Data Mining

Lesson 1

Introduction to data mining and machine learning

MSc in Computer Science
University of New York Tirana
Assoc. Prof. Dr. Marenglen Biba

Course Outline

- Introduction to data mining and machine learning
- Inductive learning
- Decision trees
- Rule induction
- Instance-based learning
- Bayesian learning
- Neural networks
- Support vector machines
- Other machine learning models
- Engineering data mining tasks

Objectives

- Understand principles of data mining and machine learning.
- Understand and describe the main models for data analysis and the differences among them.
- Assess raw input data, and process it appropriately to provide suitable input for a range of data mining facilities/algorithms.
- Critically evaluate and select appropriate data-mining facilities/ algorithms/ models and be able to apply them and interpret and report the output appropriately.
- Understand and apply machine learning algorithms to data mining tasks.

Assessment

Methods of Assessment	**Please identify the LAST item of assessment that a student sits with a tick**	**Grading Mode**	**Weighting %**	**Mark**	**Word Length**	**Outline Details**
Coursework			40	50%	4000	b) Group project Case study Covering Learning Outcomes: C,D,E.
Examination	√		60	50%		Covering Learning Outcomes: A,B

Books

- Machine Learning, Mcgraw-Hill International Edition. Thomas Mitchell,1st edition.

- Witten, I and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Edition. Morgan Kaufmann.

Software

- Weka
  - http://www.cs.waikato.ac.nz/~ml/weka/
- Oracle Data Miner 11g
- SQL Developer

Course Web Page

- http://www.marenglenbiba.net/dm/

Contact

- **Instructor:** Assoc. Prof. Dr. Marenglen Biba
  - Email: marenglenbiba@unyt.edu.al
  - Important: Please include Data Mining in the mail subject for all the communications regarding this course
  - Office: CSD Second Floor

Data Mining: Content

- Introduction to data mining and machine learning
- Inductive learning
- Decision trees
- Rule induction
- Instance-based learning
- Bayesian learning
- Neural networks
- Support vector machines
- Other machine learning models
- Engineering data mining tasks

Analyzing high volumes of data

- We are overwhelmed with data.
- The amount of data in the world, in our lives, seems to go on and on increasing — and there's no end in sight.
- Thus, there is strong motivation for developing methods that automatically analyze high volumes of data which cannot be handled manually.
- This has given birth to

  data mining, knowledge discovery and machine learning.

Data mining

- **Data mining** is the process of finding valid, useful and comprehensible patterns in raw data.

- This process is also known as **KDD** Knowledge Discovery in Databases.

Looking for Patterns

- Data Mining is about looking for patterns in data.
- There is nothing new about this.
- People have been seeking patterns in data since human life began. Hunters seek patterns in animal migration behavior, farmers seek patterns in crop growth, politicians seek patterns in voter opinion etc.
- *In data mining, the data is stored electronically and the search is automated — or at least augmented — by computer.*
- Even this is not particularly new.
  - Economists, statisticians, forecasters, and communication engineers have long worked with the idea that patterns in data can be sought automatically, identified, validated, and used for prediction.

Data Mining today

- What is new today is the staggering increase in opportunities for finding patterns in data.
- The unbridled growth of databases in recent years, databases on such everyday activities as customer choices, brings data mining to the *forefront of new business technologies*.
- It has been estimated that the amount of data stored in the world's databases *doubles* every 20 months, and although it would surely be difficult to justify this figure in any quantitative sense, we can all relate to the pace of growth qualitatively.
- *Data mining is about solving problems by analyzing data already present in databases.*

Structural Patterns

- How are the patterns expressed?
- Useful patterns allow us to make nontrivial predictions on new data.
- There are two extremes for the expression of a pattern:
  - as a black box which is effectively incomprehensible and
  - as a transparent box whose construction reveals the structure of the pattern.
- Both, we are assuming, make good predictions.

Structural Patterns

- The difference is whether or not the patterns that are mined are represented in terms of a structure that can be:
  - examined,
  - reasoned about
  - used to inform future decisions.

- Such patterns we call *structural* because they capture the decision structure in an explicit way.
  - In other words, they help to explain something about the data.

Data Mining, Machine Learning and structural patterns

- Now, finally, we can say what this course is about.

- *It is about techniques for finding and describing structural patterns in data.*

- Most of the techniques that we cover have developed within a field known as *machine learning.*
  - But we will first look at what structural patterns are.

Describing structural patterns

Table 1.1 The contact lens data.

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

Structural Description

- If tear production rate = reduced then recommendation = none

  Otherwise, if age = young and astigmatic = no then recommendation = soft

Issues in real data mining scenarios

- In most learning situations, the set of examples given as input is far from complete, and part of the job is to generalize to other, new examples.

- Real-life datasets invariably contain examples in which the values of some features, for some reason or other, are unknown — for example, measurements were not taken or were lost.

- The preceding rules classify the examples correctly, whereas often, because of errors or noise in the data, misclassifications occur even on the data that is used to train the classifier.

Learning and Machine Learning

- What is learning? What is machine learning?

- We defined data mining operationally as the process of discovering patterns, automatically or semiautomatically, in large quantities of data — and the patterns must be useful.

- An operational definition can be formulated in the same way for learning:
  - *Things learn when they change their behavior in a way that makes them perform better in the future.*

Training

- But there's still a problem. Learning is a rather slippery concept.
- Lots of things change their behavior in ways that make them perform better in the future, yet we wouldn't want to say that they have actually *learned.*
- A good example is a comfortable slipper. Has it *learned* the shape of your foot?
- It has certainly changed its behavior to make it perform better as a slipper! Yet we would hardly want to call this *learning.*
- In everyday language, we often use the word "*training*" to denote a mindless kind of learning.

Training and Learning

- We train animals and even plants, although it would be stretching the word a bit to talk of training objects such as slippers that are not in any sense alive.
- But learning is different.
- Learning implies thinking. Learning implies purpose.
- Something that learns has to do so intentionally. That is why we wouldn't say that a vine has learned to grow round a trellis in a vineyard — we'd say it has been *trained.*
- Learning without purpose is merely training.
- Or, more to the point, in learning the purpose is the learner's, whereas in training it is the teacher's.

Learning and Data Mining

- The kind of learning techniques explained in this course do not present these conceptual problems — they are called machine learning without really presupposing any particular philosophic stance about what learning actually is.

- Data mining is a practical topic and involves learning in a practical, not a theoretical sense.
  - We are interested in techniques for finding and describing structural patterns in data as a tool for helping to explain that data and make predictions from it.

Learning and Data Mining

- The data will take the form of a set of examples — examples of customers who have switched loyalties, for instance, or situations in which certain kinds of contact lenses can be prescribed.

