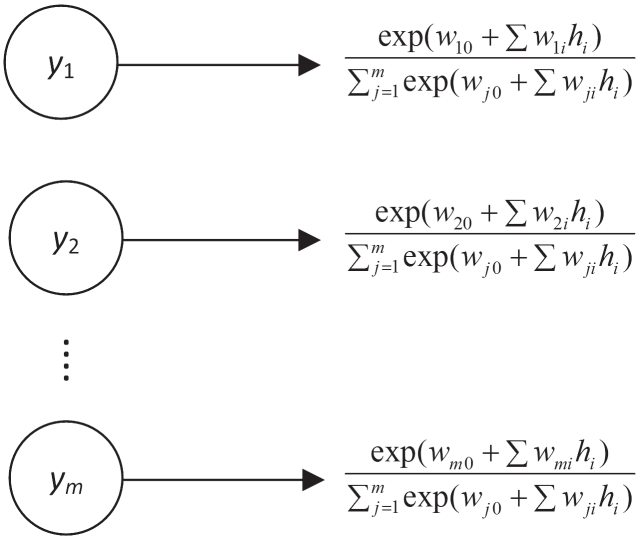

Figure 15.12 The output layer for classifying a variable with m values using the softmax function.

15.3.1.2 Multinomial Classification

In multinomial classification we want to classify or predict a variable that has m possible values. An important example, in which neural networks are often applied, is image recognition. For example, we may have a handwritten symbol of one of the digits 0-9, and our goal is to classify the symbol as the digit intended by the writer. To do multinomial classification, we use an extension of the sigmoid function called the **softmax function**. When using this function, we develop one output node for each of the m possible outcomes, and we assign the following activation function to the kth output node:

$$f_k(\mathbf{h}) = \frac{\exp(w_{k0} + \sum w_{ki}h_i)}{\sum_{j=1}^{m} \exp(w_{j0} + \sum w_{ji}h_i)}.$$

In this way, the kth output node provides $P(Y = k|\mathbf{h})$. Figure 15.12 shows the output layer for classifying a variable with m values using the softmax function.

Example 15.5 Suppose our output function is the softmax function, we have 3 outputs, y_1, y_2, and y_3, and

$$w_{10} = 2, w_{11} = -4, w_{12} = 3$$
$$w_{20} = 1, w_{21} = 9, w_{22} = -3$$
$$w_{30} = 10, w_{31} = 7, w_{32} = -4.$$

Suppose further that for a particular input

$$h_1 = 2, h_2 = 4.$$

Then

$$w_{10} + w_{11}h_1 + w_{12}h_2 = 2 - 4 \times 2 + 3 \times 4 = 6$$
$$w_{20} + w_{21}h_1 + w_{22}h_2 = 1 + 9 \times 2 - 3 \times 4 = 7$$
$$w_{30} + w_{31}h_1 + w_{32}h_2 = 10 + 7 \times 2 - 4 \times 4 = 8.$$

$$f_1(\mathbf{h}) = \frac{\exp(w_{10} + \sum w_{1i}h_i)}{\sum_{j=1}^{3} \exp(w_{j0} + \sum w_{ji}h_i)} = \frac{\exp(6)}{\exp(6) + \exp(7) + \exp(8)} = 0.090$$

$$f_2(\mathbf{h}) = \frac{\exp(w_{20} + \sum w_{2i}h_i)}{\sum_{j=1}^{3} \exp(w_{j0} + \sum w_{ji}h_i)} = \frac{\exp(7)}{\exp(6) + \exp(7) + \exp(8)} = 0.245$$

$$f_3(\mathbf{h}) = \frac{\exp(w_{30} + \sum w_{3i}h_i)}{\sum_{j=1}^{3} \exp(w_{j0} + \sum w_{ji}h_i)} = \frac{\exp(8)}{\exp(6) + \exp(7) + \exp(8)} = 0.665.$$

So,

$$\begin{aligned} P(Y &= 1|\mathbf{h}) = 0.090 \\ P(Y &= 2|\mathbf{h}) = 0.245 \\ P(Y &= 3|\mathbf{h}) = 0.665. \end{aligned}$$

15.3.1.3 Normal Output Distributions

Instead of modeling that the output can only have one of m values (discrete), we can assume it is normally distributed. We have then that

$$\rho(y|\mathbf{x}) = \text{NormalDen}(y; \mu, \sigma^2),$$

where $\mathbf{x}$ is the input vector. For example, we may want to predict a normal distribution of systolic blood pressure based on a patient's features. In this case, we can have a single output node, where its value is the mean of the normal distribution. So, the activation function is simply the **linear activation function**:

$$\mu = f(\mathbf{h}) = w_0 + \sum w_i h_i.$$

Example 15.6 Suppose our output function is the mean of a normal distribution, and

$$w_0 = 2, w_1 = -4, w_2 = -3, w_3 = 5.$$

Suppose further that for a particular input

$$h_1 = 3, h_2 = 7, h_3 = 8.$$

Then the mean of the output normal distribution is as follows:

$$\begin{aligned} \mu = f(\mathbf{h}) &= w_0 + w_1 \times h_1 + w_2 \times h_2 + w_3 \times h_3 \\ &= 2 - 4 \times 3 - 3 \times 7 + 5 \times 8 \\ &= 9. \end{aligned}$$

15.3.2 Hidden Nodes

The most popular activation function for hidden nodes (especially in applications involving image recognition) is the **rectified linear activation function,** which we have already used. That function is as follows:

$$f(\mathbf{h}) = \max(0, w_0 + \sum w_i h_i).$$

Notice that this activation function is similar to the linear activation function just discussed, except that it returns 0 when the linear combination is negative.

Example 15.7 Suppose our activation function is the rectified linear function, and

$$w_0 = -5, w_1 = -4, w_2 = -3, w_3 = 5.$$

Suppose further that for a particular input

$$h_1 = 8, h_2 = 7, h_3 = 9.$$

Then

$$\begin{aligned} f(\mathbf{h}) &= \max(0, w_0 + w_1 \times h_1 + w_2 \times h_2 + w_3 \times h_3) \\ &= \max(0, -5 - 4 \times 8 - 3 \times 7 + 5 \times 9) \\ &= \max(0, -13) \\ &= 0. \end{aligned}$$

Another activation function used for hidden nodes is the **maxout activation function**. For this function, we have r weight vectors, where r is a parameter, and we take the maximum of the weighted sums. That is, we have

$$\begin{aligned} z_1 &= w_{10} + \sum w_{1i} h_i \\ z_2 &= w_{20} + \sum w_{2i} h_i \\ &\vdots \\ z_r &= w_{r0} + \sum w_{ri} h_i, \end{aligned}$$

and

$$f(\mathbf{h}) = \max(z_1, z_2, ..., z_r).$$

Example 15.8 Suppose our activation function is the maxout function, $r = 3$, and

$$w_{10} = 2, w_{11} = -4, w_{12} = 3$$

$$w_{20} = 1, w_{21} = 9, w_{22} = -3$$

$$w_{30} = 10, w_{31} = 7, w_{32} = -4.$$

Suppose further that for a particular input

$$h_1 = 2, h_2 = 4.$$

Then

$$z_1 = w_{10} + w_{11} h_1 + w_{12} h_2 = 2 - 4 \times 2 + 3 \times 4 = 6$$

$$z_2 = w_{20} + w_{12} h_1 + w_{22} h_2 = 1 + 9 \times 2 - 3 \times 4 = 7$$

$$z_3 = w_{30} + w_{31} h_1 + w_{32} h_2 = 10 + 7 \times 2 - 4 \times 4 = 8.$$

$$f(\mathbf{h}) = \max(z_1, z_2, z_3) = \max(6, 7, 8) = 8.$$

The sigmoid activation function (Equation 15.4) can also be used as the activation function for hidden notes. A related function, which is also used, is the **hyperbolic tangent activation function**, which is as follows:

$$f(h) = \tanh(w_0 + \sum w_i h_i).$$

Figure 15.13 Examples of the handwritten digits in the MNIST dataset.

