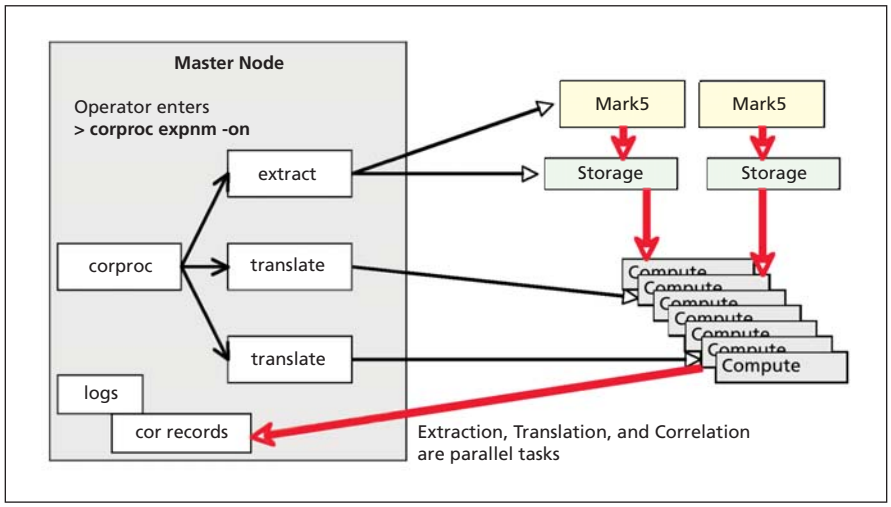


Figure 2. **Extraction, Translation, and Correlation** are run as parallel tasks.

Beowulf cluster to process separate scans in parallel until all scans have been processed. Due to the single-stream sequential playback of the Mark5 data, some ramp-up time is required before all nodes can have access to required scan data. Core functions of each processing step are accomplished using optimized C programs. The coordination and execution of these programs across the cluster is accomplished using Pearl scripts, PostgreSQL commands, and a handful of miscellaneous system utilities.

Mark5 data modules are loaded on Mark5 Data systems playback units, one per station. Data processing is started when the operator scans the Mark5 systems and runs a script that reads various configuration files and then creates an experiment-dependent status database used to delegate parallel tasks between nodes and storage areas (see Figure 2). This script forks into three processes: extract, translate, and correlate. Each of these processes iterates on available scan data and updates the status database as the work for each scan is completed.

The extract process coordinates and monitors the transfer of data from each of the Mark5s to the Beowulf RAID storage systems. The translate process monitors and executes the data conversion processes on available scan files, and writes the translated files to the slave nodes. The correlate process monitors the execution of SoftC correlation processes on the slave nodes for scans that have completed translation.

A comparison of the JVC and the legacy Block II correlator outputs reveals they are well within a formal error, and that the data are comparable with respect to their use in flight navigation. The processing speed of the JVC is improved over the Block II correlator by a factor of 4, largely due to the elimination of the reel-to-reel tape drives used in the Block II correlator.

This work was done by Stephen P. Rogstad, Andre P. Jongeling, Susan G. Finley, Leslie A. White, Gabor E. Lanyi, John E. Clark, and Charles E. Goodhart of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov. NPO-46279

Hybrid NN/SVM Computational System for Optimizing Designs

The NN and the SVM help each other "learn" in an iterative process.

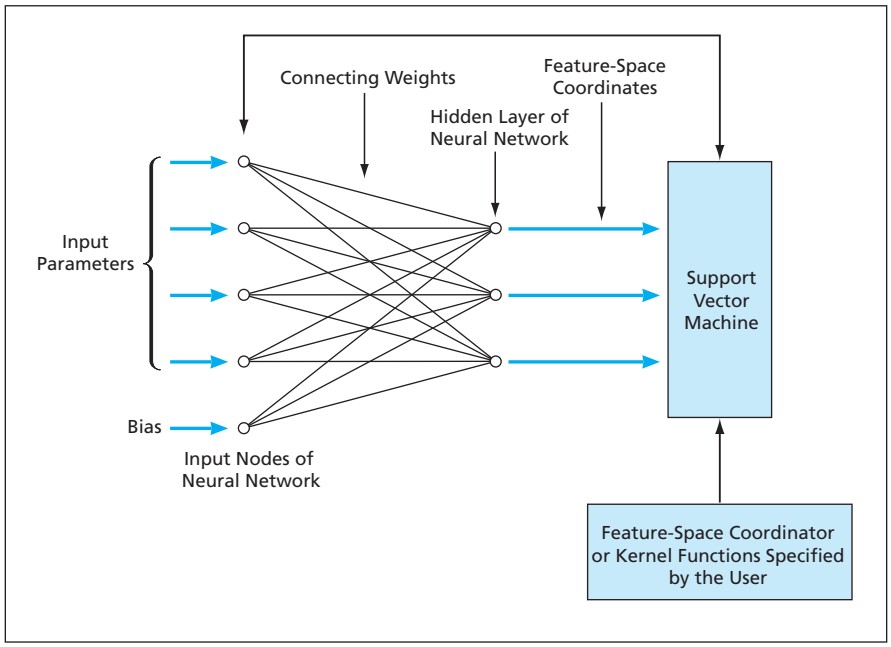
Ames Research Center, Moffett Field, California