- The output takes the form of predictions about new examples — a prediction of whether a particular customer will switch or a prediction of what kind of lens will be prescribed under given circumstances.

Gaining Knowledge with DM

- Many learning techniques look for structural descriptions of what is learned, descriptions that can become fairly complex and are typically expressed as *sets of rules* such as the ones described previously.
- Because they can be understood by people, these descriptions serve to explain what has been learned and explain the basis for new predictions.
- Experience shows that in many applications of machine learning to data mining, the *explicit knowledge structures* that are acquired, the structural descriptions, are *at least as important*, and often very much more important, than the ability to perform well on new examples.
- People frequently use data mining to gain knowledge, not just predictions.
- Gaining knowledge from data certainly sounds like a good idea if you can do it. To find out how, go on attending this course!

Simple examples: The weather problem

- The weather problem is a tiny dataset that we will use to illustrate machine learning methods.
- Entirely fictitious, it supposedly concerns the conditions that are suitable for playing some unspecified game.
- In general, instances in a dataset are characterized by the values of features, or *attributes,* that measure different aspects of the instance.
- In this case there are four attributes: *outlook, temperature, humidity,* and *windy.*
- The outcome is whether to play or not.

Weather data

Table 1.2 The weather data.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Rules

- A set of rules learned from this information — not necessarily a very good one — might look as follows:
  - If outlook = sunny and humidity = high then play = no
  - If outlook = rainy and windy = true then play = no
  - If outlook = overcast then play = yes
  - If humidity = normal then play = yes
  - If none of the above then play = yes

Decision List

- These rules are meant to be interpreted in order: the first one, then if it doesn't apply the second, and so on.
- A set of rules that are intended to be interpreted in sequence is called a *decision list.*
- Interpreted as a decision list, the rules correctly classify all of the examples in the table, whereas taken individually, out of context, some of the rules are incorrect.
- For example, the rule if humidity = normal then play = yes gets one of the examples wrong.
- The meaning of a set of rules depends on how it is interpreted — not surprisingly!

Numeric attributes

- In a slightly more complex form of the problem, two of the attributes — temperature and humidity — have numeric values.

- This means that any learning method must create inequalities involving these attributes rather than simple equality tests, as in the former case.

- This is called a *numeric-attribute problem* — in this case, a *mixed-attribute problem* because not all attributes are numeric.

Numeric attributes

Table 1.3 **Weather data with some numeric attributes.**

Outlook	Temperature	Humidity	Windy	Play
sunny	85	85	false	no
sunny	80	90	true	no
overcast	83	86	false	yes
rainy	70	96	false	yes
rainy	68	80	false	yes
rainy	65	70	true	no
overcast	64	65	true	yes
sunny	72	95	false	no
sunny	69	70	false	yes
rainy	75	80	false	yes
sunny	75	70	true	yes
overcast	72	90	true	yes
overcast	81	75	false	yes
rainy	71	91	true	no

- Now the first rule given earlier might take the following form:
  - If outlook = sunny and humidity > 83 then play = no

Classification and Association Rules

- The rules we have seen so far are *classification rules:* they predict the classification of the example in terms of whether to play or not.

- It is equally possible to disregard the classification and just look for any rules that strongly associate different attribute values.

- These are called *association rules.* Many association rules can be derived from the weather data in Table 1.2.

Association Rules

1. If temperature = cool then humidity = normal
2. If humidity = normal and windy = false then play = yes
3. If outlook = sunny and play = no then humidity = high
4. If windy = false and play = no then outlook = sunny and humidity = high.

- All these rules are 100% correct on the given data; they make no false predictions.
- The first two apply to four examples in the dataset, the third to three examples, and the fourth to two examples.
- There are many other rules: in fact, nearly 60 association rules can be found that apply to two or more examples of the weather data and are completely correct on this data.
- If you look for rules that are less than 100% correct, then you will find many more.

Association rules for contact lenses

```
If tear production rate = reduced then recommendation = none
If age = young and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = pre-presbyopic and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = presbyopic and spectacle prescription = myope and
   astigmatic = no then recommendation = none
If spectacle prescription = hypermetrope and astigmatic = no and
   tear production rate = normal then recommendation = soft
If spectacle prescription = myope and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = young and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = pre-presbyopic and
   spectacle prescription = hypermetrope and astigmatic = yes
   then recommendation = none
If age = presbyopic and spectacle prescription = hypermetrope
   and astigmatic = yes then recommendation = none
```

Figure 1.1 Rules for the contact lens data.

Decision Trees for Contact Lens data

- People frequently use machine learning techniques to gain insight into the structure of their data rather than to make predictions for new cases.

- In fact, a prominent and successful line of research in machine learning began as an attempt to compress a huge database of possible chess endgames and their outcomes into a data structure of reasonable size.

- The data structure chosen for this enterprise was not a set of rules but a decision tree.

Decision Trees for Contact Lens data

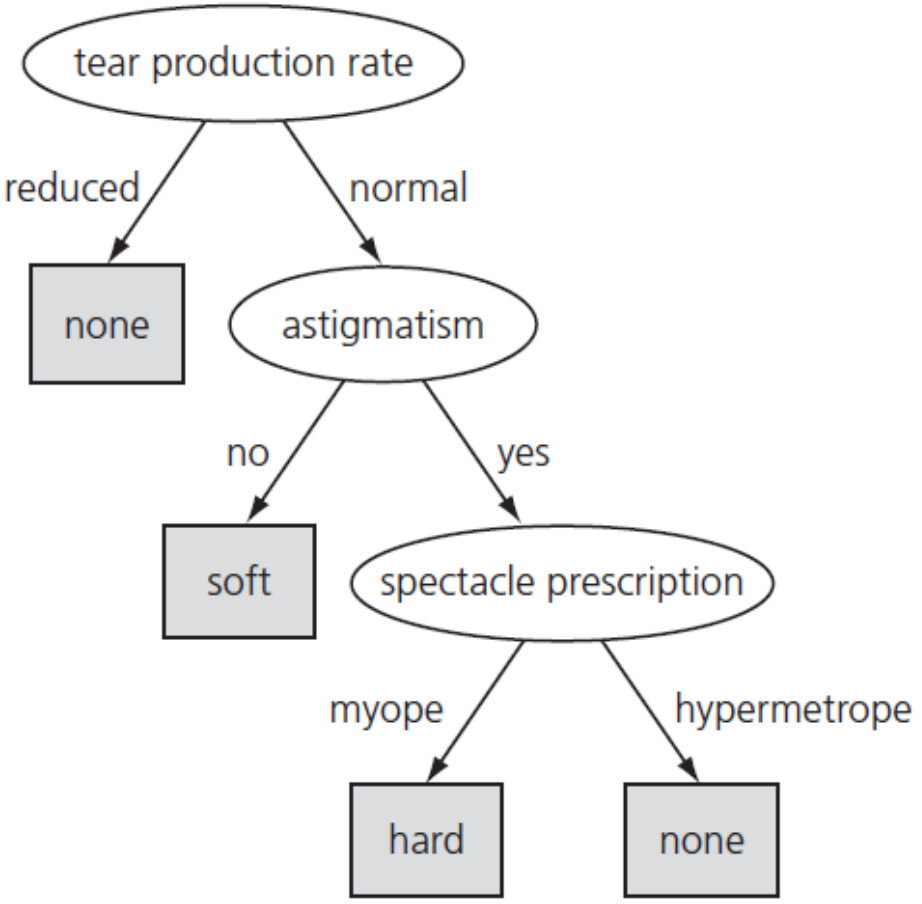


Figure 1.2 Decision tree for the contact lens data.