15.4 Application to Image Recognition

The MNIST dataset (http://yann.lecun.com/exdb/mnist/) is an academic dataset used to evaluate the performance of classification algorithms. The dataset consists of 60,000 training images and 10,000 test images. Each image is one of the digits 0-9, which is handwritten. It is a standardized 28 by 28 pixel greyscale image. So, there are 784 pixels in all. Figure 15.13 shows examples of the digits in the dataset.

Candel et al. [2015] developed a neural network to solve the problem of classifying images in the test dataset by learning a system from the training dataset. The network has 784 inputs (one for each pixel), 10 outputs (one for each digit) and three hidden layers, with each layer containing 200 hidden nodes. Each hidden node uses the rectified linear activation function, and the output nodes use the softmax function. Figure 15.14 depicts the network.

The neural network had a classification error rate of 0.0083, which ties the best error rate previously achieved by Microsoft.

15.5 Discussion and Further Reading

The title of this chapter is "Neural Networks and Deep Learning." Yet we never mentioned "deep learning" in the text. As mentioned in Section 1.1.2, in the 1940s foundational efforts at AI involved modeling the neurons in the brain, which resulted in the field of **neural networks** [Hebb, 1949]. Once the logical approach to AI became dominant in the 1950s, neural networks fell from popularity. However, new algorithms for training neural networks and dramatically increased computer processing speed resulted in a re-emergence of the use of neural nets in the field called **deep learning** [Goodfellow et al., 2016]. **Deep learning neural network** architectures differ from older neural networks in that they often have more hidden layers. Furthermore, deep learning networks can be trained using both unsupervised and supervised learning. We presented only the supervised learning approach. **Convolutional neural networks** are ones whose architectures make the explicit assumption that the inputs are images, and therefore encode specific properties of images. Recurrent neural networks are a class of neural nets that feed their state at the previous time step into the current time step. They are applied, for example, to automatic text generation (discussed below).

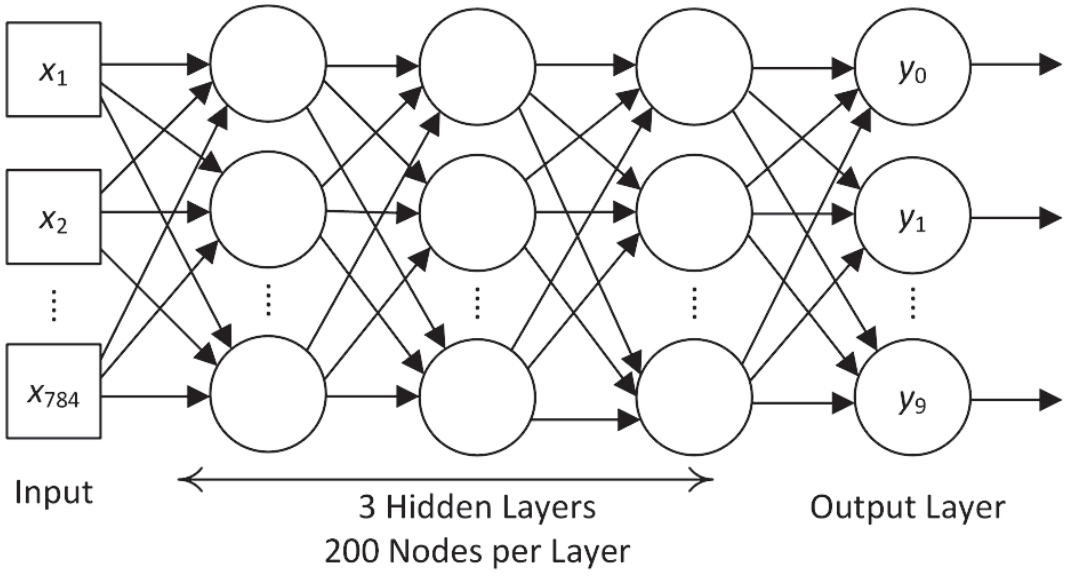

Figure 15.14 The neural network used to classify digits in the MNIST dataset.

This has only been a brief introduction to the basics of neural networks. For a more thorough coverage, including a discussion of the gradient descent algorithms used to learn neural network parameters, you are referred to [Goodfellow et al., 2016] and [Theodoridis, 2015]. You can download software that implements neural networks. Two such products are H20 (https://www.h2o.ai/) and tenserflow (https://www.tensorflow.org/).

Deep learning has been used to solve a variety of problems, which were difficult with other approaches. We close by listing some specific applications.

1. Object detection and classification in a photograph [Krizhevsky et al., 2012]. This problem involves classifying objects in a photograph as one of a set of previously known objects.

2. Adding color to black and white photographs [Zang et al., 2016]. This problem concerns adding color to black and white photographs.

3. Automatic image caption generation [Karpathy and Fei-Fei, 2015]. This task concerns generating a caption that describes the contents of an image.

4. Adding sound to silent videos [Owens et al., 2016]. This problem concerns synthesizing sounds that best match what is happening in a scene in a video.

5. Automatic language translation [Sutskever et al., 2014]. This task involves translating a word, phrase, or sentence from one language to another language.

6. Automatic handwriting generation [Graves, 2014]. This problem involves generating new handwriting for a given word or phrase based on a set of handwriting examples.

7. Automatic text generation [Sutskever et al., 2011]. This problem involves using a large amount of text to learn a model that can generate a new word, phrase, or sentence based on a partial word or text.

8. Automatic game playing [Mnih et al., 2015]. This problem concerns learning how to play a computer game based on the pixels on the screen.

When modeling a problem using a neural network, the model is a black box in the sense that the structure and parameters in the layers do not provide us with a model of reality, which we can grasp. Bayesian networks, on the other hand, provide a relationship among variables, which can often be interpreted as causal. Furthermore, Bayesian networks enable us to model and understand complex human decisions. So, although both architectures can be used to model many of the same problems, neural networks have more often been successfully applied to problems that involve human intelligence which cannot be described at the cognitive level. These problems include computer vision, image processing, and text analysis. Bayesian networks, on the other hand, have more often been applied successfully to problems that involve determining the relationships among related random variables, and exploiting these relationships to do inference and make decisions. A classic example is a medical decision support system (See Section 7.7).

EXERCISES

Section 15.1

Exercise 15.1 Does the resultant line in Example 15.1 linearly separate the data? If not, work through more iterations of the repeat loop until it does.

Exercise 15.2 Suppose we set $\lambda = 0, 2$, and we have the following data:

x_1	x_2	y
2	3	-1
4	5	1

Work through iterations of the repeat loop in Algorithm 15.1 until the resultant line linearly separates the data.

Exercise 15.3 It was left as an exercise to implement Algorithm 15.1, apply it to the data in Table 15.1, and compare the results to those obtained with logistic regression. Do this.

