ethem Alpaydin, Introduction to machine learning, 2nd edition, PHI.

8. URLs and Other E-Learning Resources

1. https://www.coursera.org/learn/machine-learning

9. Digital Learning Materials:

1. http://nptel.ac.in/courses/106106139/
2. http://nptel.ac.in/courses/106105152/

10. Lecture Schedule / Lesson Plan:

Topic	No. of Periods	
	Theory	Tutorial
UNIT I : Introduction		
Well- posed learning problems	3	1
Designing a learning system,	3	
Perspectives and issues in machine learning.	1	
UNIT - II: Concept Learning		
Introduction, A concept learning task,	1	1
Concept learning as search, Find-s: finding a maximally specific hypothesis,	2	
Version spaces and the candidate elimination algorithm	2	1
Remarks on version spaces and candidate elimination	1	
Inductive bias	1	
UNIT - III: Decision Tree Learning		
Decision tree representation	1	1
Appropriate problems for decision tree learning	1	
The basic decision tree learning algorithm	4	
Hypothesis space search in decision tree learning	2	1
Inductive bias in decision tree learning	2	
Issues in decision tree learning.	2	
UNIT - IV: Bayesian learning		
Bayes theorem	1	1
Byes theorem and concept learning	3	
Maximum likelihood and least squared error hypotheses	1	
Maximum likelihood hypotheses for predicting probabilities	2	1
Bayes optimal classifier	1	
An example learning to classify text	1	
Bayesian belief networks	2	
UNIT - V: Computational learning theory - 1		
Probability learning an approximately correct hypothesis	2	1
Sample complexity for infinite Hypothesis spaces	3	
The mistake bound model of learning	3	
Instance Based learning- Introduction.	1	
UNIT - VI: Computational learning theory - 2		

K- Nearest Neighbour Learning	3	1
Locally Weighted Regression	2	
Radial Basis Functions	2	
Case-Based Reasoning	2	
Remarks on Lazy and Eager Learning	1	
Total No. of Periods	56	9

UNIT- I

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

In other words, Machine learning is a set of tools which allows us to "teach" computers how to perform tasks by providing examples of how they should be done.

For **example**, suppose we wish to write a program to distinguish between valid email messages and unwanted spam. We could try to write a set of simple rules, for example, flagging messages that contain certain features (such as the word "GOOD NEWS" or obviously-fake headers). However, writing rules to accurately distinguish which text is valid can actually be quite difficult to do well, resulting either in many missed spam messages, or, worse, many lost emails. Worse, the spammers will actively adjust the way they send spam in order to trick these strategies (e.g., writing "G00D NEW$"). Writing effective rules - and keeping them up-to-date - quickly becomes an insurmountable task. Fortunately, machine learning has provided a solution. Modern spam filters are "learned" from examples: we provide the learning algorithm with example emails which we have manually labelled as "ham" (valid email) or "spam" (unwanted email), and the algorithms learn to distinguish between them automatically.

1.1 WELL-POSED LEARNING PROBLEMS

A computer program is said to ***learn*** from **experience E** with respect to some class of **tasks *T*** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

A computer program that learns **to play checkers** might improve its performance as ***measured by its ability to win*** at the class of tasks involving ***playing checkers games,*** through experience ***obtained*** by ***playing games against itself.***

In general, to have a well-defined learning problem, we must identity these three features:

- The class of tasks, T
- The measure of performance, P to be improved, and
- The source of experience, E

***Example1*: The Checkers learning problem:**

- **Task T:** playing checkers
- **Performance measure P:** percent of games won against opponents
- **Training experience E**: playing practice games against itself

***Example 2:* A handwriting recognition learning problem:**

- **Task T**: recognizing and classifying handwritten words within images
- **Performance measure *P:*** percent of words correctly classified
- **Training experience E**: a database of handwritten words with given classifications

***Example 3*: A robot driving learning problem:**

- **Task T**: driving on public four-lane highways using vision sensors
- **Performance measure P**: average distance travelled before an error (as judged by human overseer)
- **Training experience E**: a sequence of images and steering commands recorded while observing a human driver

Basic Terminology:

Training data: A set of examples used for learning.