Irises: A classic numeric dataset

- The iris dataset, which dates back to seminal work by the eminent statistician R.A. Fisher in the mid-1930s and is arguably the most famous dataset used in data mining, contains 50 examples each of three types of plant: *Iris setosa, Iris versicolor,* and *Iris virginica.*

- There are four attributes: *sepal length, sepal width, petal length,* and *petal width* (all measured in centimeters).

- Unlike previous datasets, all attributes have values that are numeric.

Irises

Table 1.4 The iris data.

	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
1	5.1	3.5	1.4	0.2	*Iris setosa*
2	4.9	3.0	1.4	0.2	*Iris setosa*
3	4.7	3.2	1.3	0.2	*Iris setosa*
4	4.6	3.1	1.5	0.2	*Iris setosa*
5	5.0	3.6	1.4	0.2	*Iris setosa*
. . .					
51	7.0	3.2	4.7	1.4	*Iris versicolor*
52	6.4	3.2	4.5	1.5	*Iris versicolor*
53	6.9	3.1	4.9	1.5	*Iris versicolor*
54	5.5	2.3	4.0	1.3	*Iris versicolor*
55	6.5	2.8	4.6	1.5	*Iris versicolor*
. . .					
101	6.3	3.3	6.0	2.5	*Iris virginica*
102	5.8	2.7	5.1	1.9	*Iris virginica*
103	7.1	3.0	5.9	2.1	*Iris virginica*
104	6.3	2.9	5.6	1.8	*Iris virginica*
105	6.5	3.0	5.8	2.2	*Iris virginica*
. . .					

Rules for Irises data

- If petal length < 2.45 then Iris setosa
- If sepal width < 2.10 then Iris versicolor
- If sepal width < 2.45 and petal length < 4.55 then Iris versicolor
- If sepal width < 2.95 and petal width < 1.35 then Iris versicolor
- If petal length ≥ 2.45 and petal length < 4.45 then Iris versicolor
- If sepal length ≥ 5.85 and petal length < 4.75 then Iris versicolor
- These rules are very cumbersome, and we will see in the next lectures how more compact rules can be expressed that convey the same information.

CPU performance: Introducing numeric prediction

- Although the iris dataset involves numeric attributes, the outcome — the type of iris — is a category, not a numeric value.
- There are many problems for which the outcome and the attributes are numeric.
- One problem concerns the relative performance of computer processing power on the basis of a number of relevant attributes.

Predicting CPU performance

Table 1.5 The CPU performance data.

	Cycle time (ns) MYCT	Main memory (KB) Min. MMIN	Main memory (KB) Max. MMAX	Cache (KB) CACH	Channels Min. CHMIN	Channels Max. CHMAX	Performance PRP
1	125	256	6000	256	16	128	198
2	29	8000	32000	32	8	32	269
3	29	8000	32000	32	8	32	220
4	29	8000	32000	32	8	32	172
5	29	8000	16000	32	8	16	132
. . .							
207	125	2000	8000	0	2	14	52
208	480	512	8000	32	0	0	67
209	480	1000	4000	0	0	0	45

Regression

- The classic way of dealing with continuous prediction is to write the outcome as a linear sum of the attribute values with appropriate weights, for example:

  PRP = -55.9 + 0.0489 MYCT + 0.0153 MMIN + 0.0056 MMAX + 0.6410 CACH - 0.2700 CHMIN + 1.480 CHMAX.
- This is called a *regression equation*, and the process of determining the weights is called *regression*, a well-known procedure in statistics.
- However, the basic regression method is incapable of discovering nonlinear relationships (although variants do exist)
- In the iris and central processing unit (CPU) performance data, all the attributes have numeric values.
- Practical situations frequently present a mixture of numeric and nonnumeric attributes.

Labor negotiations: A more realistic example

- The labor negotiations dataset summarizes the outcome of Canadian contract negotiations in 1987 and 1988.
- It includes all collective agreements reached in the business and personal services sector for organizations with at least 500 members (teachers, nurses, university staff, police, etc.).
- Each case concerns one contract, and the outcome is whether the contract is deemed *acceptable* or *unacceptable.*
- The acceptable contracts are ones in which agreements were accepted by both labor and management.
- The unacceptable ones are either known offers that fell through because one party would not accept them or acceptable contracts that had been significantly perturbed to the extent that, in the view of experts, they would not have been accepted.

The labor negotiations data

Table 1.6 The labor negotiations data.

Attribute	Type	1	2	3	. . .	40
duration	years	1	2	3		2
wage increase 1st year	percentage	2%	4%	4.3%		4.5
wage increase 2nd year	percentage	?	5%	4.4%		4.0
wage increase 3rd year	percentage	?	?	?		?
cost of living adjustment	{none, tcf, tc}	none	tcf	?		none
working hours per week	hours	28	35	38		40
pension	{none, ret-allw, empl-cntr}	none	?	?		?
standby pay	percentage	?	13%	?		?
shift-work supplement	percentage	?	5%	4%		4
education allowance	{yes, no}	yes	?	?		?
statutory holidays	days	11	15	12		12
vacation	{below-avg, avg, gen}	avg	gen	gen		avg
long-term disability assistance	{yes, no}	no	?	?		yes
dental plan contribution	{none, half, full}	none	?	full		full
bereavement assistance	{yes, no}	no	?	?		yes
health plan contribution	{none, half, full}	none	?	full		half
acceptability of contract	{good, bad}	bad	good	good		good

