

CSCI 400 Artificial Intelligence
David Keil, Framingham State University

Topic 5: Supervised learning and natural language

1. Symbol-based learning
2. Connectionist learning
3. Evolutionary computation
4. Natural-language processing

Inquiry

- What type of learning occurs in static environments?

Objectives

- 5a. Describe and use methods of symbol-based *supervised learning*
- 5b. Describe the *connectionist* approach to AI
- 5c. Describe evolutionary computation
- 5d. Explain concepts of *natural-language processing*

Learning

- *Principle*: use percepts to improve future ability to act
- *Useful to learn*:
 - State-action mapping
 - inference of what the state of the environment is, from percept
 - Model of how environment changes and results of possible actions
 - Utilities of states
 - Action values

Learning

- We classify learning as *supervised* or *interactive*
- Three types:
 - *Symbol-based*; e.g., generalizations
 - *Connectionist*: neural networks, inspired by brain's changing of thresholds for firing
 - *Emergent*: evolutionary-inspired techniques (topic 7)

Learning

- *Definition* (Simon, 1983): “Any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population.”
- Hence learner must *generalize* from limited experience

Types and features of learning

- Learning may be inductive generalization, explanation based, supervised/unsupervised, or by reinforcement
- All learning may be seen as a state-space search for a set of appropriate actions to use in response to percepts

Machine learning

- An agent *learns* from experience E , w.r.t. task set T , using performance measure P , iff agent's performance measure under P , on tasks in T , improves with experience E
- *Example: Driving*
 - T : driving using vision sensors
 - P : average distance without collision
 - E : images (percepts) paired with steering commands (actions)

Rote memorizing vs. learning

- Some humans “learn” this way
- It involves reading or listening and choosing some words to remember
- It is more processing-intensive than loading data from a device or port
- But it isn't what AI research calls *learning*
- Constructivist theory of learning: (human) learning is the learner's construction of knowledge

Concept learning

- *Domain*: a set X of possible instances of a concept
- Target concept: $c : X \rightarrow \{ T, F \}$
- *Training examples*: a set of pairs (x, y) , with $x \in X, y = c(x)$
- Where $y = F$, example is “negative”
- Where $y = T$, example is “positive”
- *Goal of learning*: to find c by generalizing from training examples
- *Method*: Search for a hypothesis h s.t. $h(x) = c(x)$ for all $x \in X$

1. Symbol-based learning

Inductive inference

- A deterministic supervised-learning algorithm
- Given examples of concept c , return an approximation h of c
- h is a hypothesis; it may be subject to improvement
- Inductive inference applies *Ockham's Razor*: prefer the simplest hypothesis that fits the data

Set cover based abduction

- Abduction is “reasoning to the best explanation” for the presence of an item of data
- If $(p \rightarrow q) \wedge q$, then a plausible *explanation* for q is p
- *Set cover approach*: a covering of predicates describing observations by use of predicates denoting hypotheses
- *Basis*: let $R \subseteq \text{Hypotheses} \times \text{Observations}$ where R is causal

Set cover example

- Let S be all possible observations; let T be ...
- Let set $S_2 \subseteq S$ be a set of observations,
- Let H be a set of hypotheses sufficient to explain S_2
- *Set cover*: Given $R \subseteq S \times T$, $S' \subseteq S$ is a set cover w.r.t. R if every t in T is in a pair that is in R , i.e.:

$$(\forall t \in T) (\exists s \in S') (s, t) \in R$$

- [Need a more concrete example; want to identify T]

Logic-based abduction

- An *abductive explanation* of a set of observations O may be defined as a minimal set of hypotheses H consistent with some set K of background domain knowledge, where $H \cup K$ entails O
- $\text{Abduce}(K, O) = H$ iff
 $\neg(O \rightarrow K) \wedge (O \rightarrow H) \wedge \text{consistent}(H \cup K) \wedge$
 (no subset of H has the previous three properties)
- *Abduction* backchains from conclusions (observations) to antecedents (hypotheses)

Abductive inference: $((p \rightarrow q) \wedge q) \rightarrow p$

- Unsound because not always true; but quite often applicable
- *Example*: excess sugar produces cavities, and I have a cavity, so I must be eating excess sugar
- It is valid to reason this way with a certain degree of confidence
- $p \rightarrow q$ (0.9) means if p is believed to be true then it is believed that q will hold 90% of the time
- Abduction helps us to see patterns of causality in the world