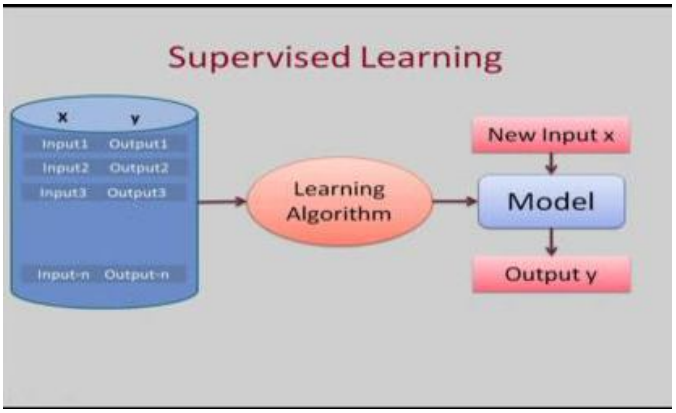
Test data: A set of examples used only to assess the performance of a fully-trained classifier.

Labelled and unlabeled data: If we consider data in the form of <x,y> (where x is input and y is output) then labelled data consists of both x and y where as unlabeled data has only x.

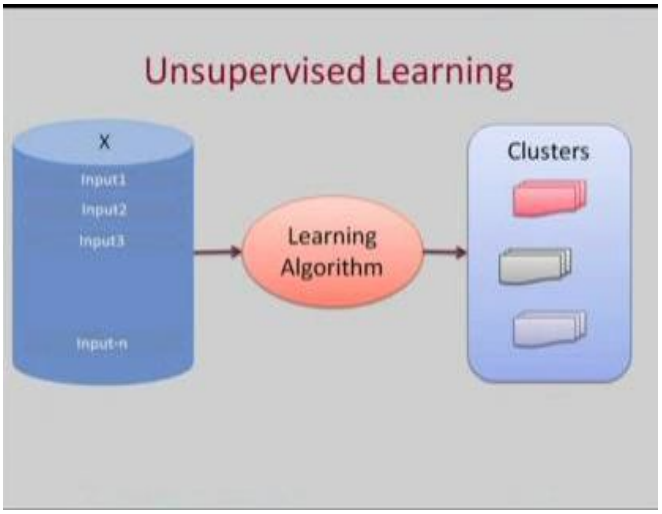
Types of Machine Learning paradigms:

Some of the main types of machine learning are:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning



Supervised Learning: In Supervised Learning, the training data is labelled with the correct answers, for example whether the mail is "spam" or "ham". The two most common types of supervised learning are classification (where the outputs are discrete labels, as in spam filtering) and regression (where the outputs are continuous/real-valued). A simple demonstration of supervised learning is shown in the figure:



Unsupervised Learning: In Unsupervised learning, a collection of unlabeled data is given, which need to be analyzed and patterns discovered. The most important example of unsupervised learning is clustering. A simple

demonstration of unsupervised learning is shown in the figure:

Reinforcement learning: In Reinforcement learning, an agent (e.g., a robot or controller) seeks to learn the optimal actions to take based the outcomes of past actions.

Some Applications of Machine Learning:

Machine learning tools can be used in many domains like - medicine (eg: diagnose a disease), vision, robot control (eg: design autonomous mobile robots), etc. Some of the successful applications of machine learning include the following:

- **Learning to recognize spoken words**: For example, the SPHINX system learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal.
- **Learning to drive an autonomous vehicle:** For example, the ALVINN system has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars.
- **Learning to classify new astronomical structures:** For example, decision tree learning algorithms have been used by NASA to learn how to classify celestial objects from the second Palomar Observatory Sky Survey. This system is now used to automatically classify all objects in the Sky Survey, which consists of three terabytes of image data.
- **Learning to play world-class backgammon**: For example, the world's top computer program for backgammon, TD-GAMMON learned its strategy by playing over one million practice games against itself. It now plays at a level competitive with the human world champion.