Decision Tree for labor data

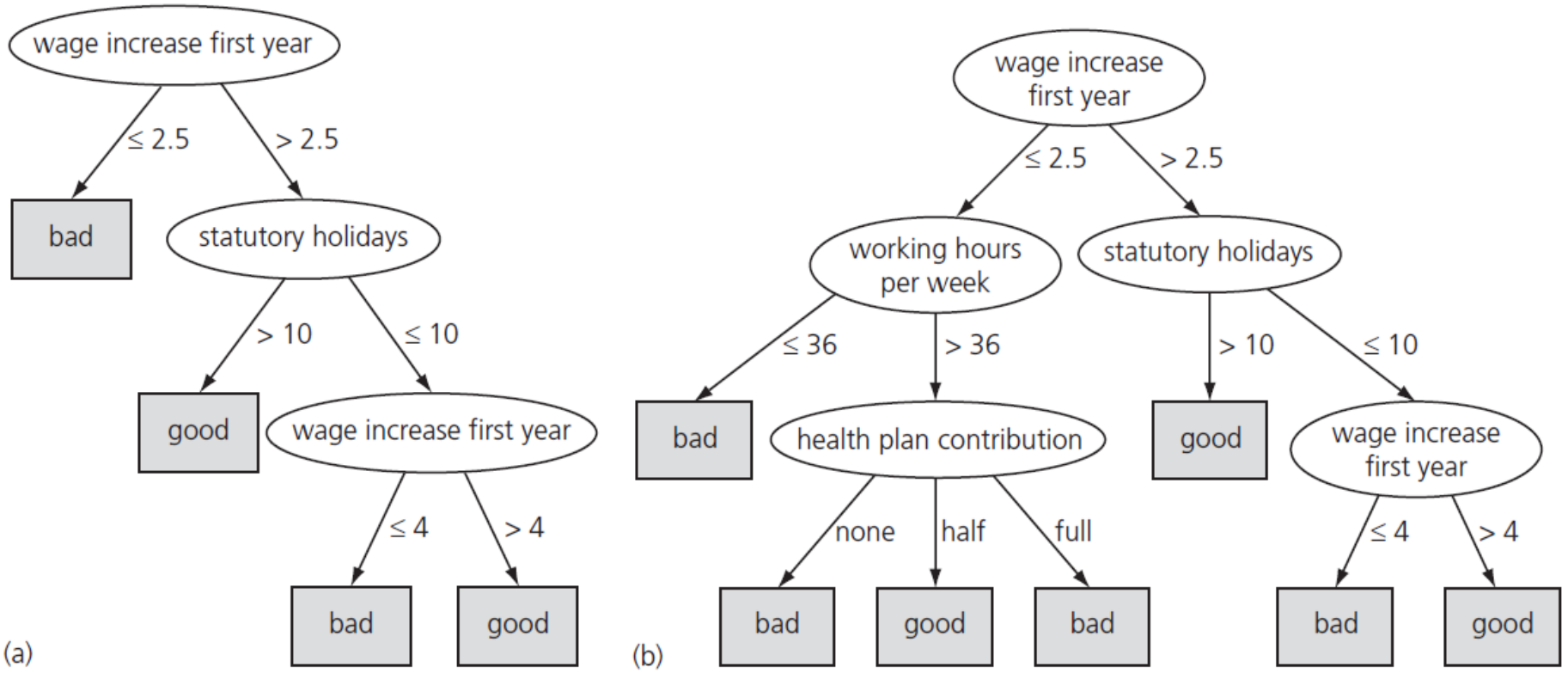


Figure 1.3 Decision trees for the labor negotiations data.

Soybean classification: A classic machine learning success

- An often-quoted early success story in the application of machine learning to practical problems is the identification of rules for diagnosing soybean diseases.
- The data is taken from questionnaires describing plant diseases.
- There are about 680 examples, each representing a diseased plant.
- Plants were measured on 35 attributes, each one having a small set of possible values.
- Examples are labeled with the diagnosis of an expert in plant biology: there are 19 disease categories.

Table 1.7 The soybean data.

	Attribute	Number of values	Sample value
Environment	time of occurrence	7	July
	precipitation	3	above normal
	temperature	3	normal
	cropping history	4	same as last year
	hail damage	2	yes
	damaged area	4	scattered
	severity	3	severe
	plant height	2	normal
	plant growth	2	abnormal
	seed treatment	3	fungicide
	germination	3	less than 80%
Seed	condition	2	normal
	mold growth	2	absent
	discoloration	2	absent
	size	2	normal
	shriveling	2	absent
Fruit	condition of fruit pods	3	normal
	fruit spots	5	—
Leaf	condition	2	abnormal
	leaf spot size	3	—
	yellow leaf spot halo	3	absent
	leaf spot margins	3	—
	shredding	2	absent
	leaf malformation	2	absent
	leaf mildew growth	3	absent
Stem	condition	2	abnormal
	stem lodging	2	yes
	stem cankers	4	above soil line
	canker lesion color	3	—
	fruiting bodies on stems	2	present
	external decay of stem	3	firm and dry
	mycelium on stem	2	absent
	internal discoloration	3	none
	sclerotia	2	absent
Root	condition	3	normal
Diagnosis		19	diaporthe stem canker

Rules for Soybean classification

If leaf condition is normal and

stem condition is abnormal and
stem cankers is below soil line and
canker lesion color is brown]

then

diagnosis is rhizoctonia root rot

If leaf malformation is absent and

stem condition is abnormal and
stem cankers is below soil line and
canker lesion color is brown

then

diagnosis is rhizoctonia root rot

Performance of rules

- At the same time, the plant pathologist who had produced the diagnoses was interviewed, and his expertise was translated into diagnostic rules.

- Surprisingly, the computer-generated rules outperformed the expert-derived rules on the remaining test examples.

- They gave the correct disease top ranking 97.5% of the time compared with only 72% for the expert-derived rules.

Fielded applications

- The examples that we opened with are speculative research projects, not production systems.
- And the preceding illustrations are *toy problems*: they are deliberately chosen to be small so that we can use them to work through algorithms later in the book.
- Where's the beef? We will investigate some applications of machine learning that have actually been put into use.
- This course also describes the use of learning systems to gain knowledge from decision structures that are inferred from the data.
- In three of the examples that follow, the fact that the *decision structure is comprehensible* is a key feature in the successful adoption of the application.

Decisions involving judgment

- When you apply for a loan, you have to fill out a questionnaire that asks for relevant financial and personal information. This information is used by the loan company as the basis for its decision as to whether to lend you money.
- Such decisions are typically made in two stages. First, statistical methods are used to determine clear “accept” and “reject” cases.
- The *remaining borderline cases* are more difficult and call for human judgment.
- For example, one loan company uses a statistical decision procedure to calculate a numeric parameter based on the information supplied in the questionnaire.
- Applicants are accepted if this parameter exceeds a preset threshold and rejected if it falls below a second threshold. This accounts for 90% of cases, and the remaining 10% are referred to loan officers for a decision.

Decisions involving judgment

- On examining historical data on whether applicants did indeed repay their loans, however, it turned out that *half of the borderline applicants who were granted loans actually defaulted*.
- Although it would be tempting simply to deny credit to borderline customers, credit industry professionals pointed out that if only their repayment future could be reliably determined it is precisely these customers whose business should be gained.
- This is what they did with machine learning => next slide.