Information theory and AI (exploration)

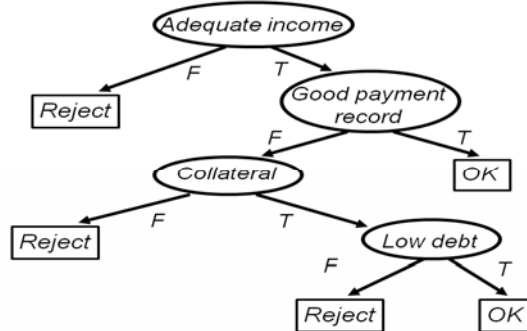
- In exploring an environment or data set, we want to take steps first that help us home in most quickly on a solution
- Some answers to questions give us more information than others
- *Example*: In assessing credit risk, three attributes correlate highly to risk: debt, bad history, inverse of income, inverse of collateral. Answers to these give more information
- Information in a communication can be quantified as the *amount of surprise*

Decision-tree learning

- A *decision tree* performs a series of binary tests resulting in a yes/no answer
- Examples are given in a *training set*
- DTL classifies instances of concepts in trees whose leaves are classifications, e.g., yes/no
- *Input*: a set of attributes; *Output*: a decision
- Discrete outputs occur in *classification learning*, continuous outputs in *regression learning*
- DTL is robust to errors, unlike other concept-learning algorithms

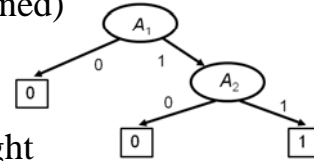
DT example: loan policy

- Suppose granting credit requires adequate income, plus a good payment record or both collateral and low debt
- The following DT implements this loan policy:



Decision-tree learning

- Given *example data* showing results of decision-tree execution for combinations of attribute values, decision trees may be *induced* (learned)
- Example:* the sample data below gives attribute and result values; corresponding decision tree is at right



	Attributes			
Examples	A_1	A_2	A_3	y
x_1	0	0	1	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1

Computational learning theory

- Probably approximately correct* (PAC) learning: any hypothesis consistent with a sufficiently large training set is unlikely to be very mistaken
- For hypothesis h , error (h) = $P(h(x) \neq f(x) \mid x \text{ in training set distribution})$
- Approximately correct h is h s.t. $\text{error}(h) \leq \epsilon$, a small constant

Learning logical assertions from examples

- A form of inductive learning
- The universe of hypotheses is a set of logical assertions
- Hypotheses inconsistent with the example set must be ruled out – examples that are false negatives or positives
- This elimination of hypotheses is analogous to the resolution rule
- For each example, generalize hypothesis if false negative is found, specialize it if false positive is found

Version space learning

- *Least commitment* approach seeks refinement of set of all hypotheses consistent with a growing set of examples
- This version space is the set bounded by the most specific and most general sets consistent with all examples so far
- *Drawbacks*: version space will collapse in case of noise
- Not practical for most real-world problems

Knowledge-based induction

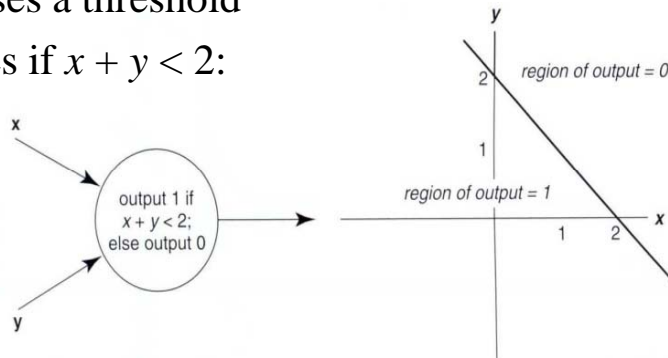
- Cumulative development of knowledge used in inference
- Hypothesis \wedge Descriptions \models Classifications
(*entailment constraint*)
- *Examples*: traveler hearing Portuguese spoken infers that all inhabitants speak Portuguese, but meeting Fernando does not infer that all inhabitants are named Fernando

2. Connectionist learning

- *Connectionist* systems are also known as *parallel distributed processing*
- Non symbolic; based on model of *neuron*
- Intelligence is seen as occurring in systems of connected components in which connections are adjusted in the process of adaptation
- *Applications*: perception related; classification; pattern recognition; memory recall; prediction; optimization; noise filtering