1.2 DESIGNING A LEARNING SYSTEM

The performance measure for playing checkers is the percent of games it wins in the world tournament. Designing a program to learn to play checkers, with

the goal of entering it in the world checkers tournament is briefly explained below:

The major choices in designing a learning system are:

- Choosing the Training Experience
- Choosing the Target Function
- Choosing a representation for the Target Function
- Choosing a Function approximation algorithm

1.2.1 Choosing the Training Experience:

- The type of training experience available can have a significant impact on success or failure of the learner.
- The three important attributes of Training experience are
    a. Direct/Indirect feedback
    b. The degree to which the learner controls the sequence of training examples
    c. The distribution of examples

Direct/Indirect feedback regarding the choices made by the performance system:

- One of the key attributes of Training experience are whether the training experience has direct/indirect feedback regarding the choices made by the performance system.

    Example: In learning to play checkers, the system might learn from ***direct*** training examples consisting of individual checkersboard states and the correct move for each. Alternatively, it might have available only ***indirect*** information consisting of the move sequences and final outcomes of various games played.
- In Indirect feedback there is a problem of **credit assignment** - determining the degree to which each move in the sequence deserves credit or blame for the final outcome. Hence, learning from direct training feedback is typically easier than learning from indirect feedback.

The degree to which the learner controls the sequence of training examples:

- For example, in learning to play checkers, the learner might rely on the teacher to select informative board states and to provide the correct move for each.
- Alternatively, the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move or the learner may have complete control over both the board states and (indirect) training classifications, as it does when it learns by playing against itself with no teacher present.

The distribution of examples:

- Learning is most reliable when the training examples follow a distribution similar to that of future test examples.
- In checkers learning scenario, the performance metric P is the percent of games the system wins in the world tournament. If its training experience E consists only of games played against itself, there is no obvious danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested.
- For example, the learner might never encounter certain crucial board states that are very likely to be played by the human checkers champion. In practice, it is often necessary to learn from a distribution of examples that is somewhat different from those on which the final system will be evaluated (e.g., the world checkers champion might not be interested in teaching the program!).

A checkers learning problem:

- Task **T**: playing checkers
- Performance measure ***P:*** percent of games won in the world tournament
- Training experience **E**: games played against itself

In order to complete the design of the learning system, we must now choose

1. the exact type of knowledge to be learned
2. a representation for this target knowledge
3. a learning mechanism

1.2.2 Choosing the Target Function:

- This design choice determines exactly what type of knowledge will be learned and how this will be used by the performance program.
- Checkers-playing program must be able to generate the ***legal*** moves from any board state. The program needs only to learn how to choose the ***best*** move from among these legal moves.
- Let us call this function ***ChooseMove*** and use the notation ***ChooseMove*: B→M** to indicate that this function accepts as input any board from the set of legal board states *B* and produces as output some move from the set of legal moves ***M.***
- We can view our task of improving performance P at task T to the problem of learning a target function such as *ChooseMove*. Hence the choice of target function is the key design choice.
- An alternative target function and one that will turn out to be easier to learn in checkers-playing program is an evaluation function that assigns a numerical score to any given board state. Let us call this target function ***V*** and again use the notation ***V*** : B→ ℝ to denote that ***V*** maps any legal board state from the set B to some real value .
- What exactly should be the value of the target function ***V*** for any given board state? Of course any evaluation function that assigns higher scores to better board states will do. Nevertheless, we will find it useful to define one particular target function ***V*** among the many that produce optimal play. As we shall see, this will make it easier to design a training algorithm. Let us therefore define the target value ***V(b)*** for an arbitrary board state ***b*** in *B*, as follows:

1. if b is a final board state that is won, then V(b) = 100
2. if b is a final board state that is lost, then V(b) = -100
3. if b is a final board state that is drawn, then V(b) = 0
4. if b is a not a final state in the game, then V(b) = V(b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game (assuming the opponent plays optimally, as well).