Section 15.2

Exercise 15.4 Suppose the three lines determining the regions in Figure 15.7 are as follows:

$$\begin{aligned} h_{11} &: 4 - 2x_1 - x_2 = 0 \\ h_{12} &: 1 - x_1 - x_2 = 0 \\ h_{13} &: 3 + 2x_1 - x_2 = 0. \end{aligned}$$

Plot the three lines and show the regions corresponding to classes C_1 and C_2. Develop parameters for the neural network in Figure 15.8 such that the network properly classifies the points in the classes C_1 and C_2.

Section 15.3

Exercise 15.5 Suppose our output function is the sigmoid function for binary output and

$$w_0 = 1, w_1 = -4, w_2 = -5, w_3 = 4.$$

Suppose further that for a particular input

$$h_1 = 4, h_2 = 5, h_3 = 6.$$

Compute $f(\mathbf{h})$. What is $P(Y = 1|\mathbf{h})$?

Exercise 15.6 Suppose our output function is the softmax function, and we have 4 outputs, y_1, y_2, y_3, y_4. Suppose further that

$$\begin{aligned} w_{10} = 1, w_{11} = -3, w_{12} = 2 \\ w_{20} = 2, w_{21} = 7, w_{22} = -2 \\ w_{30} = 6, w_{31} = 5, w_{32} = -4. \\ w_{40} = 5, w_{41} = -3, w_{42} = 6. \end{aligned}$$

Suppose further that for a particular input

$$h_1 = 3, h_2 = 4, h_3 = 5.$$

Compute $f_1(\mathbf{h}), f_2(\mathbf{h}), f_3(\mathbf{h}), f_4(\mathbf{h})$. What is $P(Y = i|\mathbf{h})$ for $i = 1, 2, 3, 4$?

Exercise 15.7 Suppose our activation function is the rectified linear function, and

$$w_0 = 5, w_1 = -4, w_2 = 2, w_3 = 4, w_4 = 8.$$

Suppose further that for a particular input

$$h_1 = 8, h_2 = 7, h_3 = 6, h_4 = 5.$$

Compute $f(\mathbf{h})$.

Exercise 15.8 Suppose our activation function is the maxout function, $r = 2$, and

$$\begin{aligned} w_{10} = 1, w_{11} = -2, w_{12} = 6, w_{13} = 5 \\ w_{20} = 2, w_{21} = 8, w_{22} = -2, w_{23} = 4. \end{aligned}$$

Suppose further that for a particular input

$$h_1 = 3, h_2 = 2, h_3 = 5.$$

Compute $f(\mathbf{h})$.

Section 15.4

Exercise 15.9 The Metabric dataset is introduced in [Curtis, et al. 2012]. It provides data on breast cancer patient features such as tumor size and outcomes such as death. This dataset can be obtained at https://www.synapse.org/#!Synapse:syn1688369/wiki/27311. Gain access to the dataset. Then download one of the neural network packages discussed in Section 15.5. Divide the dataset into a training dataset containing 2/3 of the data, and a test dataset containing 1/3 of the data. Apply various parameter settings (e.g., number of hidden layers and number of hidden nodes per layer) to the training set. For each setting do a 5-fold cross validation analysis, where the goal is to classify/predict whether the patient dies. Determine the Area Under an ROC Curve (AUROC) for each of the settings, and apply the settings with the best AUROC to the test data. Determine the AUROC for the test data.

A **naive Bayesian network** is a network in which there is one root, and all other nodes are children of the root. There are no edges among the children. Naive Bayesian network are used for discrete classification by making the target the root, and the predictors the children. XLSTAT includes a naive Bayesian network module. Free naive Bayesian network software is available at various sites including http://www.kdnuggets.com/software/bayesian.html. Download a naive Bayesian network software package, and do the same study as outlined above using this package. Vary whatever parameters are available in the software, and do the 5-fold cross validation analysis for each parameter setting. Compare the AUROCs obtained by the neural network method and the naive Bayesian network method when applied to the test data.

Exercise 15.10 As discussed in Section 15.4, the MNIST dataset is an academic dataset used to evaluate the performance of classification algorithms. The dataset consists of 60,000 training images and 10,000 test images. Each image is one of the digits 0-9, which is handwritten. It is a standardized 28 by 28 pixel greyscale image. So, there are 784 pixels in all. Download this dataset. Then download one of the neural network packages discussed in Section 15.5. Apply various parameter settings (e.g., number of hidden layers and number of hidden nodes per layer) to the training set. For each setting do a 5-fold cross validation analysis, where the goal is to classify/predict the correct digit. Determine the Area Under an ROC Curve (AUROC) for each of the settings, and apply the settings with the best AUROC to the test data. Determine the AUROC for the test data.

Than apply a naive Bayesian network to the dataset. Again use various parameter settings and 5-fold cross validation on the training dataset, and apply the best parameter values to the test dataset. Compare the AUROCs obtained by the neural network method and the naive Bayesian network method when applied to the test dataset.

CHAPTER

7

AI-COMPLETENESS: THE PROBLEM DOMAIN OF SUPERINTELLIGENT MACHINES

AI-Completeness

The Problem Domain of Superintelligent Machines[*]

1.1 INTRODUCTION

Since its inception in the 1950s, the field of artificial intelligence (AI) has produced some unparalleled accomplishments while failing to formalize the problem space that concerns it. This chapter addresses this shortcoming by extending previous work (Yampolskiy 2012a) and contributing to the theory of AI-Completeness, a formalism designed to do for the field of AI what the notion of NP-Completeness (where NP stands for nondeterministic polynomial time) did for computer science in general. It is my belief that such formalization will allow for even faster progress in solving remaining problems in humankind's quest to build an intelligent machine.

According to Wikipedia, the term *AI-Complete* was proposed by Fanya Montalvo in the 1980s ("AI-Complete" 2011). A somewhat general definition of the term included in the 1991 "Jargon File" (Raymond 1991) states:

> AI-complete: [MIT, Stanford, by analogy with "NP-complete"] adj. Used to describe problems or subproblems in AI, to indicate that the solution presupposes a solution to the "strong AI

[*] Reprinted from Roman V. Yampolskiy, Artificial intelligence, evolutionary computation and metaheuristics. *Studies in Computational Intelligence* 427:3–17, 2013, with kind permission of Springer Science and Business Media.

problem" (that is, the synthesis of a human-level intelligence). A problem that is AI-complete is, in other words, just too hard.