Machine Learning for bank loans

- The input was 1000 training examples of borderline cases for which a loan had been made, that specified whether the borrower had finally paid off or defaulted.
- For each training example*, about 20 attributes* were extracted from the questionnaire, such as age, years with current employer, years at current address, years with the bank, and other credit cards possessed.
- A machine learning procedure was used to produce a small set of classification rules that made correct predictions on two-thirds of the borderline cases in an independently chosen test set.
- *Not only did these rules improve the success rate of the loan decisions, but the company also found them attractive because they could be used to explain to applicants the reasons behind the decision.*
- Although the project was an exploratory one that took only a small development effort, the loan company was apparently so pleased with the result that the rules were put into use immediately.

Other ML applications

- There are countless other applications of machine learning. We briefly mention a few more areas to illustrate the breadth of what has been done
- **Screening images**
  - detect oil slicks from satellite images
- **Load forecasting**
  - In the electricity supply industry, determine future demand for power as far in advance as possible
- **Diagnosis**
  - Diagnosis is one of the principal application areas
- **Marketing and sales**
  - Predicting sales for season or region

Other ML applications

Tweaking control parameters

- Separating crude oil from natural gas is an essential prerequisite to oil refinement, and controlling the separation process is a tricky job.
  - British Petroleum used machine learning to create rules for setting the parameters.
  - This now takes just 10 minutes, whereas previously human experts took more than a day.
- Westinghouse faced problems in their process for manufacturing nuclear fuel pellets and used machine learning to create rules to control the process.
  - This was reported to save them more than $10 million per year (in 1984).
- The Tennessee printing company R.R. Donnelly applied the same idea to control rotogravure printing presses to reduce artifacts caused by inappropriate parameter settings, reducing the number of artifacts from more than 500 each year to less than 30.

Other ML applications

Customer Support

- In the realm of customer support and service, we have already described adjudicating loans, and marketing and sales applications.
- Another example arises when a customer reports a telephone problem and the *company must decide what kind of technician to assign to the job*.
- An expert system developed by Bell Atlantic in 1991 to make this decision was replaced in 1999 by a set of rules learned using machine learning, which saved more than $10 million per year by making fewer incorrect decisions.

Scientific Applications

- In biology, machine learning is used to help identify the thousands of genes within each new genome.
- In biomedicine, it is used to predict drug activity by analyzing not just the chemical properties of drugs but also their three-dimensional structure. This accelerates drug discovery and reduces its cost.
- In astronomy, machine learning has been used to develop a fully automatic cataloguing system for celestial objects that are too faint to be seen by visual inspection.
- In chemistry, it has been used to predict the structure of certain organic compounds from magnetic resonance spectra.

- *In all these applications, machine learning techniques have attained levels of performance — or should we say skill? — that rival or surpass human experts.*

Machine Learning and the Design Principles of a Learning System

Quotes about ML

- “A breakthrough in machine learning would be worth ten Microsofts” (Bill Gates, Chairman, Microsoft)

- Peter Norvig
  - The head of research of Google
  - Author of *Artificial Intelligence: A Modern Approach*
  - http://www.norvig.com/

Machine Learning

- Machine learning is a multidisciplinary field and it is concerned with the question of how to construct computer programs that automatically improve with experience.
- In recent years many successful machine learning applications have been developed, ranging from:
  - data-mining programs that learn to detect fraudulent credit card transactions
  - information-filtering systems that learn users' reading preferences
  - autonomous vehicles that learn to drive on public highways.
- At the same time, there have been important advances in the theory and algorithms that form the foundations of this field.

ML initial achievements

- Some examples of initial achievements of ML are:
  - programs have been developed that successfully learn to recognize spoken words (Waibel 1989; Lee 1989),
  - predict recovery rates of pneumonia patients (Cooper et al. 1997)
  - detect fraudulent use of credit cards
  - drive autonomous vehicles on public highways (Pomerleau 1989)
  - play games such as backgammon at levels approaching the performance of human world champions (Tesauro 1992, 1995).

Learning to recognize spoken words

- All of the most successful speech recognition systems employ machine learning in some form.
- For example, the Sphinx system (e.g., Lee 1989) learns speaker-specific strategies for recognizing the primitive sounds (phonemes) and words from the observed speech signal.
- Neural network learning methods (e.g., Waibel et al. 1989) and methods for learning hidden Markov models (e.g., Lee 1989) are effective for automatically customizing to individual speakers, vocabularies, microphone characteristics, background noise, etc.
- Similar techniques have potential applications in many signal-interpretation problems.

Learning to drive an autonomous vehicle

- Machine learning methods have been used to train computer-controlled vehicles to steer correctly when driving on a variety of road types.

- For example, the ALVINN system (Pomerleau 1989) has used its learned strategies to drive unassisted at 70 miles per hour for 90 miles on public highways among other cars.

- Similar techniques have possible applications in many sensor-based control problems.

Demo of an autonomous vehicle

- Autonomous vehicle trained to drive
  - Video

Learning to classify new astronomical structures

- Machine learning methods have been applied to a variety of large databases to learn general regularities implicit in the data.

- 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 (Fayyad et al. 1995).

- This system is now used to automatically classify all objects in the Sky Survey, which consists of three terrabytes of image data.

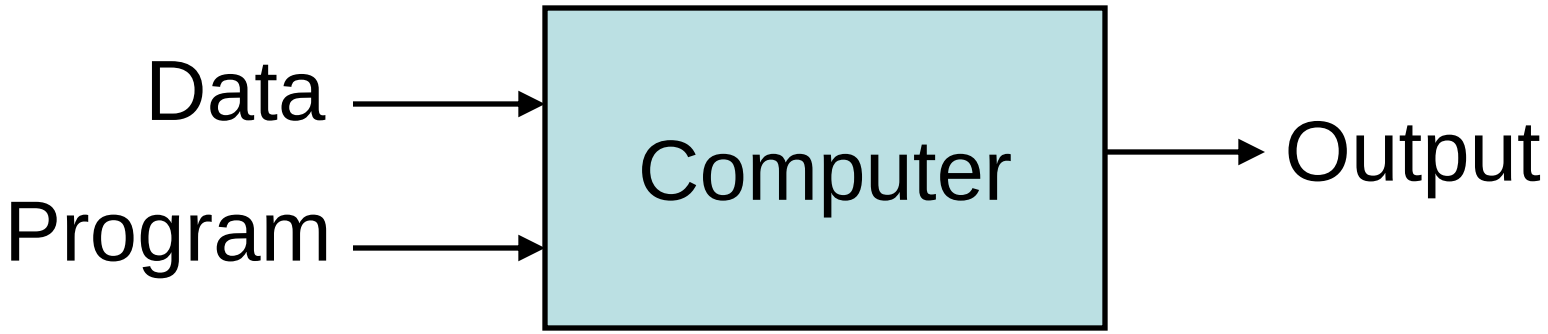
Learning to play world-class backgammon

- The most successful computer programs for playing games such as backgammon are based on machine learning algorithms.
- For example, the world's top computer program for backgammon, TD-Gammon (Tesauro 1992, 1995), learned its strategy by playing over one million practice games against itself.
- It now plays at a level competitive with the human world champion.
- Similar techniques have applications in many practical problems where very large search spaces must be examined efficiently.