Artificial neurons

- *Artificial neural nets* are composed of simulated *neurons* with input and output
- Neuron fires (outputs 1) if sum of inputs passes a threshold
- Fires if $x + y < 2$:



Neural nets

- *Output activation* of a neuron i :

$$a_i = g(\sum_{j=0 \text{ to } n} W_{j,i} a_j)$$
 where a_j is output activation of neuron j , and $W_{j,i}$ is weight of synapse (j, i)
- *Activation function* may be threshold ($g(x) = 0$ if $x < k$, else 1) or *sigmoid* [PIC]
- *AND, OR, NOT* may be implemented with threshold activation functions [PIC]

Types of neural nets

- *Perceptrons* are single-layer
- Can only compute *linearly separable* functions; XOR for example is not LS
- *Feed-forward* (acyclic)
- *Recurrent* (cyclic) has internal state
- Feed-forward nets are arranged in layers that pass signals to each other
- Multi-layer nets have hidden neuron units

Perceptron example

- Linear functions may be computed by perceptrons; see Assignment 5 and sites referenced there
- [PIC]

Perceptron learning

- Devised by Rosenblatt, 1958
- Single-layer neural net
- Inputs and outputs are in $\{-1, 1\}$
- Neurons begin learning process with default weights
- An optimization search in *weight space*
- Perceptrons have a probabilistic interpretation in weight-update vector

Updating weights

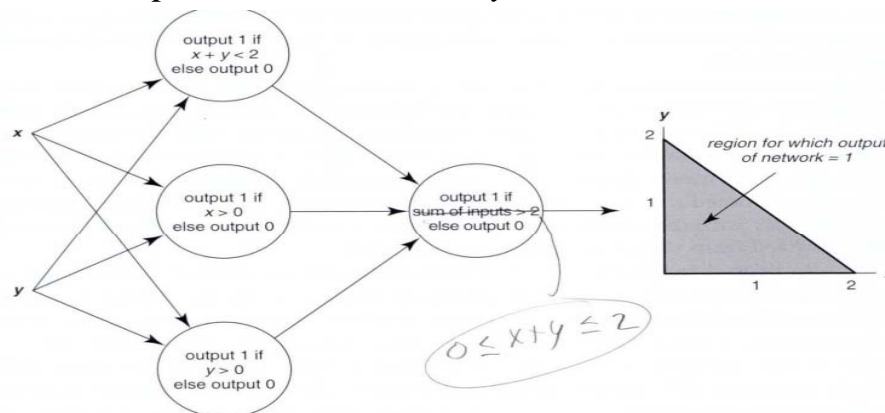
- Sum of squares of errors is a standard measure used to update weights
- *Update:* $w_j \leftarrow w_i + \alpha \times \text{Err} \times g'(\text{in}) \times x_j$
where α is learning rate

Weight adjustment

- To train a NN, sample inputs are provided, and weights are adjusted according to results of training tests
- $\Delta w_i = c(d - \text{sign}(\sum w_i - x_i)) x_i$ where i is a constant and d is a desired output, sign is actual output
- Result of adjustment is a set of weights that minimizes average error over training set
- If some set of weights will generate correct out-put for all of training set, perceptron will find it
- Only *linearly separable* sets of data points are learnable, e.g., not including XOR

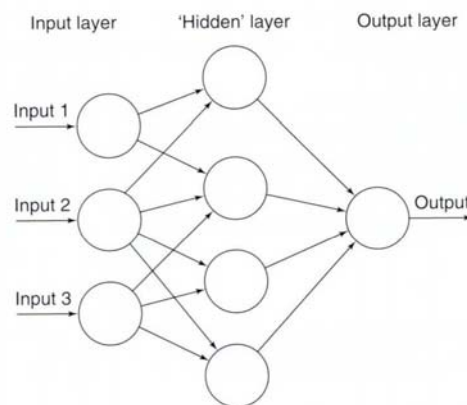
Multi-neuron network

- A multi-neuron network may fire under a more complex condition
- *Example:* Fires if $0 \leq x + y \leq 2$



Hidden layers

- *Hidden layers* of neural nets are neurons that are neither external inputs nor external outputs



