Algorithms for Learning Preferences for Sets of Objects

The user gives examples of preferred sets; the algorithms do the rest.

NASA's Jet Propulsion Laboratory, Pasadena, California

A method is being developed that provides for an artificial-intelligence system to learn a user's preferences for sets of objects and to thereafter automatically select subsets of objects according to those preferences. The method was originally intended to enable automated selection, from among large sets of images acquired by instruments aboard spacecraft, of image subsets considered to be scientifically valuable enough to justify use of limited communication resources for transmission to Earth. The method is also applicable to other sets of objects: examples of sets of objects considered in the development of the method include food menus, radio-station music playlists, and assortments of colored blocks for creating mosaics.

The method does not require the user to perform the often-difficult task of quantitatively specifying preferences; instead, the user provides examples of preferred sets of objects. This method goes beyond related prior artificial-intelligence methods for learning which individual items are preferred by the user: this method supports a concept of set-based preferences, which include not only preferences for individual items but also preferences regarding types and degrees of diversity of items in a set. Consideration of diversity in this method involves recognition that members of a set may interact with each other in the sense that when considered together, they may be regarded as being complementary, redundant, or incompatible to various degrees. The effects of such interactions are loosely summarized in the term "portfolio effect."

The learning method relies on a preference representation language, denoted DD-PREF, to express set-based preferences. In DD-PREF, a preference is represented by a tuple that includes quality ("depth") functions to estimate how desired a specific value is, weights for each feature preference, the desired diversity of feature values, and the relative importance of diversity versus depth. The system applies statistical concepts to estimate quantitative measures of the user's preferences from training examples (preferred subsets) specified by the user. Once preferences have been learned, the system uses those preferences to select preferred subsets from new sets.

The method was found to be viable when tested in computational experiments on menus, music playlists, and rover images. Contemplated future development efforts include further tests on more diverse sets and development of a submethod for (a) estimating the parameter that represents the relative importance of diversity versus depth, and (b) incorporating background knowledge about the nature of quality functions, which are special functions that specify depth preferences for features.

This work was done by Kiri L. Wagstaff of Caltech and Marie desJardins and Eric Eaton of the University of Maryland, Baltimore County, for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

The software used in this innovation is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-43828.

Model for Simulating a Spiral Software-Development Process

A prior model for simulating a waterfall process has been extended.

John F. Kennedy Space Center, Florida

A discrete-event simulation model, and a computer program that implements the model, have been developed as means of analyzing a spiral software-development process. This model can be tailored to specific development environments for use by software project managers in making quantitative cases for deciding among different software-development processes, courses of action, and cost estimates.

A spiral process can be contrasted with a waterfall process, which is a traditional process that consists of a sequence of activities that include analysis of requirements, design, coding, testing, and support. A spiral process is an iterative process that can be regarded as a repeating modified waterfall process. Each iteration includes assessment of risk, analysis of requirements, design, coding, testing, delivery, and evaluation. A key difference between a spiral and a waterfall process is that a spiral process can accommodate changes in requirements at each iteration, whereas in a waterfall process, requirements are considered to be fixed from the beginning and, therefore, a waterfall process is not flexible enough for some projects, especially those in which requirements are not known at the beginning or may change during development. For a given project, a spiral process may cost more and take more time than does a waterfall process, but may better satisfy a customer's expectations and needs.

Models for simulating various waterfall processes have been developed previously, but until now, there have been no models for simulating spiral processes. The present spiral-process-simulating model and the software that implements it were developed by extending a discrete-event simulation process model of the IEEE 12207 Software Development Process, which was built using commercially available software known as the