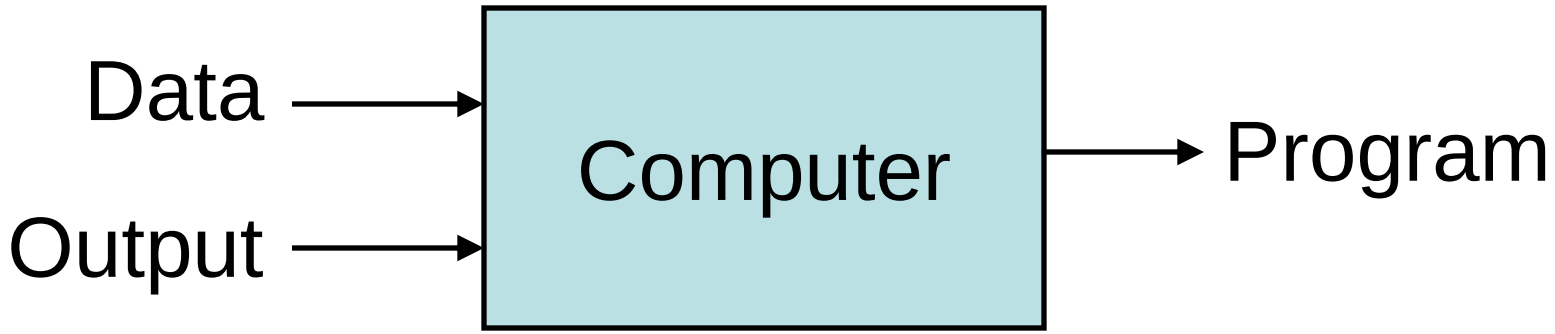
Current and future application areas of ML

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- And many other areas…

Traditional Programming



Machine Learning



From P. Domingos, CSD-UW

Programming computers to learn

- If we could understand how to program computers to learn — to improve automatically with experience — the impact would be dramatic.
- Imagine:
  - computers learning from medical records which treatments are most effective for new diseases,
  - houses learning from experience to optimize energy costs based on the particular usage patterns of their occupants, or
  - personal software assistants learning the evolving interests of their users in order to highlight especially relevant stories from the online morning newspaper.

ML: a multidisciplinary field

- Artificial intelligence

  Learning symbolic representations of concepts. Machine learning as a search problem. Learning as an approach to improving problem solving. Using prior knowledge together with training data to guide learning.

- Bayesian methods

  Bayes' theorem as the basis for calculating probabilities of hypotheses. The naive Bayes classifier. Algorithms for estimating values of unobserved variables.

- Computational complexity theory

  Theoretical bounds on the inherent complexity of different learning tasks, measured in terms of the computational effort, number of training examples, number of mistakes, etc. required in order to learn.

- Control theory

  Procedures that learn to control processes in order to optimize predefined objectives and that learn to predict the next state of the process they are controlling.

ML: a multidisciplinary field

- Information theory

  Measures of entropy and information content. Minimum description length approaches to learning. Optimal codes and their relationship to optimal training sequences for encoding a hypothesis.

- Philosophy

  Occam's razor, suggesting that the simplest hypothesis is the best. Analysis of the justification for generalizing beyond observed data.

- Psychology and neurobiology

  The power law of practice, which states that over a very broad range of learning problems, people's response time improves with practice according to a power law. Neurobiological studies motivating artificial neural network models of learning.

- Statistics

  Characterization of errors (e.g., bias and variance) that occur when estimating the accuracy of a hypothesis based on a limited sample of data. Confidence intervals, statistical tests.

Well-posed Learning Problems

- *Definition: 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.*

Well-posed Learning Problems

- For example, 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
  - the measure of performance to be improved
  - and the source of experience

A checkers learning problem

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

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.

A robot driving learning problem

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

Choosing the Training Experience

- The first design choice we face is to choose the type of training experience from which our system will learn.

- *The type of training experience available can have a significant impact on success or failure of the learner.*

- One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.

Direct or indirect feedback

- In learning to play checkers, the system might learn from direct training examples consisting of individual checkers board 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 this later case, information about the correctness of specific moves early in the game must be inferred indirectly from the fact that the game was eventually won or lost.
- Here the learner faces an additional problem of credit assignment, or determining the degree to which each move in the sequence deserves credit or blame for the final outcome.
- Credit assignment can be a particularly difficult problem because the game can be lost even when early moves are optimal, if these are followed later by poor moves.
- *Hence, learning from direct training feedback is typically easier than learning from indirect feedback.*

Controlling the sequence of training examples

- A second important attribute of the training experience is the degree to which the learner controls the sequence of training examples.
- For example, 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.

Distribution of examples

- A third important attribute of the training experience is how well it represents the distribution of examples over which the final system performance P must be measured.
- In general, learning is most reliable when the training examples follow a distribution similar to that of future test examples.
- In our 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 an 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.

Specifying the task

- To proceed with our design, let us decide that our system will train by playing games against itself.

- This has the advantage that no external trainer need be present, and it therefore allows the system to generate as much training data as time permits.

- Let us now develop a fully specified learning task, => next slide.

The checkers game

- English draughts (British English) or checkers (American English and Canadian English), also called American checkers or straight checkers is a form of the board game draughts.
- Unlike international draughts, it is played on an eight-by-eight squared board (with sixty-four total squares) with twelve pieces on each side.
- The pieces move and capture diagonally.
- They may only move forward until they reach the opposite end of the board, when they are crowned or kinged and may henceforth move and capture both backward and forward.

The checkers learning problem

- 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

Choosing the Target Function

- The next design choice is to determine exactly what type of knowledge will be learned and how this will be used by the performance program.
- Let us begin with a checkers-playing program that can 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.
- This learning task is representative of a large class of tasks for which the legal moves that define some large search space are known a priori, but for which the best search strategy is not known.
- Many optimization problems fall into this class, such as the problems of scheduling and controlling manufacturing processes where the available manufacturing steps are well understood, but the best strategy for sequencing them is not.