- The goal of learning in this case is to discover an ***operational*** description of ***V;*** that is, a description that can be used by the checkers-playing program to evaluate states and select moves within realistic time bounds.
- Now our learning task has become as the problem of determining "an operational description of ideal target function" which is very difficult to learn so we try to learn an "approximation to target function" denoted by

1.2.3 Choosing a Representation for the Target Function:

- We must choose a representation that the learning program will use to describe the function that it will learn.
- can be represented by using a large table or collection of rules or a quadratic polynomial function or an ANN (Artificial Neural Network).
- In general, this choice of representation involves a crucial tradeoff.
- On one hand, we wish to pick a very expressive representation to allow representing as close an approximation as possible to the ideal target function V.
- On the other hand, the more expressive the representation, the more training data the program will require in order to choose among the alternative hypotheses it can represent.

Example: For Checkers-playing program, a simple representation for the function □ will be calculated as a linear combination of the following board features:

X_1: the number of black pieces on the board

X_2: the number of red pieces on the board

X_3: the number of black kings on the board

X_4: the number of red kings on the board

X_5: the number of black pieces threatened by red (i.e., which can be captured on red's next turn)

X_6: the number of red pieces threatened by black

Thus, our learning program will represent □(b) as a linear function of the form

$$□(b) = W_0 + W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6$$

where W_0 through W6 are numerical coefficients, or weights, to be chosen by the learning algorithm.

To summarize our design choices thus far, we have elaborated the original formulation of the learning problem by choosing a type of training experience, a target function to be learned, and a representation for this target function. Our elaborated learning task is now

Partial design of a checkers learning program:

- Task T: playing checkers
- Performance measure ***P:*** percent of games won in the world tournament
- Training experience E: games played against itself
- Target function: V:Board + ℝ
- Target function representation

  $$□(b) = W_0 + W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6$$

The first three items above correspond to the specification of the learning task, whereas the final two items constitute design choices for the implementation of the learning program. Notice the net effect of this set of design choices is to

reduce the problem of learning a checkers strategy to the problem of learning values for the coefficients W_0 through W_6 in the target function representation.

1.2.4 Choosing a Function Approximation Algorithm:

In order to learn the target function □ we require a set of training examples, each describing a specific board state b and the training value $V_{train(b)}$ for b. In other words, each training example is an ordered pair of the form <b, $V_{train(b)}$>.

- **Estimating Training Values:** It is ambiguous to assign training values to intermediate states of a learning problem. But, one simple approach has been found to be surprisingly successful for this. The approach is to assign the training value of $V_{train(b)}$ for any intermediate board state *b* to be □*(Successor(b))* where □ is the learner's current approximation to *V* and *Successor(b)* denotes the next board state following *b* for which it is again the program's turn to move (i.e., the board state following the program's move and the opponent's response). This rule for estimating training values can be summarized as

  $$\mathbf{V_{train(b)} \leftarrow \square(Successor(b))}$$

- **Adjusting the Weights:** Now we have to choose the weights Wi to best fit the set of training examples <b, $V_{train(b)}$>. One common approach is to define set of weights, as that which minimizes the squared error E between the training values and the values predicted by the hypothesis □*:*

  $$\mathbf{E \equiv \sum_{<b,\ Vtrain(b)> \in\ Training\ Examples} (V_{train(b)} - \square(b))^2}$$

  Thus, we seek the weights, or equivalently the □, that minimize E for the observed training examples.

  LMS (Least Mean Square) Algorithm: LMS is a weight adjustment algorithm, which will incrementally refine the weights as new training examples become available and that will be robust to errors in the estimated training values. The LMS algorithm is defined as follows:

For each training example <b, $V_{train(b)}$>

- **Use the current weights to calculate □(b)**
- **For each weight W_i, update it as**

$$W_i \leftarrow W_i + \square\ (V_{train(b)} - \square(b))\ X_i$$

Here □ is a small constant (e.g., 0.1) that moderates the size of the weight update.