As such, the term *AI-Complete* (or sometimes AI-Hard) has been a part of the field for many years and has been frequently brought up to express the difficulty of a specific problem investigated by researchers (see Mueller 1987; Mallery 1988; Gentry, Ramzan, and Stubblebine 2005; Phillips and Beveridge 2009; Bergmair 2004; Ide and Véronis 1998; Navigli and Velardi 2005; Nejad 2010; Chen et al. 2009; McIntire, Havig, and McIntire 2009; McIntire, McIntire, and Havig 2009; Mert and Dalkilic 2009; Hendler 2008; Leahu, Sengers, and Mateas 2008; Yampolskiy 2011). This informal use further encouraged similar concepts to be developed in other areas of science: Biometric-Completeness (Phillips and Beveridge 2009) or Automatic Speech Recognition (ASR)-Complete (Morgan et al. 2003). Although recently numerous attempts to formalize what it means to say that a problem is AI-Complete have been published (Ahn et al. 2003; Shahaf and Amir 2007; Demasi, Szwarcfiter, and Cruz 2010), even before such formalization attempts, systems that relied on humans to solve problems perceived to be AI-Complete were utilized:

- **AntiCaptcha** systems use humans to break the CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) security protocol (Ahn et al. 2003; Yampolskiy 2007a, 2007b; Yampolskiy and Govindaraju 2007) either by directly hiring cheap workers in developing countries (Bajaj 2010) or by rewarding correctly solved CAPTCHAs with presentation of pornographic images (Vaas 2007).
- The **Chinese room** philosophical argument by John Searle shows that including a human as a part of a computational system may actually reduce its perceived capabilities, such as understanding and consciousness (Searle 1980).
- **Content development** online projects such as encyclopedias (Wikipedia, Conservapedia); libraries (Project Gutenberg, video collections [YouTube]; and open-source software [SourceForge]) all rely on contributions from people for content production and quality assurance.
- **Cyphermint**, a check-cashing system, relies on human workers to compare a snapshot of a person trying to perform a financial

transaction to a picture of a person who initially enrolled with the system. Resulting accuracy outperforms any biometric system and is almost completely spoof proof (see http://cyphermint.com for more information).

- **Data-tagging** systems entice a user into providing metadata for images, sound, or video files. A popular approach involves developing an online game that, as a by-product of participation, produces a large amount of accurately labeled data (Ahn 2006).
- **Distributed Proofreaders** employs a number of human volunteers to eliminate errors in books created by relying on Optical Character Recognition process (see http://pgdp.net/c/ for more information).
- **Interactive evolutionary computation** algorithms use humans in place of a fitness function to make judgments regarding difficult-to-formalize concepts such as aesthetic beauty or taste (Takagi 2001).
- **Mechanical Turk** is an attempt by Amazon.com to create Artificial AI. Humans are paid varying amounts for solving problems that are believed to be beyond current abilities of AI programs (see https://www.mturk.com/mturk/welcome for more information). The general idea behind the Turk has broad appeal, and the researchers are currently attempting to bring it to the masses via the generalized task markets (GTMs) (Shahaf and Horvitz 2010; Horvitz and Paek 2007; Horvitz 2007; Kapoor et al. 2008).
- **Spam prevention** is easy to accomplish by having humans vote on e-mails they receive as spam or not. If a certain threshold is reached, a particular piece of e-mail could be said with a high degree of accuracy to be spam (Dimmock and Maddison 2004).

Recent work has attempted to formalize the intuitive notion of AI-Completeness. In particular, three such endowers are worth reviewing next (Yampolskiy 2012a). In 2003, Ahn et al. attempted to formalize the notion of an AI-Problem and the concept of AI-Hardness in the context of computer security. An AI-Problem was defined as a triple:

$\mathcal{P} = (S, D, f)$, where S is a set of problem instances, D is a probability distribution over the problem set S, and $f : S \rightarrow \{0; 1\}^*$

> answers the instances. Let $\delta \in (0; 1]$. We require that for an $\alpha > 0$ fraction of the humans H, $Pr_{x \leftarrow D}[H(x) = f(x)] > \delta$. … An AI problem $\mathcal{P}$ is said to be (δ, τ)-*solved* if there exists a program A, running in time at most τ on any input from S, such that $Pr_{x \leftarrow D, r}[A_r(x) = f(x)] \geq \delta$. ($A$ is said to be a (δ, τ) solution to $\mathcal{P}$.) $\mathcal{P}$ is said to be a (δ, τ)-*hard AI problem* if no current program is a (δ, τ) solution to $\mathcal{P}$. (Ahn et al. 2003, 298).

It is interesting to observe that the proposed definition is in terms of democratic consensus by the AI community. If researchers say the problem is hard, it must be so. Also, time to solve the problem is not taken into account. The definition simply requires that some humans be able to solve the problem (Ahn et al. 2003).

In 2007, Shahaf and Amir presented their work on the theory of AI-Completeness. Their work puts forward the concept of the human-assisted Turing machine and formalizes the notion of different human oracles (HOs; see the section on HOs for technical details). The main contribution of the paper comes in the form of a method for classifying problems in terms of human-versus-machine effort required to find a solution. For some common problems, such as natural language understanding (NLU), the work proposes a method of reductions that allow conversion from NLU to the problem of speech understanding via text-to-speech software.

In 2010, Demasi et al. (Demasi, Szwarcfiter, and Cruz 2010) presented their work on problem classification for artificial general intelligence (AGI). The proposed framework groups the problem space into three sectors:

- **Non-AGI-Bound**: problems that are of no interest to AGI researchers
- **AGI-Bound**: problems that require human-level intelligence to be solved
- **AGI-Hard**: problems that are at least as hard as any AGI-Bound problem.

The work also formalizes the notion of HOs and provides a number of definitions regarding their properties and valid operations.

String Human (String input) {

return output; }

FIGURE 1.1 Human oracle: $\text{Human}_{\text{Best}}$, a union of minds.

1.2 THE THEORY OF AI-COMPLETENESS

From people with mental disabilities to geniuses, human minds are cognitively diverse, and it is well known that different people exhibit different mental abilities. I define a notion of an HO function capable of computing any function computable by the union of all human minds. In other words, any cognitive ability of any human being is repeatable by my HO. To make my HO easier to understand, I provide Figure 1.1, which illustrates the *Human* function.

Such a function would be easy to integrate with any modern programming language and would require that the input to the function be provided as a single string of length N, and the function would return a string of length M. No encoding is specified for the content of strings N or M, so they could be either binary representations of data or English language phrases—both are computationally equivalent. As necessary, the Human function could call regular Turing Machine (TM) functions to help in processing data. For example, a simple computer program that would display the input string as a picture to make human comprehension easier could be executed. Humans could be assumed to be cooperating, perhaps because of a reward. Alternatively, one can construct a Human function that instead of the union of all minds computes the average decision of all human minds on a problem encoded by the input string as the number of such minds goes to infinity. To avoid any confusion, I propose naming the first HO $\text{Human}_{\text{Best}}$ and the second HO $\text{Human}_{\text{Average}}$. Problems in the AI domain tend to have a large degree of ambiguity in terms of acceptable correct answers. Depending on the problem at hand, the simplistic notion of an average answer could be replaced with an aggregate

answer as defined in the wisdom-of-crowds approach (Surowiecki 2004). Both functions could be formalized as human-assisted Turing machines (Shahaf and Amir 2007).

The human function is an easy-to-understand and -use generalization of the HO. One can perceive it as a way to connect and exchange information with a real human sitting at a computer terminal. Although easy to intuitively understand, such description is not sufficiently formal. Shahaf et al. have formalized the notion of HO as an Human-Assisted Turing Machine (HTM) (Shahaf and Amir 2007). In their model, a human is an oracle machine that can decide a set of languages L_i in constant time: $H \subseteq \{L_i \mid L_i \subseteq \sum^{\star}\}$. If time complexity is taken into account, answering a question might take a nonconstant time, $H \subseteq \{<L_i, f_i> \mid L_i \subseteq \sum^{\star}, f_i : \mathbb{N} \rightarrow \mathbb{N}\}$, where f_i is the time-complexity function for language L_i, meaning the human can decide if $x \in L_i$ in f_i ($|x|$) time. To realistically address capabilities of individual humans, a probabilistic oracle was also presented that provided correct answers with probability p: $H \subseteq \{<L_i, p_i> \mid L_i \subseteq \sum^{\star}, 0 \leq p_i \leq 1\}$. Finally, the notion of reward is introduced into the model to capture humans' improved performance on "paid" tasks: $H \subseteq \{<L_i, u_i> \mid L_i \subseteq \sum^{\star}, u_i : \mathbb{N} \rightarrow \mathbb{N}\}$ where u_i is the utility function (Shahaf and Amir 2007).