Backpropagation learning

Handwriting-recognition learning

- Problem with digits may be stated as NIST standard 60,000 images paired with decimal digits
- *Neural nets* in different versions solve the problem, with error rates as low as 0.7%
- *Kernel machine* (support vector architecture) finds optimal linear separators even for problems that are not linearly separable
- Optimal linear separator finds largest *margin* between separator and positive and negative examples
- *Shape matching* maps points in a pair of images that correspond to each other

Hebbian coincidence learning

- Hebb, 1949: in nature, when a neuron contributes to the firing of another one, the connection between them tends to be strengthened
- Unsupervised Hebbian learning simulates Pavlovian conditioned response
- New stimulus can be associated with old response by presenting a new stimulus, together with old stimulus
- New stimulus elicits same response as conditioned earlier on old stimulus

3. Evolutionary computation

- A probabilistic, population-based way to develop approximate solutions to difficult problems using notions of utility and fitness
- Modeled on natural evolution of species and behaviors of some life forms
- A growing research area encompassing genetic algorithms, evolutionary programming, evolution strategies, ant computing, swarm computing, particle swarm optimization
- EC is ontogenetic learning

The function-optimization problem

- Let $f: \mathbf{N}^k \rightarrow \mathbf{R}$ for some k (the *arity* of f)
- *Problem*: Find some $x \in \mathbf{N}^k$ s.t. $f(x)$ is maximal graph
- *Example*: If x is the set of proportions of ingredients in a fuel mixture, then $f(x)$ is fuel cost-efficiency under this mixture
- *Optimizing* $f(x)$ means finding the most cost-efficient mixture
- For an *algorithm* to optimize a function, we must have $f: X \rightarrow Y$ with X, Y finite

Fitness and function optimization

- *Example:* Suppose f is viewed as a *fitness* function and x is the set of attributes of individuals of a population
- Then finding x s.t. $f(x)$ is maximal is finding the fittest possible individual of the species, i.e., those with the best attributes to assure survival
- Evolution by natural selection tends to optimize fitness, over many generations
- Optimization problems in general are hard

Evolutionary computation

- State space is explored using and comparing a *population* of states rather than one at a time
- Evolutionary algorithm repeatedly modifies and selects from a population of solutions
- Selection is driven by fitness of each member of the population
- Randomization is used to explore the state space

The evolutionary algorithm

```

t ← 0
Initialize (P0)
V ← Evaluate (P0)
While not Terminate (V, t) do
  t ← t + 1
  Pt ← Select (Pt-1, V)
  Pt ← Alter (Pt)
  V ← Evaluate (Pt)
Return Pt

```

time

vector of fitness values

population at time t

Mutate or cross over

- *Evaluate* applies fitness function to individuals
- *Select* chooses some members of P to survive
- *Alter* changes P randomly, e.g., by mutation or crossover
- *Terminate* ends algorithm after a given goal or a time deadline is reached

Genetic algorithms

- At each step, the population is selected and regenerated using genetic operators, e.g., mutation and crossover
- *Parameters*: % of population to retain at each generation, which mate and produce offspring, which probability measures, if any, to use

Example: Checkers (Samuel, 1950s)

- Let fitness function $f: \mathbf{N}^k \rightarrow \mathbf{R}$ be an evaluator of checkers board positions from a black or red angle
- Let $x \in \mathbf{N}^k$ be a k -tuple of *weights* for each of k different criteria for evaluating a checkers position
- *Example:* let x_1 be relative importance of number of kings, x_2 be relative importance of number of opponent checkers threatened, etc.
- Then writing a good checkers-playing program reduces to finding a good set of relative weights x_1, \dots, x_k for these criteria
- *Result:* machine play at very high level

Evolving perceptron

EC has addressed *static* environments

- Environment is *static* in the checkers example because the game rules don't change during evolution
- *Static environment* =
single (unchanging) fitness function
- In a *dynamic* environment, fitness or reward will *change* as the environment changes
- An interactive agent in a changing environment must adapt its response as environment changes
- Single fitness function in EC \Rightarrow
Environment cannot be dynamic

Evolutionary computation in dynamic environments

- The function-optimization problem defined above is *static*, assumes function f will be the same later as now
- Optimization in the real world, e.g., fitness in natural environments, such as climate, is often *dynamic*
- A separate, but related, category of environment is *stochastic* (imperfectly predictable)

