Knowledge Acquisition and Machine Learning

- The knowledge bottleneck problem
- Three types of knowledge acquisition
- The knowledge engineering process
- Knowledge acquisition by interview
- Interactive knowledge transfer
- Rule Induction
- Knowledge refinement
- Learning strategies
The Knowledge Bottleneck Problem

• Knowledge is power!!
• The core of AI systems is knowledge.
• Knowledge acquisition normally takes three fourths of the development time of knowledge-based systems.

• Main difficulty:
  – Knowledge (especially heuristics) is easy to use, but difficult to articulate and explain.
  – Two examples: handwriting recognition and theorem proving.
Three types of knowledge acquisition

• Knowledge Acquisition by Interview: Experts hold knowledge, knowledge engineers (designers of knowledge based systems) acquire and program it into a program.

• Interactive Knowledge Transfer: Experts hold knowledge and the knowledge is transformed into a system via an interactive computer interface. The interface must have a specific representation scheme and be well structured.

• Automatic Knowledge Acquisition: Experts don’t have the knowledge readily usable, but can provide case studies of their skill at work. A computer program is designed to induce knowledge (e.g., rules) from the case studies.

  – Rule induction is the main stream of automatic knowledge acquisition.
The knowledge engineering process

- AI programming requires **a nontranditional development cycle based on early prototyping and incremental revision of the knowledge.**

  - This exploratory programming methodology is complicated by the interaction between the knowledge engineer (expert system designer/AI programmer) and the domain expert.

- Knowledge-based systems need never be considered “finished”. A large knowledge base will always have limitations and improvements.

- Knowledge acquisition consists of the acquisition of a knowledge base and its refinement.
Knowledge Acquisition by Interview

• Difficulties:

  Knowledge engineer:
  (1) Questionnaire?
  (2) Summarize the interview: How to fill in the details?

  Expert:
  (1) Easy to use, difficult to say.
  (2) Language: “big mouse” and “small elephant”.
  (3) Logic: inclusive and exclusive or, if and only if.
  (4) CF values.

  Problems: Interviews are unstructured, boring.

• A possible solution: Knowledge engineers have knowledge about the problem domain instead of relying on the experts only.
Interactive Knowledge Transfer

- An expert interacts with a computer program and inputs domain knowledge via the dialogue.

  TEIRESIAS (in MYCIN): When some error is found, TEIRESIAS’s approach is to ask a focused question: “What do you know that the expert system does not know that makes your expert diagnosis different in this case?”

  SIKT (in KEshell): The target of the program is to build a domain network. Each root in the network denotes a problem or goal (e.g., a disease in a medical domain) and each leaf denotes an input (e.g., a symptom) from the user or databases. The procedure works in a top-down way. It first asks for all the roots and then iteratively acquires the nodes in the network until all the leaves have been input.

- Problems: Knowledge representation is fixed in the program. If it is not suitable to describe the knowledge the expert is describing, this way will not work.
Rule Induction

- Deduction: General $\rightarrow$ specific, truth-preserving.
  $P(X) \rightarrow Q(X), P(a) \rightarrow Q(a)$

- Induction: Specific $\rightarrow$ general.
  Mathematical induction: truth-preserving.
  
  \[ 1 + 2 + \ldots + n = \frac{n \times (1 + n)}{2} \]

  
  \[ P(a), P(b), P(c), \sim (\sim P(\_)) \rightarrow P(X) \]

- Abduction: $P(a) \rightarrow Q(b), Q(b) \rightarrow P(a)$
  Inexact reasoning.

- Induction to be covered in more detail (with examples) in “Building Intelligent Learning Database Systems”...
Knowledge Refinement

- After a knowledge base has been built, we would like to add some new knowledge and modify the existing content:

- Problems: When new knowledge is coming, we need to check for and remove inconsistency and redundancy.
  
  - **Assimilation**: Assume the existing content of the knowledge base is correct, only when the new knowledge is consistent with the existing knowledge base will it be integrated.
  
  - **Accommodation**: Assume the new knowledge is more reliable, we need to remove any possible inconsistency from the existing knowledge base before the new knowledge is put in.
  
  - **Generalization**: Use more general concepts or remove some of the conjunctions from the LFH of a rule to make the rule more general.
  
  - **Specialization**: Use more specific concepts or add more conjunctions to the LFH of a rule to make the rule more specific.
Machine Learning: Automatic Knowledge Acquisition

- The **source information**: input to the learning engine.
- The **learning engine**: carries out the learning task and produces knowledge (e.g., rules) for the knowledge base.
- The **performance engine** makes sure that the knowledge produced is useful. In the case of an expert system, the performance engine is its inference engine.
- The **feedback** from the performance engine produces feedback to the learning engine decides between the following actions:
  - continue the learning process,
  - ask for more information from the source information, and
  - stop.
Learning Strategies: Overview (1)

- The **source information** is the most important factor affecting the design of learning systems. It determines the task to be carried out in the **learning engine**. In general, different learning strategies have different levels of source information:

  - **Rote learning**. The input to the learning engine is exactly what is needed in the knowledge base. All the learning engine needs to do is **remember** and organize the input in the knowledge base.

    There are 2 different kinds of cases with rote learning. One is that the source information provides the exact knowledge for the performance engine. The other is that the learning engine remembers each specific user’s problem (which is not mentioned in the diagram and is supposed to be solved by the performance engine) and the problem solving process or outcome of the performance engine in the knowledge base.

    Mechanical (or interview-based) knowledge acquisition falls into this category: knowledge engineers (the developers/programmers of expert systems) elicit knowledge from experts and then store it via an editor (the simplest learning engine) in the knowledge base.
Learning Strategies: Overview (2)

- **Learning by being told** or **learning by advice taking**. The information provided by the source information is too abstract or too general to be adopted in the knowledge base directly, and thus, the learning engine needs to transform the input into a form readily usable by the performance engine, and fill in the details.

Interactive (or semi-automatic) knowledge acquisition normally makes use of both rote learning and learning by being told. Domain experts hold knowledge, most of which is readily usable by the performance engine, and input the knowledge out of the direct dialogue with an intelligent editor (the learning engine). What the editor needs to do is transform the experts’ input into the knowledge base and check inconsistencies.
Learning Strategies: Overview (3)

- **Learning from examples.** The input to the learning engine is too specific and too detailed cases, e.g., medical cases. The learning engine is to generate general rules which are applicable to these cases and other unseen ones as well. For example, the rules can be the relationships between medical symptoms and certain diseases.

  According to the format of the examples and whether the knowledge base has already possessed relevant domain knowledge, it can be further divided into many learning paradigms: supervised learning (like rule induction, where each example is classified) and unsupervised learning (no clear classification is available).

- **Learning by analogy** or **case-based reasoning.** Given as input a collection of past cases, which are relevant to the new problem to be solved by the performance engine, and the past experiences of solving these past cases, the learning engine is supposed to find out the hypotheses and/or strategies for the new problem.