1.2.1 Definitions

Definition 1: A problem C is **AI-Complete** if it has two properties:

1. It is in the set of AI problems (HO solvable).
2. Any AI problem can be converted into C by some polynomial time algorithm.

Definition 2: **AI-Hard:** A problem H is AI-Hard if and only if there is an AI-Complete problem C that is polynomial time Turing reducible to H. ■

Definition 3: **AI-Easy:** The complexity class AI-Easy is the set of problems that are solvable in polynomial time by a deterministic Turing machine with an oracle for some AI problem. In other words, a problem X is AI-Easy if and only if there exists some AI problem Y such that X is polynomial time Turing reducible to Y. This means that given an oracle for Y, there exists an algorithm that solves X in polynomial time. ■

Figure 1.2 illustrates the relationship between different AI complexity classes. The right side of the figure shows the situation if it is ever proven that AI problems = AI-Complete problems. The left side shows the converse.

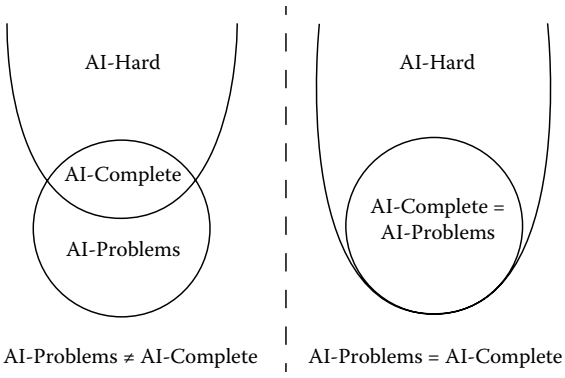

FIGURE 1.2 Relationship between AI complexity classes.

1.2.2 Turing Test as the First AI-Complete Problem

In this section, I show that a Turing test (TT; Turing 1950) problem is AI-Complete. First, I need to establish that a TT is indeed an AI problem (HO solvable). This trivially follows from the definition of the test itself. The test measures if a human-like performance is demonstrated by the test taker, and HOs are defined to produce human-level performance. While both *human* and *intelligence test* are intuitively understood terms, I have already shown that HOs could be expressed in strictly formal terms. The TT itself also could be formalized as an interactive proof (Shieber 2006, 2007; Bradford and Wollowski 1995).

The second requirement for a problem to be proven to be AI-Complete is that any other AI problem should be convertible into an instance of the problem under consideration in polynomial time via Turing reduction. Therefore, I need to show how any problem solvable by the Human function could be encoded as an instance of a TT. For any HO-solvable problem *h*, we have a string *input* that encodes the problem and a string *output* that encodes the solution. By taking the *input* as a question to be used in the TT and *output* as an answer to be expected while administering a TT, we can see how any HO-solvable problem could be reduced in polynomial time to an instance of a TT. Clearly, the described process is in polynomial time, and by similar algorithm, any AI problem could be reduced to TT. It is even theoretically possible to construct a complete TT that utilizes all other problems solvable by HO by generating one question from each such problem.

1.2.3 Reducing Other Problems to a TT

Having shown a first problem (TT) to be AI-Complete, the next step is to see if any other well-known AI problems are also AI-Complete. This is an effort similar to the work of Richard Karp, who showed some 21 problems were NP-Complete in his 1972 work and by doing so started a new field of computational complexity (Karp 1972). According to the *Encyclopedia of Artificial Intelligence* (Shapiro 1992), the following problems are all believed to be AI-Complete and so will constitute primary targets for our effort of proving formal AI-Completeness on them (Shapiro 1992, 54–57):

- **Natural Language Understanding:** "Encyclopedic knowledge is required to understand natural language. Therefore, a complete Natural Language system will also be a complete Intelligent system."
- **Problem Solving:** "Since any area investigated by AI researchers may be seen as consisting of problems to be solved, all of AI may be seen as involving Problem Solving and Search."
- **Knowledge Representation and Reasoning:** "The intended use is to use explicitly stored knowledge to produce additional explicit knowledge. This is what reasoning is. Together Knowledge representation and Reasoning can be seen to be both necessary and sufficient for producing general intelligence—it is another AI-complete area."
- **Vision or Image Understanding:** "If we take 'interpreting' broadly enough, it is clear that general intelligence may be needed to do this interpretation, and that correct interpretation implies general intelligence, so this is another AI-complete area."

Now that the TT has been proven to be AI-Complete, we have an additional way of showing other problems to be AI-Complete. We can either show that a problem is both in the set of AI problems and all other AI problems can be converted into it by some polynomial time algorithm or can reduce any instance of TT problem (or any other problem already proven to be AI-Complete) to an instance of a problem we are trying to show to be AI-Complete. This second approach seems to be particularly powerful. The general heuristic of my approach is to see if all information encoding the question that could be asked during administration of a TT could be encoded as an instance of a problem in question and likewise if any potential solution to that problem would constitute an answer to the

relevant TT question. Under this heuristic, it is easy to see that, for example, chess is not AI-Complete as only limited information can be encoded as a starting position on a standard-size chessboard. Not surprisingly, chess has been one of the greatest successes of AI; currently, chess-playing programs dominate all human players, including world champions.

Question answering (QA) (Hirschman and Gaizauskas 2001; Salloum 2009) is a subproblem in natural language processing. Answering questions at a level of a human is something HOs are particularly good at based on their definition. Consequently, QA is an AI-Problem that is one of the two requirements for showing it to be AI-Complete. Having access to an oracle capable of solving QA allows us to solve TT via a simple reduction. For any statement *S* presented during administration of TT, transform said statement into a question for the QA oracle. The answers produced by the oracle can be used as replies in the TT, allowing the program to pass the TT. It is important to note that access to the QA oracle is sufficient to pass the TT only if questions are not restricted to stand-alone queries, but could contain information from previous questions. Otherwise, the problem is readily solvable even by today's machines, such as IBM's Watson, which showed a remarkable performance against human *Jeopardy* champions (Pepitone 2011).

Speech understanding (SU) (Anusuya and Katti 2009) is another subproblem in natural language processing. Understanding speech at a level of a human is something HOs are particularly good at based on their definition. Consequently, SU is an AI-Problem that is one of the two requirements for showing it to be AI-Complete. Having access to an oracle capable of solving SU allows us to solve QA via a simple reduction. We can reduce QA to SU by utilizing any text-to-speech software (Taylor and Black 1999; Chan 2003), which is both fast and accurate. This reduction effectively transforms written questions into the spoken ones, making it possible to solve every instance of QA by referring to the SU oracle.