Dynamic persistent environments

- *Definition:* A *dynamic persistent environment* (DPE) is a CAPS, E , that may interact with some other CAPS, M , with E changing state due to M 's actions in a way perceptible to M . We call E a *dynamic persistent environment with respect to M* .
- *Discussion:* If some inputs received from E by M have *reward value*, then E induces a *fitness function* w.r.t. M at each interaction step
- This function maps M 's output to the value of M 's immediate reward
- Note that this function varies with the state of E , i.e., evolves over time

Policy in a dynamic environment

- The *policy* of agent M , with respect to environment E , is a computable function from possible perceptions, or models, of E , to M 's set of outputs.
- The *fitness of a policy in environment E* , is the expected long-term reward in E of an agent with that policy.

Dynamic-environment example

- Suppose we play *cat-and-mouse* on a grid
- *Agent* is mouse; *Environment* is cat and grid
- Goals of mouse policy: escape by fleeing or hiding
- Assume mouse policy is to be evolved; fitness function is survival rate of a policy
- If cat speeds up over time, then mouse policy must switch from *flee* to *hide*
- Traditional evolutionary algorithm *fails* here because it assumes static environment

4. Natural-language processing

- *Speech act*: utterance for purpose of communication
- “Speech” is generic, as written, signed, etc.
- Used in coordination of multi-agent systems: querying, informing, requesting actions, acknowledging, committing to act, declaring (as in, “Strike 3”)
- *Uniqueness of human language*: only it can specify an infinity of distinct messages

Steps of communication

- *Intention*: meaning sought to communicate
- *Generation*: plan of creation of utterance with meaning intended
- *Synthesis*: production of utterance from symbols
- *Perception*: hearing or recognition
- *Analysis*: syntactic, semantic, pragmatic breakdown by hearer, possibly yielding multiple interpretations
- *Disambiguation*: considering probabilities
- *Incorporation* into hearer's language

Languages

- *Language*: a set of strings, concatenating terminal symbols (e.g., words)
- *Grammar*: finite set of rules to specify a language
- *Semantics*: meaning
- *Pragmatics*: meaning in context

Formal language theory

- *Language*: a set of strings over a finite alphabet Σ of symbols, e.g., letters
- Empty language (with no strings): \emptyset
- Null string (with no symbols): λ
- Operations on languages: concatenation, union, intersection, complement
- Inductive definition of universal language Σ^* of all strings over Σ : $\Sigma^* = \{xa \mid x \in \Sigma^*, a \in \Sigma\}$
- *Example*: $(aa \mid bb)^*$ is language over $\{a, b\}$ of strings in which all symbols appear in pairs

Regular languages and regular expressions

- Any RL may be specified by a *regular expression* using any combination of these operations:
 - Concatenation
 - Selection ($|$, $+$, \cup)
 - Iteration ($*$) (binds to immediate preceding symbol)
- *Examples*:
 - $(01)^*$: all strings that repeat the string 01, zero or more times
 - $01^* \mid 10^*$: all strings that either consist of a 0 followed by zero or more 1's, or consist of a 1 followed by zero or more 0's

Context-free grammars

- *Definition:* CFG G consists of *terminal symbols*; *nonterminals* (NT, names); *production rules*
- *Nonterminal symbols:* symbols that denote phrase structures, with definition rules, e.g.,
 $S \rightarrow NP \ VP$
- *Production rules* are of the form
 $X \rightarrow YZ\dots$ where $X \in NT$ and
 $Y, Z\dots \in (\Sigma \cup NT)$
- *Example:* $S \rightarrow \lambda \mid 0S1$
- By this grammar, a sentence can be a null string or a 0, followed by a sentence, followed by a 1

The Chomsky hierarchy

- *Regular languages* are recognized by finite-state machines (deterministic finite automata) and are generated by regular expressions (e.g., $\text{digit}^*.\text{digit}^*$ for fractional decimal numerals)
- *Context-free languages* are recognized by stack machines and are generated by context-free grammars (rewrite rules; see earlier slide)
- *Context-sensitive languages* include natural language with verb agreement
- *Recursive and recursively enumerable sets* complete the set of languages recognizable by algorithms