Choosing the Target Function

- Given this setting where we must learn to choose among the legal moves, the most obvious choice for the type of information to be learned is a program, or function, that chooses the best move for any given board state.
- 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.
- Throughout our course we will find it useful to reduce the problem of improving performance P at task T to the problem of learning some particular target function such as ChooseMove.
- The choice of the target function will therefore be a key design choice.

Evaluation function

- Although ChooseMove is an obvious choice for the target function in our example, this function will turn out to be very difficult to learn given the kind of indirect training experience available to our system.
- An alternative target function — and one that will turn out to be easier to learn in this setting — 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 (we use R to denote the set of real numbers).
- We intend for this target function V to assign higher scores to better board states.
- If the system can successfully learn such a target function V, then it can easily use it to select the best move from any current board position.
- This can be accomplished by generating the successor board state produced by every legal move, then using V to choose the best successor state and therefore the best legal move.

Evaluation Function

- 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, 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 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).

Operational Definition

- While this recursive definition specifies a value of V(b) for every board state b, this definition is not usable by our checkers player because it is not efficiently computable.
- Except for the trivial cases (cases 1-3) in which the game has already ended, determining the value of V(b) for a particular board state requires (case 4) a lot of computations.
- Because this definition is not efficiently computable by our checkers playing program, we say that it is a *nonoperational definition*.
- The goal of learning in this case is to discover an operational description of V:
  - a description that can be used by the checkers-playing program to evaluate states and select moves within realistic time bounds.

Function Approximation

- Thus, we have reduced the learning task in this case to the problem of *discovering an operational description of the ideal target function V*.
- It may be very difficult in general to learn such an operational form of V perfectly.
- In fact, we often expect learning algorithms to acquire only some *approximation to the target function*, and for this reason the process of learning the target function is often called function approximation.
- In the current discussion we will use the symbol V^ to refer to the function that is actually learned by our program, to distinguish it from the ideal target function V.

Choosing a Representation for the Target Function

- Now that we have specified the ideal target function V, we must choose a representation that the learning program will use to describe the function V that it will learn.
- We again have many options.
  - We could, for example, allow the program to represent V using a large table with a distinct entry specifying the value for each distinct board state.
  - Or we could allow it to represent V using a collection of rules that match against features of the board state, or a quadratic polynomial function of predefined board features, or an 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.

Representing the target function

- Let us choose a simple representation: for any given board state, the function V 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

Representing the target function

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

  $\hat{V}(b) = W_0 + W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6$

- where W_0 through W_6 are numerical coefficients, or weights, to be chosen by the learning algorithm.

- Learned values for the weights W_1 through W_6 will determine the relative importance of the various board features in determining the value of the board, whereas the weight W_0 will provide an additive constant to the board value.

Partial design of a checkers learning program

The elaborated learning task is now:

- 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

  $\hat{V}(b) = W_0 + W_1X_1 + W_2X_2 + W_3X_3 + W_4X_4 + W_5X_5 + W_6X_6$

Learning strategy = Learning Functions

- 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.

Choosing a Function Approximation Algorithm

- In order to learn the target function V 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 $(b, V_{train}(b))$.
  - For instance, one training example describes a board state b in which black has won the game (note $X_2 = 0$ indicates that red has no remaining pieces) and for which the target function value $V_{train}(b)$ is therefore +100.
- We will describe a procedure that first derives such training examples from the indirect training experience available to the learner, then adjusts the weights W_i to best fit these training examples.

Estimating Training Values

- Recall that according to our formulation of the learning problem, the only training information available to our learner is whether the game was eventually won or lost.
- On the other hand, we require training examples that assign specific scores to specific board states.
- While it is easy to assign a value to board states that correspond to the end of the game, it is less obvious how to assign training values to the more numerous intermediate board states that occur before the game's end.
- Of course the fact that the game was eventually won or lost does not necessarily indicate that every board state along the game path was necessarily good or bad.
- For example, even if the program loses the game, it may still be the case that board states occurring early in the game should be rated very highly and that the cause of the loss was a subsequent poor move.

Rule for estimating training values

- Despite the ambiguity inherent in estimating training values for intermediate board states, one simple approach has been found to be surprisingly successful.
- This approach is to assign the training value of $V_{train}(b)$ for any intermediate board state b to be:
  - $\hat{V}(Successor(b))$ where $\hat{V}$ is the learner's current approximation to V and where 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:
- Rule for estimating training values.

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

Adjusting the weights

- All that remains is to specify the learning algorithm for choosing the weights w, to best fit the set of training examples $\{\{b, V_{train}(b)\}\}$.
- As a first step we must define what we mean by the best fit to the training data.
- One common approach is to define the best hypothesis, or set of weights, as that which minimizes the squared error E between the training values and the values predicted by the hypothesis V:

$$E \equiv \sum_{\langle b, V_{train}(b)\rangle \in \text{ training examples}} (V_{train}(b) - \hat{V}(b))^2$$

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

Choosing the algorithm for adjusting weights

- Several algorithms are known for finding weights of a linear function that minimize E defined in this way.
- In our case, we require an algorithm that will incrementally refine the weights as new training examples become available and that will be robust to errors in these estimated training values.
- One such algorithm is called the *least mean squares*, or LMS training rule.
- For each observed training example it adjusts the weights a small amount in the direction that reduces the error on this training example.

The LMS Algorithm

- LMS weight update rule.
- For each training example (b, $V_{train}(b)$)

  Use the current weights to calculate $\hat{V}(b)$

  For each weight W_i, update it as

  $$W_i \leftarrow W_i + \eta (V_{train}(b) - \hat{V}(b)) X_i$$

- Here η is a small constant (e.g., 0.1) that moderates the size of the weight update.
- Surprisingly, in certain settings this simple weight-tuning method can be proven to converge to the least squared error approximation to the V_{train} values.

The Final Design

- The final design of our checkers learning system can be naturally described by four distinct program modules that represent the central components in many learning systems.