1.2.4 Other Probably AI-Complete Problems

I hope that my work will challenge the AI community to prove other important problems as either belonging or not belonging to that class. Although the following problems have not been explicitly shown to be AI-Complete, they are strong candidates for such classification and are problems of great practical importance, making their classification a worthy endeavor. If a problem has been explicitly conjectured to be AI-Complete in a published paper, I include a source of such speculation: dreaming (Salloum 2009);

commonsense planning (Shahaf and Amir 2007); foreign policy (Mallery 1988); problem solving (Shapiro 1992); judging a TT (Shahaf and Amir 2007); commonsense knowledge (Andrich, Novosel, and Hrnkas 2009); SU (Shahaf and Amir 2007); knowledge representation and reasoning (Shapiro 1992); word sense disambiguation (Chen et al. 2009; Navigli and Velardi 2005); Machine Translation ("AI-Complete" 2011); ubiquitous computing (Leahu, Sengers, and Mateas 2008); change management for biomedical ontologies (Nejad 2010); NLU (Shapiro 1992); software brittleness ("AI-Complete" 2011); and vision or image understanding (Shapiro 1992).

1.3 FIRST AI-HARD PROBLEM: PROGRAMMING

I define the problem of programming as taking a natural language description of a program and producing a source code, which then is compiled on some readily available hardware/software to produce a computer program that satisfies all implicit and explicit requirements provided in the natural language description of the programming problem assignment. Simple examples of programming are typical assignments given to students in computer science classes, for example, "Write a program to play tic-tac-toe." Successful students write source code that, if correctly compiled, allows the grader to engage the computer in an instance of that game. Many requirements of such an assignment remain implicit, such as that response time of the computer should be less than a minute. Such implicit requirements are usually easily inferred by students who have access to culture-instilled common sense. As of this writing, no program is capable of solving programming outside strictly restricted domains.

Having access to an oracle capable of solving programming allows us to solve TT via a simple reduction. For any statement *S* presented during TT, transform said statement into a programming assignment of the form: "Write a program that would respond to *S* with a statement indistinguishable from a statement provided by an average human" (a full transcript of the TT may also be provided for disambiguation purposes). Applied to the set of all possible TT statements, this procedure clearly allows us to pass TT; however, programming itself is not in AI-Problems as there are many instances of programming that are not solvable by HOs. For example, "Write a program to pass a Turing test" is not known to be an AI-Problem under the proposed definition. Consequently, programming is an AI-Hard problem.

1.4 BEYOND AI-COMPLETENESS

The HO function presented in this chapter assumes that the human behind it has some assistance from the computer in order to process certain human unfriendly data formats. For example, a binary string representing a video is completely impossible for a human to interpret, but it could easily be played by a computer program in the intended format, making it possible for a human to solve a video understanding-related AI-Complete problem. It is obvious that a human provided with access to a computer (perhaps with Internet connection) is a more powerful intelligence compared to an unenhanced, in such a way, human. Consequently, it is important to limit help from a computer to a human worker "inside" a HO function to assistance in the domain of input/output conversion, but not beyond, as the resulting function would be both AI-Complete and "Computer Complete".

Figure 1.3 utilizes a Venn diagram to illustrate subdivisions of problem space produced by different types of intelligent computational devices. Region 1 represents what is known as a Universal Intelligence (Legg and Hutter 2007) or a Super Intelligence (Legg 2008; Yampolskiy 2011a, 2011b, 2012b)—a computational agent that outperforms all other intelligent

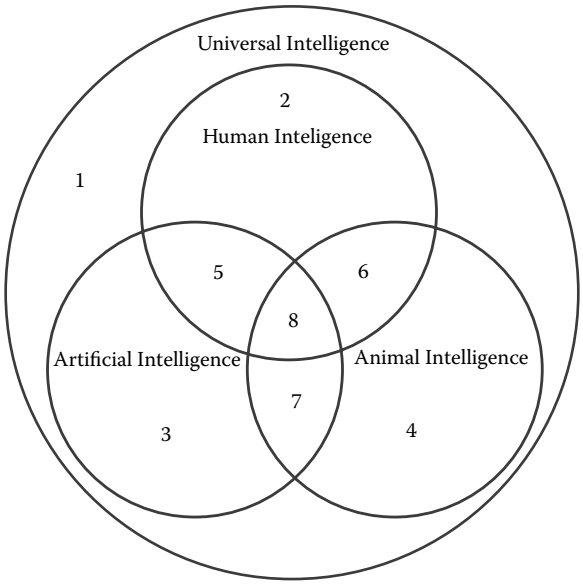

FIGURE 1.3 Venn diagram for four different types of intelligence.

agents over all possible environments. Region 2 is the standard unenhanced Human-level intelligence of the type capable of passing a TT, but at the same time incapable of computation involving large numbers or significant amount of memorization. Region 3 is what is currently possible to accomplish via state-of-the-art AI programs. Finally, Region 4 represents an abstract view of animal intelligence.

AI intelligence researchers strive to produce Universal Intelligence, and it is certainly likely to happen, given recent trends in both hardware and software developments and the theoretical underpinning of the Church/Turing Thesis (Turing 1936). It is also likely that if we are able to enhance human minds with additional memory and port those to a higher-speed hardware we will essentially obtain a Universal Intelligence (Sandberg and Boström 2008).

While the Universal Intelligence incorporates abilities of all the lower intelligences, it is interesting to observe that Human, AI and Animal intelligences have many interesting regions of intersection (Yampolskiy and Fox 2012). For example, animal minds are as good as human minds at visual understanding of natural scenes. Regions 5, 6, and 7 illustrate common problem spaces between two different types of intelligent agents. Region 8 represents common problem solving abilities of humans, computers and animals. Understanding such regions of commonality may help us to better separate the involved computational classes, which are represented by abilities of a specific computational agent minus the commonalities with a computational agent with which we are trying to draw a distinction. For example, CAPTCHA (Ahn et al. 2003) type tests rely on the inability of computers to perform certain pattern recognition tasks with the same level of accuracy as humans in order to separate AI agents from Human agents. Alternatively, a test could be devised to tell humans not armed with calculators from AIs by looking at the upper level of ability. Such a test should be easy to defeat once an effort is made to compile and formalize the limitations and biases of the human mind.

It is also interesting to consider the problem solving abilities of hybrid agents. I have already noted that a human being equipped with a computer is a lot more capable compared to an unaided person. Some research in Brain Computer Interfaces (Vidal 1973) provides a potential path for future developments in the area. Just as interestingly, combining pattern recognition abilities of animals with symbol processing abilities of AI could produce a computational agent with a large domain of human-like abilities (see work on RoboRats by Talwar et al. (2002) and on monkey controlled robots by Nicolelis

et al. 2000). It is very likely that in the near future different types of intelligent agents will be combined to even greater extent. While such work is under way, I believe that it may be useful to introduce some additional terminology into the field of AI problem classification. For the complete space of problems I propose that the computational agents which are capable of solving a specific subset of such problems get to represent the set in question. Therefore, I propose additional terms: "Computer-Complete" and "Animals-Complete" to represent computational classes solvable by such agents. It is understood that just as humans differ in their abilities, so do animals and computers. Aggregation and averaging utilized in my Human function could be similarly applied to the definition of respective oracles. As research progresses, common names may be needed for different combinations of regions from Figure 1.3 illustrating such concepts as Human-AI hybrid or Animal-Robot hybrid.

Certain aspects of human cognition do not map well onto the space of problems which have seen a lot of success in the AI research field. Internal states of the human mind, such as consciousness (stream of), self-awareness, understanding, emotions (love, hate), feelings (pain, pleasure), etc., are not currently addressable by our methods. Our current state-of-the-art technologies are not sufficient to unambiguously measure or detect such internal states, and consequently even their existence is not universally accepted. Many scientists propose ignoring such internal states or claim they are nothing but a byproduct of flawed self-analysis. Such scientists want us to restrict science only to measurable behavioral actions; however, since all persons have access to internal states of at least one thinking machine, interest in trying to investigate internal states of the human mind is unlikely to vanish.