English-language grammar

- *Lexicon*: a set of words, with their parts of speech (N, V, adj, adv, article...)
- *Grammar*: a set of production rules:
 $S \rightarrow NP VP \mid S \text{ conj } S$
 $NP \rightarrow \text{noun} \mid \text{article noun}$
 $VP \rightarrow \text{verb} \mid VP NP VP \text{ adj}$
- *Parsing* applies rules to a sentence, starting with sentence symbol (S), replacing left nonterminal with a definition string:
 $S \Rightarrow NP VP \Rightarrow \text{noun VP} \Rightarrow \text{noun verb} \Rightarrow$
 $\text{John verb} \Rightarrow \text{John goes}$

Semantic interpretation

- May be the process of extracting a FOL sentence from a natural-language sentence
- *Examples*:
 - $\text{Sem}(\text{"the wumpus"}) = \text{Wumpus1}$
 - $\text{Sem}(\text{"the wumpus is dead"}) = \text{Dead}(\text{Wumpus1})$
- *Lambda calculus*: "John loves Mary" = $(\lambda x \text{ Loves}(x, \text{Mary}))(\text{John})$, where λx defines a predicate whose argument is John
- *Quantification*: "I sleep" = $(\exists e) e \in \text{Sleep}(\text{Speaker}) \wedge \text{During}(\text{Now}, e)$

Querying conceptual graphs

- *Sentence*: Fido likes June
- *Semantic representation*:
[pic]
- *Query*: Who likes June?
- [pic]

Semantic ambiguity

- “Every dog has a day” =
 - $(\forall d) d \in \text{Dogs} \Rightarrow (\exists a) a \in \text{Days} \wedge (\exists e) e \in \text{Has}(d, a, \text{Now})$
 - or
 - $(\exists a) a \in \text{Days} \wedge (\forall d) d \in \text{Dogs} \Rightarrow \text{Has}(d, a, \text{Now})$
- Each dog has its own day vs. there is a day shared by all dogs
- Ambiguity may be lexical, syntactic, or semantic
- *Metonymy*: replacing one object for another, as in “Chrysler announced...”

Knowledge models

- *World*: proposition occurs in the world
- *Mental*: speaker intends to communicate the information to hearer
- *Language*: correspondence of word choice to fact
- *Acoustic*: generation of sounds corresponding to words

Discourse understanding

- *Discourse*: a string, usually more than one sentence
- *Issues*:
 - *Reference resolution*: interpretation of pronoun or other reference
 - *Coherence*: depends on order; issues: enabling, causation, explanation, setting, exemplification, generalization, expectation

Probabilistic language models

- As opposed to *logical* models (grammars)
- Enable processing of text in which speakers use different versions of language
- Enable disambiguation by selection of likely interpretation
- Can generate bigram (2-word), unigram, and trigram probabilistic models from large text bases

Information extraction

- *Definition:* creation of database entries from text
- *Example:* “17 in X VGA Monitor for only \$249.99” generates $(\exists m) (m \in \text{Monitors} \wedge \text{Size}(m, \text{Inches}(17)) \wedge \text{Price}(m, 249.99))$
- *Regular expressions* can help analysis of text for attribute information
- Works well in restricted domains when subjects of discourse are known

Application: machine translation

- Translation from source to target language
- *Types*
 - Rough
 - Restricted source (limited subject matter and format)
 - Pre-redited source
 - Literary (beyond current software capabilities)
- *Challenge*: different languages have expressions with overlapping meanings, lacking 1-to-1 correspondence convenient for translating

Concepts

abduction	decision tree	machine translation
ambiguity	learning	multi-layer net
analogical reasoning	explanation based	neural network
artificial neuron	learning	nonterminal symbol
associative memory	feed-forward net	PAC learning
attractor	grammar	parse tree
backpropagation	Hebbian learning	perceptron
Chomsky hierarchy	hidden layer	pragmatics
computational learning	inductive bias	recurrent net
theory	inductive learning	regular language
concept	language	semantics
connectionist learning	learnability	supervised learning
context-free language	learning	syntax
decision tree learning		

References

George Luger. *Artificial Intelligence*. Addison Wesley, 2005.

Tom Mitchell. *Machine learning*. McGraw-Hill, 1997.

Stuart Russell and Peter Norvig. *AI: A Modern Approach, 2nd ed.* Prentice Hall, 2003.

Paul Thagard. *Mind: Introduction to Cognitive Science, 2nd ed.* MIT press, 2005.