1.2.5 Final Design of a Learning System:

In general, the central components in any learning problem are the following:

- Performance System
- Critic
- Generalizer
- Experiment Generator

The **Performance System** is the module that must solve the given performance task, in this case playing checkers, by using the learned target function(s). It takes an instance of a new problem (new game) as input and produces a trace of its solution (game history) as output. In our case, the strategy used by the Performance System to select its next move at each step is determined by the learned □ evaluation function. Therefore, we expect its performance to improve as this evaluation function becomes increasingly accurate.

The **Critic** takes as input the history or trace of the game and produces as output a set of training examples of the target function. As shown in the diagram, each training example in this case corresponds to some game state in the trace, along with an estimate ***Vtrai*n,** if the target function value for this example.

The **Generalizer** takes as input the training examples and produces an output hypothesis that is its estimate of the target function. It generalizes from the specific training examples, hypothesizing a general function that covers these examples and other cases beyond the training examples. In our example, the

Generalizer corresponds to the LMS algorithm, and the output hypothesis is the function described by the learned weights ***wo,*** . . . , ***W6.***

The **Experiment Generator** takes as input the current hypothesis (currently learned function) and outputs a new problem (i.e., initial board state) for the Performance System to explore. Its role is to pick new practice problems that will maximize the learning rate of the overall system. In our example, the Experiment Generator follows a very simple strategy: It always proposes the same initial game board to begin a new game. More sophisticated strategies could involve creating board positions designed to explore particular regions of the state space.

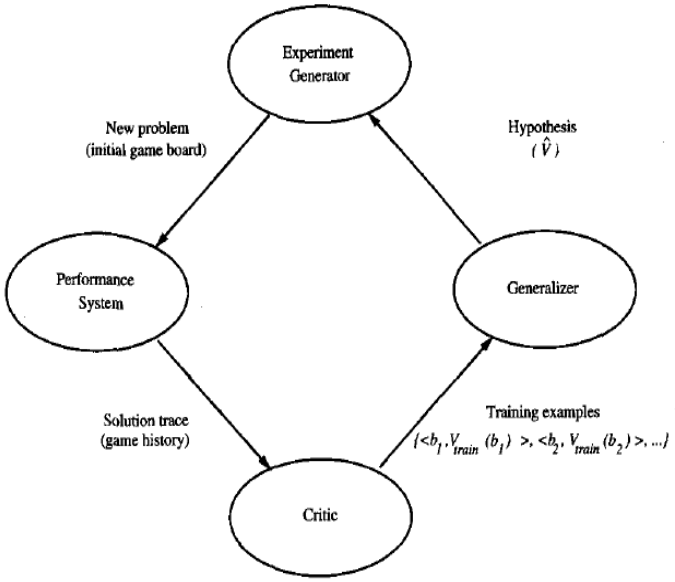


Figure: Summary of choices in designing the checkers learning program.

1.3 Perspectives in Machine Learning:

- One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner.
- For example, consider the space of hypotheses that could in principle be output by the above checkers learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for the weights W_0 through W_6**.** The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with the available training examples.
- The LMS algorithm for fitting weights achieves this goal by iteratively tuning the weights, adding a correction to each weight each time the hypothesized evaluation function predicts a value that differs from the training value. This algorithm works well when the hypothesis representation considered by the learner defines a continuously parameterized space of potential hypotheses.

Issues in Machine Learning:

The checkers example raises a number of generic questions about machine learning and they are given below:

- What algorithms exist for learning general target functions from specific training examples? In what settings will particular algorithms converge to the desired function, given sufficient training data? Which algorithms perform best for which types of problems and representations?
- How much training data is sufficient? What general bounds can be found to relate the confidence in learned hypotheses to the amount of training experience and the character of the learner's hypothesis space?
- When and how can prior knowledge held by the learner guide the process of generalizing from examples? Can prior knowledge be helpful even when it is only approximately correct?