While I am able to present a formal theory of AI-Completeness based on the concept of HOs, the theory is not strong enough to address problems involving internal states of the mind. In fact, one of the fundamental arguments against our ability to implement understanding in a system that is based on symbol manipulation, Searle's Chinese Room thought experiment, itself relies on a generalized concept of a human as a part of a computational cycle. It seems that the current Turing/Von Neumann architecture is incapable of dealing with the set of problems which are related to internal states of human mind. Perhaps a new type of computational architecture capable of mimicking such internal states will be developed in the future. It is likely that it will be inspired by a better understanding of human biology and cognitive science. Research on creating Artificial Consciousness (AC) is attracting a lot of attention, at least in terms of number of AC papers published.

As a part of my ongoing effort to classify AI related problems, I propose a new category specifically devoted to problems of reproducing internal states of the human mind in artificial ways. I call this group of problems Consciousness-Complete or C-Complete for short. An oracle capable of solving C-Complete problems would be fundamentally different from the Oracle Machines proposed by Turing. C-Oracles would take input in the same way as their standard counterparts but would not produce any symbolic output. The result of their work would be a novel internal state of the oracle, which may become accessible to us if the new type of hardware discussed above is developed.

Just as SAT was shown to be the first NP-Complete problem and TT to be the first AI-Complete problem, I suspect that Consciousness will be shown to be the first C-Complete problem, with all other internal-state related problems being reducible to it. Which of the other internal state problems are also C-Complete is beyond the scope of this preliminary work. Even with no consciousness-capable hardware available at the moment of this writing, the theory of C-Completeness is still a useful tool, as it allows for formal classification of classical problems in the field of Artificial Intelligence into two very important categories: potentially *solvable* (with current technology) and unsolvable (with current technology). Since the only information available about HOs is their output and not internal states, they are fundamentally different from C-Oracles, creating two disjoint sets of problems.

The history of AI research is full of unwarranted claims of anticipated breakthroughs and, conversely, overestimations regarding the difficulty of some problems. Viewed through the prism of my AI-Complete/C-Complete theories, the history of AI starts to make sense. Solutions for problems that I classify as AI-Complete have been subject to continuous steady improvement, while those falling in the realm of C-Completeness have effectively seen zero progress (computer pain, Bishop 2009 and Dennett 1978; artificial consciousness, Searle 1980 and Dreyfus 1972; etc.). To proceed, science needs to better understand what the difference between a feeling and a thought is. Feeling pain and knowing about pain are certainly not the same internal states. I am hopeful that future research in this area will bring some long-awaited answers.

1.5 CONCLUSIONS

Progress in the field of artificial intelligence requires access to well-defined problems of measurable complexity. The theory of AI-Completeness aims to provide a base for such formalization. Showing certain problems to be

AI-Complete/-Hard is useful for developing novel ways of telling computers from humans. Also, any problem shown to be AI-Complete would be a great alternative way of testing an artificial intelligent agent to see if it attained human level intelligence (Shahaf and Amir 2007).

REFERENCES

Ahn, Luis von. June 2006. Games with a purpose. *IEEE Computer Magazine* 96–98.

Ahn, Luis von, Manuel Blum, Nick Hopper, and John Langford. 2003. CAPTCHA: Using Hard AI Problems for Security. Paper read at Eurocrypt. Advances in Cryptology — EUROCRYPT 2003. International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003. Published in *Lecture Notes in Computer Science* 2656 (2003): 294–311.

AI-Complete. 2011. Accessed January 7. http://en.wikipedia.org/wiki/AI-complete.

Andrich, Christian, Leo Novosel, and Bojan Hrnkas. 2009. Common Sense Knowledge. Exercise Paper—Information Search and Retrieval. http://www.iicm.tu-graz.ac.at/cguetl/courses/isr/uearchive/uews2009/Ue06-CommonSenseKnowledge.pdf

Anusuya, M. A. and S. K. Katti. 2009. Speech recognition by machine: a review. *International Journal of Computer Science and Information Security (IJCSIS)* no. 6(3):181–205.

Bajaj, Vikas. April 25, 2010. Spammers pay others to answer security tests. *New York Times*.

Bergmair, Richard. December 2004. Natural Language Steganography and an "AI-Complete" Security Primitive. In 21st Chaos Communication Congress, Berlin.

Bishop, Mark. 2009. Why computers can't feel pain. *Minds and Machines* 19(4):507–516.

Bradford, Philip G. and Michael Wollowski. 1995. A formalization of the Turing Test. *SIGART Bulletin* 6(4):3–10.

Chan, Tsz-Yan. 2003. Using a text-to-speech synthesizer to generate a reverse Turing test. Paper presented at the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03), Washington, DC, November 3–5.

Chen, Junpeng, Juan Liu, Wei Yu, and Peng Wu. November 30, 2009. Combining Lexical Stability and Improved Lexical Chain for Unsupervised Word Sense Disambiguation. Paper presented at the Second International Symposium on Knowledge Acquisition and Modeling (KAM '09), Wuhan, China.

Demasi, Pedro, Jayme L. Szwarcfiter, and Adriano J. O. Cruz. March 5–8, 2010. A Theoretical Framework to Formalize AGI-Hard Problems. Paper presented at the Third Conference on Artificial General Intelligence, Lugano, Switzerland.

Dennett, Daniel C. July 1978. Why you can't make a computer that feels pain. *Synthese* 38(3):415–456.

Dimmock, Nathan and Ian Maddison. December 2004. Peer-to-peer collaborative spam detection. *Crossroads* 11(2): 17–25.

Dreyfus, Hubert L. 1972. *What Computers Can't Do: A Critique of Artificial Reason*: New York: Harper & Row.

Gentry, Craig, Zulfikar Ramzan, and Stuart Stubblebine. June 5–8, 2005. Secure Distributed Human Computation. Paper presented at the 6th ACM Conference on Electronic Commerce, Vancouver, BC, Canada.

Hendler, James. September 2008. We've come a long way, maybe … . *IEEE Intelligent Systems* 23(5):2–3.

Hirschman, L., and R Gaizauskas. 2001. Natural language question answering. The view from here. *Natural Language Engineering* 7(4):275–300.

Horvitz, E. 2007. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine—Special Issue on Mixed-Initiative Assistants* 28(2): 11–18.

Horvitz, E. and T. Paek. 2007. Complementary computing: policies for transferring callers from dialog systems to human receptionists. *User Modeling and User Adapted Interaction* 17(1):159–182.

Ide, N. and J. Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics* 24(1):1–40.

Kapoor, A., D. Tan, P. Shenoy, and E. Horvitz. September 17–19, 2008. Complementary Computing for Visual Tasks: Meshing Computer Vision with Human Visual Processing. Paper presented at the IEEE International Conference on Automatic Face and Gesture Recognition, Amsterdam.

Karp, Richard M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, edited by R. E. Miller and J. W. Thatcher, 85–103. New York: Plenum.

Leahu, Lucian, Phoebe Sengers, and Michael Mateas. September 21–24, 2008. Interactionist AI and the Promise of ubicomp, or, How to Put Your Box in the World Without Putting the World in Your Box. Paper presented at the *Tenth International Conference on Ubiquitous Computing*. Seoul, South Korea.