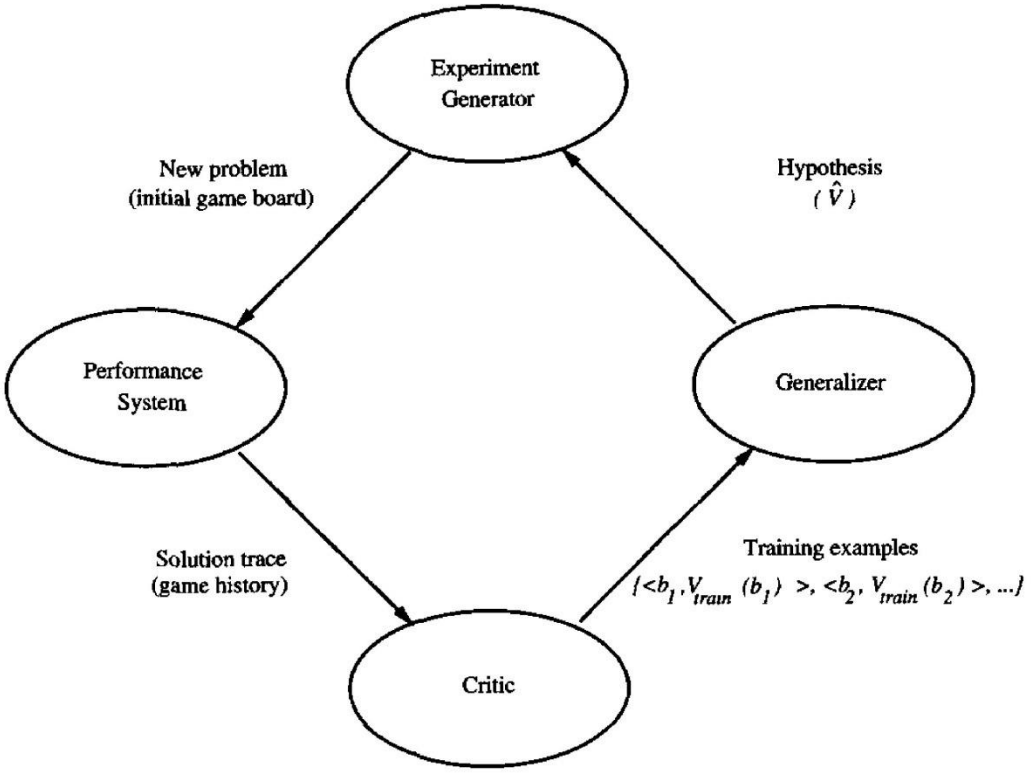


FIGURE 1.1
Final design of the checkers learning program.

Performance System

- 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 V evaluation function.
- Therefore, we expect its performance to improve as this evaluation function becomes increasingly accurate.

The Critic

- 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 V_{train} of the target function value for this example.
- In our example, the Critic corresponds to the training rule given by Equation (1.1).

The Generalizer

- 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 V described by the learned weights W_0,..,W_6.

Experiment Generator

- 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.

Characterizing a ML system

- Together, the design choices we made for our checkers program produce specific instantiations for the performance system, critic, generalizer, and experiment generator.

- Many machine learning systems can be usefully characterized in terms of these four generic modules.

Summary of Design Choices

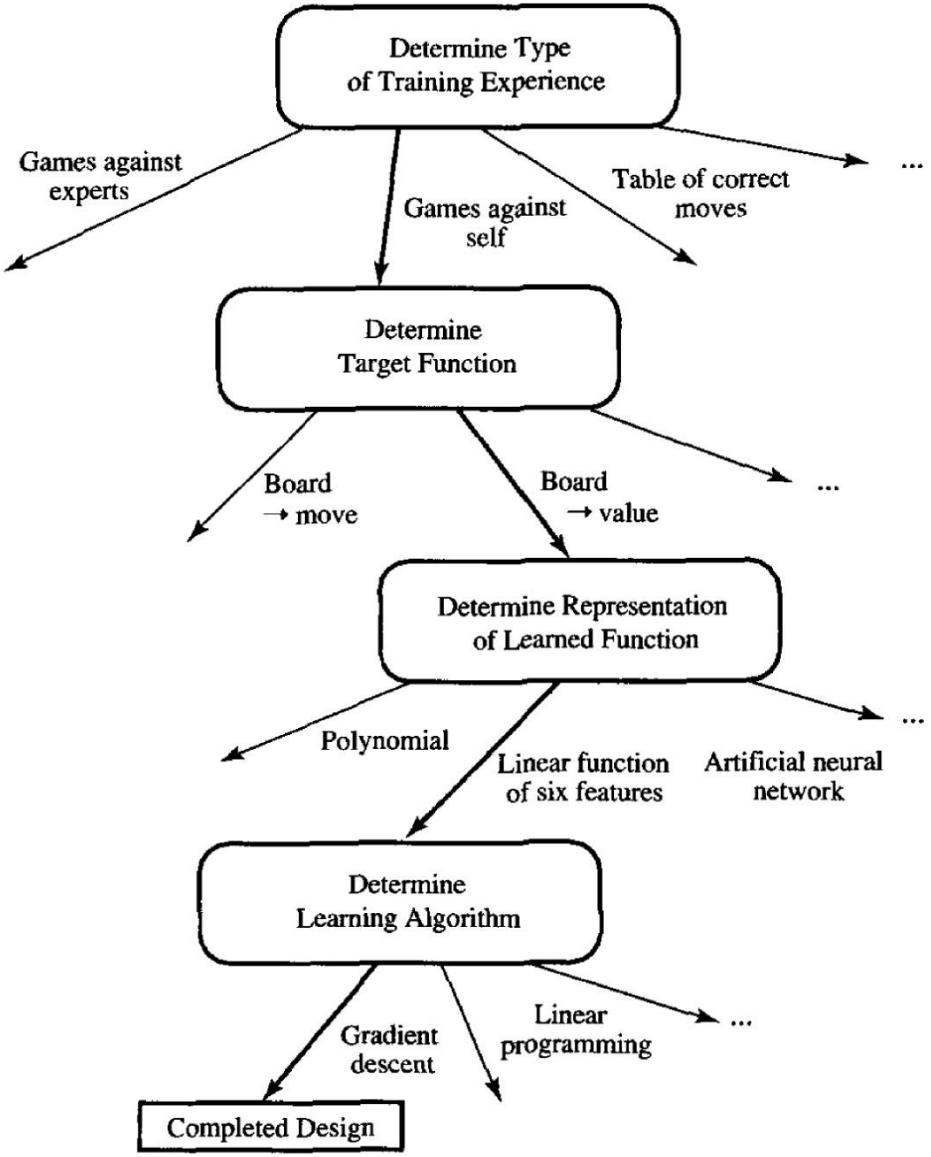


FIGURE 1.2
Summary of choices in designing the checkers learning program.

Components of ML algorithms

- Machine learning algorithms have three components:
  - Representation
  - Evaluation
  - Optimization

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

Types of Learning

- Supervised (inductive) learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions
    - Robots are trained this way (just as animals)

Issues in ML

- 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?

Issues in ML

- 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?

Issues in ML

- 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 problem
  - 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?

Data Mining and Machine Learning in Practice

- Understanding domain, prior knowledge, and goals
- Data integration, selection, cleaning, pre-processing, etc.
- Learning models
- Interpreting results
- Consolidating and deploying discovered knowledge
- Loop

Readings

- Machine Learning. T. Mitchell
  - Chapter 1

- Data Mining. Witten and Frank
  - Chapter 1