A computational method and system based on a hybrid of an artificial neural network (NN) and a support vector machine (SVM) (see figure) has been conceived as a means of maximizing or minimizing an objective function, optionally subject to one or more constraints. Such maximization or minimization could be performed, for example, to optimize solve a data-regression or data-classification problem or to optimize a design associated with a response function. A response function can be considered as a subset of a response surface, which is a surface in a vector space of design and performance parameters. A typical example of a design problem that the method and system can be used to solve is that of an airfoil, for which a response function could be the spatial distribution of pressure over the airfoil. In this example, the response surface would describe the pressure distribution as a function of the operating conditions and the geometric parameters of the airfoil.

The use of NNs to analyze physical objects in order to optimize their responses under specified physical conditions is well known. NN analysis is suitable for multidimensional interpolation of data that lack structure and enables the representation and optimization of a succession of numerical solutions of increasing complexity or increasing fidelity to the real world. NN analysis is especially useful in helping to satisfy multiple design objectives. Feedforward NNs can be used to make estimates based on nonlinear mathematical models. One difficulty associated with use of a feedforward NN arises from the need for nonlinear optimization to determine connection weights among input, intermediate, and output variables. It can be very expensive to train an NN in cases in which it is necessary to model large amounts of information.

Less widely known (in comparison with NNs) are support vector machines (SVMs), which were originally applied in statistical learning theory. In terms that are necessarily oversimplified to fit the scope of this article, an SVM can be characterized as an algorithm that (1) effects a nonlinear mapping of input vectors into a higher-dimensional feature space and (2) involves a dual formulation of governing equations and constraints. One advantageous feature of the SVM approach is that an objective function (which one seeks to minimize to obtain coefficients that define an SVM mathematical model) is convex, so that unlike in the cases of many NN models, any local minimum of an SVM model is also a global minimum.

In the SVM approach as practiced heretofore, underlying feature-space co-



A **Hybrid NN/SVM System** offers capabilities greater than those of an NN or a conventional SVM alone.

ordinates or functions must be specified. In the NN approach as practiced heretofore, resampling of data is needed to implement a process, known in the art as model hybridization, in which a superior neural network is generated from the synaptic-connection weight vectors of multiple neural networks that yield local minima with acceptably low errors. What is needed is a machine-learning algorithm that combines the desirable features of the NN and SVM approaches and does not require intimate *a priori* familiarity with operational details of the object to be optimized. Preferably, the algorithm should automatically provide a characterization of many or all of the aspects in feature space needed for the analysis.

A hybrid NN/SVM system (see figure) accepts inputs in the form of parameter values, which are regarded as independent coordinates in an input vector space. In the construction of the SVM, the input coordinates are mapped into a feature space of appropriately greater dimensionality, wherein the coordinates include computed combinations (e.g., powers and/or polynomials) of the input space coordinates. The NN is initially programmed with random synaptic-connection weights and used to construct inner products for the SVM. The inner products are, in turn, used to compute Lagrange multipliers. A training error associated with the connection weights and Lagrange multipliers is calculated. If the training error is too large, one or more connection weights are changed and all of the foregoing (except the initial programming with random weights) steps are repeated. If the training error is not too large, the connection weights and the Lagrange multipliers are accepted as optimal.

An important advantage of this system over a conventional SVM is that the feature-space coordinates that must be specified *a priori* are determined by the NN subsystem. Moreover, the feature-space coordinates are generated by the NN subsystem to correspond to the data at hand; in other words, the feature space provided by the NN subsystem evolves to match or correspond to the data. A feature space that evolves in this manner is referred to as "data-adaptive." The feature-space coordinates generated by the NN subsystem can be easily augmented with additional feature-space coordinates (combinations of parameters) and kernel functions provided by the user.

This work was done by Man Mohan Rai of Ames Research Center. Further information is contained in a TSP (see page 1).

This invention has been patented by NASA (U.S. Patent No. 6,961,719). Inquiries concerning rights for the commercial use of this invention should be addressed to the Ames Technology Partnerships Division at (650) 604-2954. Refer to ARC-14586.