Legg, Shane. June 2008. Machine Super Intelligence. PhD thesis, University of Lugano, Switzerland. http://www.vetta.org/documents/Machine_Super_Intelligence.pdf

Legg, Shane and Marcus Hutter. December 2007. Universal intelligence: a definition of machine intelligence. *Minds and Machines* 17(4):391–444.

Mallery, John C. 1988. Thinking about Foreign Policy: Finding an Appropriate Role for Artificial Intelligence Computers. Ph.D. dissertation, MIT Political Science Department, Cambridge, MA.

McIntire, John P., Paul R. Havig, and Lindsey K. McIntire. July 21–23, 2009. Ideas on Authenticating Humanness in Collaborative Systems Using AI-Hard Problems in Perception and Cognition. Paper presented at the IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH.

McIntire, John P., Lindsey K. McIntire, and Paul R. Havig. May 18–22, 2009. A Variety of Automated Turing tests for Network Security: Using AI-Hard Problems in Perception and Cognition to Ensure Secure Collaborations. Paper presented at the International Symposium on Collaborative Technologies and Systems (CTS '09), Baltimore.

Mert, Ezgi, and Cokhan Dalkilic. September 14–16, 2009. Word Sense Disambiguation for Turkish. Paper presented at the 24th International Symposium on Computer and Information Sciences (ISCIS 2009), Guzelyurt, Turkey.

Morgan, Nelson, D. Baron, S. Bhagat, H. Carvey, R. Dhillon, J. Edwards, D. Gelbart, A. Janin, A. Krupski, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. April 6–10, 2003. Meetings about Meetings: Research at ICSI on Speech in Multiparty Conversations. Paper presented at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03), Hong Kong.

Mueller, Erik T. March 1987. Daydreaming and Computation. PhD dissertation, University of California, Los Angeles.

Navigli, Roberto, and Paola Velardi. July 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(7):1075–1086.

Nejad, Arash Shaban. April 2010. A Framework for Analyzing Changes in Health Care Lexicons and Nomenclatures. PhD dissertation, Concordia University, Montreal, QC, Canada.

Nicolelis, Miguel A. L., Johan Wessberg, Christopher R. Stambaugh, Jerald D. Kralik, Pamela D. Beck, Mark Laubach, John K. Chapin, and Jung Kim. 2000. Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. *Nature* 408(6810):361.

Pepitone, Julianne. 2011. IBM's Jeopardy supercomputer beats humans in practice bout. *CNNMoney*. http://money.cnn.com/2011/01/13/technology/ibm_jeopardy_watson. Accessed January 13.

Phillips, P. Jonathon, and J. Ross Beveridge. September 28–30, 2009. An Introduction to Biometric-Completeness: The Equivalence of Matching and Quality. Paper presented at the IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS '09), Washington, DC.

Raymond, Eric S. March 22, 1991. Jargon File Version 2.8.1. http://catb.org/esr/jargon/oldversions/jarg282.txt

Salloum, W. November 30, 2009. A Question Answering System Based on Conceptual Graph Formalism. Paper presented at the 2nd International Symposium on Knowledge Acquisition and Modeling (KAM 2009), Wuhan, China.

Sandberg, Anders, and Nick Boström. 2008. Whole Brain Emulation: A Roadmap. Technical Report 2008-3. Future of Humanity Institute, Oxford University. http://www.fhi.ox.ac.uk/Reports/2008-3.pdf

Searle, John. 1980. Minds, brains and programs. *Behavioral and Brain Sciences* 3(3):417–457.

Shahaf, Dafna, and Eyal Amir. March 26–28, 2007. Towards a Theory of AI Completeness. Paper presented at the 8th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2007), Stanford University, Stanford, CA.

Shahaf, D., and E. Horvitz. July 2010. Generalized Task Markets for Human and Machine Computation. Paper presented at the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA.

Shapiro, Stuart C. 1992. Artificial Intelligence. In *Encyclopedia of Artificial Intelligence*, edited by Stuart C. Shapiro, 54–57. New York: Wiley.

Shieber, Stuart M. July 16–20, 2006. Does the Turing Test Demonstrate Intelligence or Not? Paper presented at the Twenty-First National Conference on Artificial Intelligence (AAAI-06), Boston.

Shieber, Stuart M. December 2007. The Turing test as interactive proof. *Nous* 41(4):686–713.

Surowiecki, James. 2004. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations.* New York: Little, Brown.

Takagi, H. 2001. Interactive evolutionary computation: fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE 89* 9:1275–1296.

Talwar, Sanjiv K., Shaohua Xu, Emerson S. Hawley, Shennan A. Weiss, Karen A. Moxon, and John K. Chapin. May 2, 2002. Behavioural neuroscience: rat navigation guided by remote control. *Nature* 417:37–38.

Taylor, P., and A. Black. 1999. Speech Synthesis by Phonological Structure Matching. Paper presented at Eurospeech99, Budapest, Hungary.

Turing, A. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Turing, Alan M. 1936. On computable numbers, with an application to the Entscheidungs problem. *Proceedings of the London Mathematical Society* 42:230–265.

Vaas, Lisa. December 1, 2007. Striptease used to recruit help in cracking sites. *PC Magazine*. http://www.pcmag.com/article2/0,2817,2210671,00.asp

Vidal, J. J. 1973. Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering* 2:157–180.

Yampolskiy, R. V. 2011. AI-Complete CAPTCHAs as zero knowledge proofs of access to an artificially intelligent system. *ISRN Artificial Intelligence* 2012:271878.

Yampolskiy, Roman V. April 13, 2007a. Embedded CAPTCHA for Online Poker. Paper presented at the 20th Annual CSE Graduate Conference (Grad-Conf2007), Buffalo, NY.

Yampolskiy, Roman V. September 28, 2007b. Graphical CAPTCHA Embedded in Cards. Paper presented at the Western New York Image Processing Workshop (WNYIPW)—IEEE Signal Processing Society, Rochester, NY.

Yampolskiy, Roman V. October 3–4, 2011a. Artificial Intelligence Safety Engineering: Why Machine Ethics Is a Wrong Approach. Paper presented at Philosophy and Theory of Artificial Intelligence (PT-AI2011), Thessaloniki, Greece.

Yampolskiy, Roman V. October 3–4, 2011b. What to Do with the Singularity Paradox? Paper presented at Philosophy and Theory of Artificial Intelligence (PT-AI2011), Thessaloniki, Greece.

Yampolskiy, Roman V. April 21–22, 2012a. AI-Complete, AI-Hard, or AI-Easy—Classification of Problems in AI. Paper presented at the 23rd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH.

Yampolskiy, Roman V. 2012b. Leakproofing singularity—artificial intelligence confinement problem. *Journal of Consciousness Studies (JCS)* 19(1–2):194–214.

Yampolskiy, Roman V., and Joshua Fox. 2012. Artificial general intelligence and the human mental model. In *In the Singularity Hypothesis: A Scientific and Philosophical Assessment*, edited by Amnon Eden, Jim Moor, Johnny Soraker, and Eric Steinhart, 129–146. New York: Springer.

Yampolskiy, Roman V., and Venu Govindaraju. 2007. Embedded non-interactive continuous bot detection. *ACM Computers in Entertainment* 5(4):1–11.