- What is the best strategy for choosing a useful next training experience, and how does the choice of this strategy alter the complexity of the learning problem?
- What is the best way to reduce the learning task to one or more function approximation problems? Put another way, what specific functions should the system attempt to learn? Can this process itself be automated?
- How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

UNIT-I

Assignment-Cum-Tutorial Questions

SECTION-A

Objective Questions

1. Machine learning is [ ]

A. The autonomous acquisition of knowledge through the use of computer programs

B. The autonomous acquisition of knowledge through the use of manual programs

C. The selective acquisition of knowledge through the use of computer programs

D. The selective acquisition of knowledge through the use of manual programs

2. Factors which affect the performance of learner system does not include [ ]

A. Representation scheme used B. Training scenario

C. Type of feedback D. Good data structures

3. What types of Machine Learning, if any, best describe the following three scenarios: [ ]

(i) A coin classification system is created for a vending machine. The developers obtain exact coin specifications from the U.S. Mint and derive a statistical model of the size, weight, and denomination, which the vending machine then uses to classify coins.

(ii) Instead of calling the U.S. Mint to obtain coin information, an algorithm is presented with a large set of labeled coins. The algorithm uses this data to infer decision boundaries which the vending machine then uses to classify its coins.

(iii) A computer develops a strategy for playing Tic-Tac-Toe by playing repeatedly and adjusting its strategy by penalizing moves that eventually lead to losing.

A. (i) Supervised Learning, (ii) Unsupervised Learning, (iii) Reinforcement Learning

B. (i) Supervised Learning, (ii) Not learning, (iii) Unsupervised Learning

C. (i) Not learning, (ii) Reinforcement Learning, (iii) Supervised Learning

D. (i) Not learning, (ii) Supervised Learning, (iii) Reinforcement Learning

E. (i) Supervised Learning, (ii) Reinforcement Learning, (iii) Unsupervised Learning

4. Which of the following problems are best suited for Machine Learning? [ ]

(i) Classifying numbers into primes and non-primes.

(ii) Detecting potential fraud in credit card charges.

(iii) Determining the time it would take a falling object to hit the ground.

(iv) Determining the optimal cycle for traffic lights in a busy intersection.

A. (ii) and (iv) B. (i) and (ii)

C. (i), (ii), and (iii) D. (iii) E. (i) and (iii)

5. I am the marketing consultant of a leading e-commerce website. I have been given a task of making a system that recommends products to users based on their activity on Facebook. I realize that user-interests could be highly variable. Hence I decide to

i. First, cluster the users into communities of like-minded people and

ii. Second, train separate models for each community to predict which product category (e.g. electronic gadgets, cosmetics, etc) would be the most relevant to that community.

The first task is a/an _____ learning problem while the second is a/an _____ problem. [ ]

Choose from the options:

A. Supervised and unsupervised B. Unsupervised and supervised

C. Supervised and supervised D. Unsupervised and unsupervised

6. Which ONE of the following are regression tasks? [ ]

A. Predict the age of a person

B. Predict the country from where the person comes from

C. Predict whether the price of petroleum will increase tomorrow

D. Predict whether a document is related to science

7. Which of the following are classification tasks? (Mark all that apply) [ ]

A. Find the gender of a person by analyzing his writing style

B. Predict the price of a house based on floor area, number of rooms etc.

C. Predict whether there will be abnormally heavy rainfall next year

D. Predict the number of copies of a book that will be sold this month

8. Which of the following are examples of unsupervised learning? [ ]

A. Group news articles based on text similarity

B. Make clusters of books on similar topics in a library

C. Filter out spam emails

D. Segment online customers into two classes based on their age group – below 25 or above 25

SECTION-B

Descriptive Questions

1. Define machine learning.
2. What do you mean by a well –posed learning problem? Explain the important features that are required to well –define a learning problem.
3. Explain well posed learning problem for the following:
   i) A checkers learning problem.
   ii) A handwritten recognition learning problem.
   iii) A robot driving learning problem.
4. Discuss different applications of machine Learning.
5. Explain the phases in designing a learning system.

6. State LMS weight update rule.

7. Discuss perspectives and issues in machine learning.

8. Define target function in machine learning with an example.

9. List the objectives of machine learning.

10. Devise a simple machine learning solution to solve Checkers game problem.

11. Explain various design choices of machine learning strategies.